HAL
open science

Systeme a microprocesseur asynchrone basse consommation

D. Rios

To cite this version:

D. Rios. Systeme a microprocesseur asynchrone basse consommation. Micro et nanotechnologies/Microélectronique. Institut National Polytechnique de Grenoble - INPG, 2008. Français. NNT : . tel-00349049

HAL Id: tel-00349049

https://theses.hal.science/tel-00349049v1

Submitted on 23 Dec 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

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

T H E S E

pour obtenir le grade de

DOCTEUR DE l'INPG

Spécialité : Micro et Nano Electronique

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

Présentée et soutenue publiquement

par

David Ríos Arámbula

Le 18 septembre 2008

Titre :

SYSTEMES A MICROPROCESSEURS ASYNCHRONES BASSE CONSOMMATION.

Directeur de Thèse : Marc Renaudin
Codirecteur : Gilles Sicard

JURY

M. Régis Leveugle	, Président
M. Christian Piguet	, Rapporteur
M. Jean-Didier Legat	, Rapporteur
M. Marc Renaudin	, Directeur
M. Gilles Sicard	, Codirecteur
M. Philippe Maurine	, Examinateur

A mi abuelo

Porque gracias a ti soy quien soy

“Ignorance is bless”

Remerciement

Quatre années de travail sont présentées dans ce manuscrit. Thèse réalisée au sein du laboratoire TIMA dans le groupe CIS.

Je souhaite remercier M. Bernard COURTOIS, directeur du laboratoire TIMA au début de cette thèse, ainsi que Mme Dominique BORRIONE, actuelle directrice.

Un remerciement très spécial à Marc Renaudin, qui m'a accueilli dans le groupe CIS pour réaliser cette thèse. J'ai pu apprendre beaucoup grâce à tes conseils surtout grâce à ta passion sur le monde asynchrone, qui je crois viendra tôt ou tard changer la vision des concepteurs de circuit.

Je voudrais également remercier Gilles Sicard, co-encadrant de ma thèse. Tu m'as beaucoup aidé pour que cette thèse se concrétise. C'est un plaisir de discuter avec toi sur des sujets tantôt scientifique que sur des sujets hors du travail.

Je voudrais remercier tous les membres du jury pour avoir participé à la lecture du manuscrit ainsi qu'à la présentation de cette dernière.

M. Christian PIGUET, directeur de recherche au Centre Suisse d'Electronique et de Microélectronique à Neuchâtel (CSEM), et Jean-Didier Legat, professeur à l'Université Catholique de Louvain, pour m'avoir fait l'honneur de participer à mon jury de thèse en tant que rapporteurs.

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

M. Philippe MAURINE, maître de conférence à l'Université Montpellier II, pour avoir accepté d'être examinateur de ma thèse.

Je remercie aussi Laurent Fesquet pour m'avoir aidé à la fin de la thèse.

Un grand merci à toutes les personnes du TIMA et CMP qui on participé indirectement à la réalisation de cette thèse : Patricia, Corine, Ahmed, Fred, Nicolas, Sophie M. (la plus part du temps en vacance mais merci quand même), Lucie, Sabrina, Anne Laure, Isabelle.

Alexandre Chagoya, merci pour ta patience et tous les service que tu m'as fait. Tous les résultats de ma thèse ont été possibles grâce à toi. No por nada somos paisanos !!! Muchas gracias.

Bertrand, tu as vraiment été un des paliers de la réussite de cette thèse, je te remercierais toujours. Ce fut un plaisir de pouvoir commencer mon séjour en France dans le même bureau que toi, la mythique N121!!! J'espère que toutes les expressions en espagnol te servent de quelque chose, et tu pourra peut être les utiliser un jour au Mexique :)

Sophie D. merci de m'avoir aidé a comprendre les outils. Tu possèdes une patience incroyable et une envie de transmettre ton savoir a tout le monde. Tu pourras toujours compter sur moi.

Un grand merci à la dream team !!! Aurelien, Yannick et Fraidy. Vous changerez le monde asynchrone !!!

And of course a big thanks to my enemy mine. You make me see think different now, and I know better your country and your way of life. I will never forget the discussion we had trying to narrow the differences between us. You have been one of the persons with whom I had argued the most. And I really appreciate all the subjects we talk about. Thanks again, you have a friend for life.

Docteur Taha, cette fois si je ne t'oubli pas. Merci d'avoir été là pour m'écouter et pour être un ami. Tu es quelqu'un de spécial, continue avec toutes tes ambitions. Je suis sur que tu réussira dans le futur. En plus avec tes 2 doctorats, le monde devra trembler !!!

Rodrigo !!! Merci pour avoir été un ami, c'est bien de pouvoir compter avec quelqu'un comme toi. Et j'espère qu'on pourra continuer à faire la fête pendant longtemps. Il faudra aussi que tu revienne pour faire du ski !!!

Julien, je suis ravie de t'avoir connu et j'espère que cette amitié restera ici pour toujours. Merci d'avoir vue et revue ma thèse, maintenant il ne reste plus aucune erreur ! Dommage qu'on ne puisse pas dire la même chose de la tienne puisque t'a jamais voulut corriger les énormes erreurs qu'il y avait. Mais bon j'ai essayé de t'aider :)

Pierre, je pense que tu es quelqu'un de formidable. J'ai eu la chance de pouvoir te connaître a la fin de la thèse, et je sais que tu sera toujours un très bon ami. C'est cool de voir que d'autre personnes se préoccupent aussi pour cette planète, et je suis fier qu'on est contribué au changement de la machine à café.

Greg, merci de m'avoir supporté cette dernière année. J'avais vraiment besoin d'un soutient pour terminer cette thèse. J'ai trouvé un ami avec toi, et je te serais toujours reconnaissant pour tout ce qu'on a fait pendant la fin de la thèse. Et c'était un plaisir de t'apprendre à jouer au magic, poker, top spin, guild wars, starcraft et la vie en général :). J'espère qu'on aura encore mil chose a faire, voyages, sortie, soirées… Merci pour tout Greg.

Cédric, t'es une des personnes les plus difficile que je connais. Pas facile de rentrer dans ton cercle !!! Mais une fois dedans ce qui est bien, c'est que c'est pour toujours. Je te considère un de mes meilleurs amis et je le serais aussi pour toujours. Bien sur il faudra que tu m'attende a que je prenne toutes mes photos quand on ira au ski :), mais vois le coter positif, tu ne te fera pas de mal au genoux comme ça. Et pour que tu ne te fâches pas après, je t'aime à toi aussi !!!

Livier, que bueno que estuviste aquí al mismo tiempo que yo. Estar en medio de los franceces es bueno pero después de un rato cansa!!! Me da gusto haber encontrado un amiga contigo. Los viajes que hicimos serán inolvidables, y todas las tonterías que dijimos, jamás las olvidaré. Y bueno al parecer aquí no se acaba la historia, nos seguiremos viendo ahora en la chamba por otro rato. Gracias por todo, me apoyaste cuando más lo necesitaba. Siempre pordas contar conmigo.

Joao, le temps qu'on a passé ici étais court, mais j'ais appris beaucoup de chose avec toi, que ce soit technique ou non. Merci d'avoir été là et de m'avoir apporté un soutient surtout quand j'en ai eu le plus besoin. On se retrouvera un jour pour continuer à discuter sur la vie.

Clem, merci pour tout ce que tu as fait pour moi, cette thèse n'est pas bourré de fautes d'orthographes grâce à toi. Je sais que relire la thèse n'est pas quelque chose de facile et surtout d'avoir a apprendre des concept qui n'on rien avoir avec ta formation. Merci beaucoup pour tout le temps que tu a consacré à ce manuscrit et surtout à tous tes conseils d'ami. Je ne l'oublierais jamais.

Et bien sur merci a tout le groupe CIS: Saeed, Hatem, Frank, Khaled, Jeremy, Oussama, hakim, Gaetan. Ca ma fait plaisir de pouvoir vous connaître à vous tous et de partager tous ces moments, qui rompaient avec la routine de tous les jours.

Merci aussi a tous mes ami, Emile, Rafa, Lydie, Virginie, Alain, Anthony, Yema, Luis, Erick, Mathilde, Jorge, Lorena et aussi a tous ce que j'oubli !

Je souhaite aussi remercier a toute "la banda amena", Dani, Daniel, Humberto, Miguel y Tock. Crecí con ustedes y gracias a ustedes puedo decir que mi vida vale la pena. Soy feliz de saber que tengo amigos como ustedes, y de saber que puedo contar con todos ustedes sin niguna condicion. Igualmente de mi parte, siempre tendrán un amigo y siempre podrán contar conmigo. Lastima que nuestras vidas se separaron después de estar juntos, después de haber hecho y deshecho el mundo

entero con miles de teorías. Pero siempre seremos amigos y siempre seremos Amenos!!! Gracias a todos ustedes por haber sido y ser parte de mi vida. No olviden que mi sueño de que en el futuro volvamos estar todos juntos, no lo olvido y espero que algún día podamos realisarlo.

Un remerciement spécial à Joseph, qui grâce a toi la vie en France fut beaucoup plus facile. Merci pour tout ce que tu a fait pour moi. Et surtout merci pour supporter ma mère !!! (ca ne doit pas être facile :)).

Je remercie également ma mère, qui a réussit à m’emmener jusqu’ici, sans elle tout ceci n’aurait jamais été possible. Je te remercie pour tous les sacrifices que tu as faits pour moi et pour tous ce que tu m’as donné.

Un remerciement pour mon père, qui m’a donné toutes les armes pour arriver jusqu’ici. Te agradesco todo lo que hiciste por mi. Si vine a hacer el doctorado es indirectamente por ti. Tu tienes la culpa de que me haya venido a este país, pero te lo agradesco ya que ha sido la mejor experiencia que he tenido en mi vida. Espero poder llegar algún día a hacer todo lo que tu haz logrado.

Un grand merci a toute ma famille. Gracias a mis abuelos, David, Josué, Leti y Tere siempre están en mi pensamiento. Gracias por todo lo que hicieron por mi.

Finalement le plus grand merci à Kattia, tu m’as soutenu jusqu'à la fin. Te agradesco todo lo que hiciste por mi y todo lo que dejaste por mi. Mi vida aquí en francia no hubiese sido tan magnífica si no hubieses estado a mi lado. Gracias por haber creido en mi. Tu apoyo en todo lo que sucedió aquí en Francia, jamás lo podré olvidar. Eres una mujer exepcional, siempre estarás en mi corazón. Independiente de lo que nos depare el futuro, lo que siento por ti jamás nadie podrá quitarmelo.

Table des matières

Table des matières i

Liste des Figures v

Liste des tableaux ix

Glossaire xi

Résumé xiii

Introduction 1

Chapitre 1 ***Techniques d'estimation d'energie. microprocesseurs asynchrones*** *5*

1 Estimation de l'énergie consommée 5

1.1 Consommation d'énergie dans un circuit 6

1.2 Estimation de l'énergie consommée 11

1.3 Techniques de réduction de la consommation d'un circuit 18

1.4 Conclusion 24

2 Microprocesseurs asynchrones 24

2.1 Histoire 24

2.2 Tendance de la technologie 25

2.3 Microprocesseurs asynchrones 27

2.4 Conclusions 33

Chapitre 2 ***Méthodologie pour la conception de circuits asynchrones qdi basse consommation*** *35*

1 Introduction 35

2 Présentation des différents types de circuits asynchrones 36

2.1 Circuits indépendants de la vitesse (DI) 36

2.2 Circuits Quasi Insensibles aux Délais (QDI) 37

2.3 Circuits indépendants de la vitesse (SI) 38
2.4 Circuits Micropipeline 39
2.5 Circuits de Huffman 40

3 Présentation du flot de conception 40
3.1 Flot de conception 41
3.2 Description du circuit avec le langage CHP 43
3.3 L'outil de conception TAST 44
3.4 Synthèse avec des portes a deux entrées 45
3.5 Projection sur la librairie TAL 46

4 Estimation de l'énergie par le calcul du nombre de transitions 51
4.1 Technique de calcul du nombre de transitions dans un circuit 52
4.2 Présentation de l'outil pour compter les transitions 52
4.3 Estimation de l'énergie 53
4.4 Limites de l'outil 54

5 Conclusion 55

Chapitre 3 analyse d'architectures asynchrones 57

1 Comparaison d'architectures asynchrones. 57
1.1 Définition des critères de comparaison 57
1.2 Synthèse des circuits asynchrones QDI 60
1.3 Comparaison de circuits synchrone et asynchrone 64
1.4 Présentation de différents blocs d'un microprocesseur 65

2 Analyse d'architectures pour réduire la consommation 88
2.1 Génération du signal d'acquittement. 89
2.2 Acquittement par bus 90
2.3 Acquittement par digit 91
2.4 Suppression des signaux de commande. 93
2.5 Structures équilibrées et non équilibrées 94

3 Conclusions 100

Chapitre 4 Asservissement de la tension d'alimentation pour reduire la puissance consommee 103

1 Principe du DVS 103

2 Synchrone et Asynchrone 105

3 Contributions 106

4 Présentation du système 107

4.1 Calcul de la vitesse 108
4.2 Contrôle du régulateur DC/DC 109
4.3 Paramètres du contrôle choisi 110
4.4 Fréquence du signal d'horloge du co-processeur 111

5 Modélisation et simulation du système 111
5.1 Processeur 112
5.2 Régulateur DC/DC 113
5.3 Profil de l'application 115
5.4 Co-processeur 119

6 Résultats obtenus 120
6.1 Régulateur à tension continue 120
6.2 Régulateur par paliers 122

7 Comparaison de la puissance consommée 126

8 Conclusions 127

Conclusion 129

Publications 133

Bibliographie 135

Annexe A 145

Annexe B 148

Annexe C 149

Liste des Figures

Figure 1-1 : Inverseur en logique pseudo-NMOS *8*
Figure 1-2 : Logique Pass-transistor *9*
Figure 1-3 : Chaîne d'inverseurs *20*
Figure 1-4 : Délais, énergie vs capacité *21*
Figure 2-1 : Porte de Muller *37*
Figure 2-2 : Hypothèse temporelle de la fourche isochrone *38*
Figure 2-3 : Modèle équivalent aux circuits QDI et SI *38*
Figure 2-4 : circuit synchrone *39*
Figure 2-5 : circuit micropipeline *39*
Figure 2-6 : circuit de Hauffman *40*
Figure 2-7 : Flot de conception *41*
Figure 2-8 : Flot de conception TAST *45*
Figure 2-9 : Exemple de fusion de cellules standard en cellules complexes *47*
Figure 2-10 : Exemple de fusion de cellules asynchrones standard en cellules complexes *48*
Figure 2-11 : Conflit dans une structure de Muller à 2 entrées *51*
Figure 2-12 : Exemple d'un profil d'activité. *53*
Figure 2-13 : Diagramme d'un inverseur *55*
Figure 3-1 : Exemple de MDD *61*
Figure 3-2 : Table de vérité *62*
Figure 3-3 : Exemple de circuit pour réaliser une opération AND et OR *62*
Figure 3-4 : MDD du circuit présenté et optimisation *63*
Figure 3-5 : circuit en portes standards a deux entrées *63*
Figure 3-6 : Half-Buffer entre de blocs asynchrones QDI *64*
Figure 3-7 : Architecture du MIPS asynchrone *65*
Figure 3-8 : Diagramme de blocs de l'unité d'exécution *66*
Figure 3-9 : Diagramme de blocs de l'unité logique *67*
Figure 3-10 : ordonnancement des entrées pour générer le bloc logique *67*
Figure 3-11 : Diagramme de bloc de l'unité logique synchrone *68*
Figure 3-12 : Énergie consommée en fonction du nombre de bits *70*
Figure 3-13 : Énergie consommée en fonction du nombre de bits *71*

Figure 3-14 : Architecture logarithmique à 5 étages *74*
Figure 3-15 : Diagramme de bloc des blocs de décalage *74*
Figure 3-16 : Diagramme de blocs de l'unité de décalage synchrone *75*
Figure 3-17 : Diagramme de blocs de l'architecture à 3 étages *76*
Figure 3-18 : Profil de consommation de deux instructions *77*
Figure 3-19 : Profil d'instruction avec entrées à temps différent. *79*
Figure 3-20 : Profil d'instruction avec entrées à temps différent. *80*
Figure 3-21 : Diagramme de blocs de l'unité de décalage avec machine à états *82*
Figure 3-22 : Machine à états *82*
Figure 3-23 : Consommation par nombre de bits a décaler. *84*
Figure 3-24 : vitesse de calcul en fonction des bits à décaler *84*
Figure 3-25 : Transitions en fonction de la distance de Hamming pour le circuit asynchrone *86*
Figure 3-26 : Transitions en fonction de la distance de Hamming pour le circuit synchrone *87*
Figure 3-27 : Energie consommée en fonction de la distance de Hamming *88*
Figure 3-28 : Pic de transitions en fonction de la distance de Hamming *88*
Figure 3-29 : Génération de l'acquittement *89*
Figure 3-30 : Acquittement par bus *90*
Figure 3-31 : Acquittement par digit *91*
Figure 3-32 : Exemple d'une structure asynchrone *93*
Figure 3-33 : Demultiplexeur avec et sans commande *94*
Figure 3-34 : Détail du chemin des opérandes dans l'alu *95*
Figure 3-35 : DÉMULTIPLEXEUR équilibré double rails *95*
Figure 3-36 : Démultiplexeur déséquilibré en dual rail *97*
Figure 3-37 : consommation démultiplexeur dual rail *98*
Figure 3-38 : Démultiplexeur équilibré 4 rail *98*
Figure 3-39 : Démultiplexeur déséquilibré 4 rails *99*
Figure 3-40 : consommation des version 2 rails, 4 rails et 2x2 rails *99*
Figure 4-1 : Application avec et sans contrôle DVS *104*
Figure 4-2 : Diagramme de bloc du système *107*
Figure 4-3 : Evolution de la vitesse pendant un changement de tension *109*
Figure 4-4 : diagramme de bloc d'un contrôle PID *110*
Figure 4-5 : Système proposé *112*
Figure 4-6 : schéma de la pompe de charge *114*
Figure 4-7 : Exemple de profil d'application *116*
Figure 4-8 : Consigne calculée. *117*
Figure 4-9 : Exemple de contrôle dynamique *118*
Figure 4-10 : Exemple de la commande pour le régulateur a tension continue *120*
Figure 4-11 : Consigne, Tension en sortie du régulateur et vitesse du processeur *121*
Figure 4-12 : Énergie consommée par : *122*

Figure 4-13 : consigne et vitesse du processeur pour le régulateur 4 bits *124*
Figure 4-14 : Consigne et vitesse obtenue du processeur avec le régulateur 2 bits *125*
Figure 4-15 : Vitesse avec calcul en temps réel de la consigne. *126*
Figure 4-16 : Consommation du processeur avec les différentes architectures de régulateurs *126*

Liste des tableaux

Tableau 2-1 : (transitions|fils) pour différentes tailles du canal *44*
Tableau 2-2 : Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222 pour la bibliothèque TAL *49*
Tableau 2-3 : Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222 pour la bibliothèque TAL2 ..... *49*
Tableau 2-4 : Gain en énergie des cellules TAL *50*
Tableau 2-5 : Gain en énergie des cellules TAL2 ..... *50*
Tableau 3-1 : Répartition des capacité à la sortie de chaque porte *71*
Tableau 3-2 : récapitulatif de l'unité logique *72*
Tableau 3-3 : énergie et temps pour exécuter chaque instruction *79*
Tableau 3-4 : énergie et temps pour exécuter chaque instruction *80*
Tableau 3-5 : Capacité commuté par bloc *83*
Tableau 3-6 : Récapitulatif de l'unité de décalage *85*
Tableau 3-7 : utilisation des instructions en fonction des benchmarks *96*
Tableau 3-8 : nombre d'instructions par bloc *96*
Tableau 4-1 : Plage de tension et fréquence d'utilisation pour quelque processeurs commerciaux *105*

Glossaire

QDI (Quasi Delay Insensitive)	2
CMOS (Complementary Metal Oxide Semi-conductor)	2
DVS (Dynamic Voltage Scaling)	3
PID (Proportional Integrative Derivative)	3
MOS (Metal Oxide Semi-conductor)	7
SOC (System On Chip)	11
RMS (Root Mean Square)	12
DSP (Digital Signal Processor)	15
RTL (Register Transfer Level)	16
ASIC (Application-Specific Integrated Circuit)	16
HDL (Hardware Description Language)	23
VLSI (Very Large Scale Integration)	26
RISC (Reduced Instruction Set Computer)	28
MIPS (Millions Instructions Per Second)	28
BTC (Branch Target Cache)	30
LTPS (Low Temperature Poly-Silicon)	33
TFT (Thin Film Transistor)	33
DI (Delay Insensitive)	36
SI (Speed Independent)	38
CHP (Communicating Hardware Processes)	41
TAST (Tool for Asynchronous circuits SynThesis)	41

MDD (Multi-valued Decision Diagram) 60

PDA (Personal Digital Assistant) 65

PLL (Phase Locked Loop) 105

OS (Operating System) 107

EDF (Earliest Deadline First) 117

Résumé

Cette thèse présente une contribution à la conception de circuits asynchrones Quasi Insensibles aux Délais (QDI) faible consommation. Les circuits asynchrones présentent plusieurs avantages intrinsèques par rapport aux circuits synchrones en terme de consommation. L'absence d'horloge dans ce type de circuit permet en soit de réduire les pics de courant qui réduisent la durée d'autonomie des batteries dans les systèmes portables. La consommation dynamique des circuits asynchrones est aussi réduite par le fait que seules les parties qui sont nécessaires au calcul sont actives pour une tâche donnée. Les circuits asynchrones présente aussi des avantages comme la faible génération de bruit et leur excellente tenue aux variations de tension. Les circuits asynchrones sont relativement récents, ils restent encore dans le domaine de la recherche. L'avancement des recherches sur ce type de circuit ces 20 dernières années font que leurs performances commencent à intéresser le monde industriel. Cependant, le manque de méthodologies et d'outils pour la conception des circuits asynchrone est actuellement un frein à leur intégration dans la vie courante d'un concepteur.

Ce travail présente dans un premier temps, une brève étude des méthodes d'estimation de l'énergie dans les circuits CMOS. Il abordera l'évolution des microprocesseurs asynchrones dans la recherche ainsi que quelques processeurs commerciaux. Dans le deuxième chapitre, la méthodologie proposée sera présentée. Cette méthodologie utilise trois outils qui permettent la synthèse, l'optimisation et l'estimation d'énergie des circuits asynchrones QDI. La conception de ces circuits se fait à partir d'un langage de haut niveau (CHP). Le troisième chapitre expose une étude sur les choix d'architectures lors de la conception des circuits asynchrones QDI en utilisant la méthodologie proposée. Une comparaison avec les équivalents synchrones des architectures étudiées sera aussi montrée. Finalement, le quatrième chapitre présente une technique pour réduire la consommation d'un circuit en régulant la tension d'alimentation de ce dernier. Cette technique utilise un asservissement à boucle fermée pour contrôler la tension d'alimentation.

Introduction

Actuellement la grande majorité des circuits du marché sont des circuits synchrones, leur fonctionnement dépend donc du signal d'horloge. Cette horloge commande ainsi tous les blocs qui composent le circuit. Le signal d'horloge est utilisé pour donner une référence temporelle pour la communication des données dans le système. Ce type de circuits présentent plusieurs avantages, le premier étant la grande connaissance que l'on a sur eux. En effet, tous les logiciels du marché sont conçus pour réaliser des circuits synchrones. Toutes les études sur les caractéristiques et optimisations se sont focalisées sur ce type de circuits ce qui rend en conséquence ces derniers les plus performants. La conception synchrone est aussi une manière de réaliser des circuits qui sont très robustes au bruit et aux aléas. Cependant, ceci est possible que s'ils fonctionnent à la vitesse maximale du bloc le plus lent. Ceci pénalise le reste du circuit qui doit rester inactif la plupart du temps en attendant le prochain front d'horloge. Le deuxième inconvénient étant que tout le circuit commute en même temps, ce qui entraîne des pics de courant très importants. En plus d'augmenter la consommation d'énergie ces pics génèrent des failles de sécurité. Le problème posant le plus d'inconvénient dans les circuits synchrones est la génération de l'arbre d'horloge qui dans les circuits actuels peut représenter une consommation qui peut aller jusqu'à 55% de la consommation totale du circuit. Sans oublier que cet arbre est très compliqué à générer à cause de la forte intégration des circuits actuels.

Les circuits asynchrones ne sont pas nouveaux, ils ont été conçus la première fois dans les années 50. Mais le signal d'horloge qui est utilisé habituellement par les concepteurs de circuits pour fournir une base de temps à tous les composant d'un circuit électrique, a été considéré comme essentiel. Certaines personnes ont considéré que l'horloge est nécessaire pour le fonctionnement d'un circuit. La citation suivante de A. Turing expose bien cette idée:

"We might say that the clock enables us to introduce a discreteness into time, so that time for some purposes can be regarded as a succession of instants instead of a continuous flow. A digital machine must essentially deal with discrete objects, and in the case of ACE[1] this is made possible by

---

[1] ACE : Automatic Computing Engine

the use of the clock. All other computing machines except for human and other brains that I know of do the same. One can think up ways of avoiding it, but they are very awkward..."

Bien que certains chercheurs se soient penchés sur les circuits asynchrones les circuits synchrones étaient plus faciles a concevoir et a implémenter à cette époque, la simplicité de conception des circuits de cette époque a dirigé la recherche vers le synchrone. Plusieurs années ont passé et malgré toutes les études de pointe sur la technologie synchrone, il commence à être très difficile d'implémenter ce type de circuit. De nos jours, les circuits asynchrones reviennent au goût du jour et il existe de plus en plus de recherches dans ce domaine. Bien sûr, avec le retard que l'asynchrone a pris par rapport au synchrone, il faudra encore quelques années pour que le marché arrête de vendre des processeurs en GHz !!!

Actuellement, le développement des méthodologies pour la conception de circuits asynchrones commence à être développé dans la recherche. Les techniques pour estimer l'énergie consommée par les circuits asynchrones se développent également, pour obtenir un ordre de grandeur de la puissance consommée par ce type de circuits. Ce travail a pour objectif de contribuer à la recherche dans la conception de circuits asynchrones. Il se focalise sur la consommation d'énergie de ces circuits et propose une méthodologie de conception qui permet d'analyser et d'optimiser les architectures du circuit en question. Il montrera aussi certain choix d'architectures pour réduire la consommation d'énergie.

Dans le premier chapitre le manuscrit expose les différentes techniques d'estimation d'énergie dans les circuits. Une étude des sources de consommation d'énergie dans les circuits CMOS (Complementary Metal Oxide Semi-conductor) est présentée dans un premier temps. Ensuite, il montre les deux grandes catégories des différentes techniques d'estimation et expose plusieurs techniques en montrant leurs bénéfices et leurs inconvénients. La deuxième partie du chapitre montre les avancées de la recherche sur les microprocesseurs asynchrones ainsi que dans le commerce.

Le deuxième chapitre montre la méthodologie proposée pour la conception de circuits asynchrones QDI (Quasi Delay Insensitive, quasi insensible au délai). Dans ce chapitre, une brève introduction des différents types de circuits asynchrones sera présentée. Il présentera par la suite la méthodologie avec le flot de conception proposé. Ce flot permet de concevoir des circuits asynchrones depuis un langage de haut niveau et donne la possibilité par la suite d'analyser le nombre de transitions qui sont effectuées lors de la simulation du circuit avec un vecteur d'entrée donné. Une estimation de l'énergie est faite grâce aux capacités de charge présente dans les cellules qui composent le circuit. Les optimisations de certaines parties du circuit sont alors possibles grâce au profil d'activité qui est généré. Ce profil donne au concepteur une vision globale des parties les plus sollicitées dans le circuit, ce qui permet la prise de décision pour le choix des architectures adéquates.

Dans le troisième chapitre, ce travail propose une analyse de différentes architectures de circuits asynchrones. Il utilise le flot de conception montré dans le premier chapitre et expose les avantages et inconvénients de certains choix d'architectures lors de la conception d'un microprocesseur asynchrone QDI. Une analyse de la conception des blocs de passage de données, tel que les démultiplexeurs, est présentée à la fin de ce chapitre. Il montrera l'importance du choix de l'architecture en fonction des probabilités de parcours des données dans le circuit.

Le quatrième chapitre expose une technique pour réduire la consommation d'énergie dans un circuit. Cette technique part à la base de la technique DVS (Dynamique Voltage Scaling) avec un contrôle en boucle fermée de la tension d'alimentation. Le système est contrôlé par un coprocesseur qui calcule, avec un contrôle de type PID (Proportional Integrative Derivative, proportionnel intégrateur dérivateur), le régulateur de tension qui fournit la tension d'alimentation. Cette technique est principalement optimisée pour les circuits asynchrones car ils peuvent être alimentés avec une tension qui varie dans le temps sans avoir à changer d'autres paramètres comme dans le cas des circuits synchrones, où changer la tension d'alimentation conduit à un changement de la fréquence d'horloge. Plusieurs résultats sur consommation d'énergie utilisant cette technique seront montrés à la fin du chapitre en utilisant plusieurs types de régulateur DC/DC.

Finalement conclusions et perspectives seront présentées, elles réalisent un bilan des résultats obtenus. Conjointement ce travail présentera les perspectives des travaux à suivre.

Chapitre 1

TECHNIQUES D'ESTIMATION D'ENERGIE. MICROPROCESSEURS ASYNCHRONES

1 Estimation de l'énergie consommée

L'évolution de l'électronique dans notre monde actuel joue un rôle fondamental dans la vie de chaque personne, l'homme étant devenu dépendant de celle-ci. Ordinateurs, téléphones portables, le système de navigation des avions, le contrôle des voitures, des lecteurs DVD, les appareils médicaux, sont devenus banals dans notre vie quotidienne. Cela a été rendu possible grâce aux avancements dans la micro et la nano électronique. Les exigences du marché mondial ne font qu'augmenter. On ne se contente plus d'avoir un téléphone chez soi ou sur son lieu de travail, on a besoin d'appeler lorsque l'on marche, lors de nos déplacements en voiture, en train… L'introduction des téléphones portable n'est pas nouvelle, depuis 1947 on cherchait déjà à intégrer un téléphone dans une voiture. Bien sûr, les batteries nécessaires pour alimenter un tel appareil étaient intransportables dans la vie courante. L'intégration des circuits et les recherches sur les batteries a beaucoup évolué depuis, et il est désormais possible d'appeler avec un téléphone qui fait à peine quelque centimètres (batterie incluse).

Or, la nature humaine fait que l'homme désire toujours plus. On ne se contente plus d'appeler, on a maintenant besoin de voir, d'écouter de la musique, de prendre des photos, des vidéos, de jouer en réseau, tout ceci avec le même appareil qui auparavant n'assurait que les communications entres deux personnes. La consommation d'énergie devient cruciale dans ce type d'application si l'on veut être capable de fournir l'énergie suffisante pour que le portable fonctionne plus d'une journée. Il en est de

même pour les applications à distance ce qui est le cas des réseaux de capteurs. La recherche non seulement en ce qui concerne la microélectronique mais aussi la micromécanique et les technologies de communications sans fils ont contribué au développement des micro-capteurs. Ces micro-capteurs permettent de capturer des données physiques tels que la chaleur, l'humidité, les vibrations et autres, puis de transmettre les données capturées pour que celles-ci soient traitées à distance. Ces réseaux peuvent être utilisés dans de nombreuses applications qui peuvent être militaires, par exemple pour la reconnaissance du terrain, des application environnementales pour détecter des incendie très tôt, jusqu'à des application médicales pour surveiller l'état de santé d'une personne. Or, en général, ces micro-capteurs sont placés sans que l'utilisateur ne connaisse leur position exacte ou dans des endroits d'accès difficiles voir impossibles. En conséquence, il est très difficile de remplacer la batterie qui les fait vivre. Les recherches dans ce domaine sont très poussées car dans certains cas, il est très difficile de changer la batterie de tous les micro-capteurs tous les jours, tous les mois et même tous les ans. Il est impensable de pratiquer une chirurgie tous les deux jours sur un patient qui pourrait avoir certains capteurs dans son corps qui contrôlent sa santé, pour changer la batterie!!! Il est en fait nécessaire que l'autonomie en énergie soit de l'ordre de quelques années et parfois de quelques décennies.

Le contrôle de la consommation des circuits est donc devenu crucial pour l'utilisation et le développement de tous les systèmes actuels. Le marché exige une connaissance approfondie de l'énergie nécessaire à l'alimentation des circuits. Cette exigence a favorisé fortement le développement des méthodologies pour la conception et l'analyse des circuits synchrones. Il existe plusieurs outils commerciaux pour concevoir, analyser et synthétiser ce type de circuits. Cadence, Leonardo, Nanosim, Design vision, Modelsim et bien d'autres outils sont disponibles sur le marché. Les circuits asynchrones, au contraire, ne bénéficient pas de ce large choix pour leur conception. Les outils de conception et d'analyse de ces circuits n'existent que dans la recherche. L'industrie commence à peine à se tourner vers l'asynchrone, mais il faudra encore quelques années pour trouver des outils performants.

A continuation, ce chapitre expose les différentes sources de consommation dans les circuits numériques CMOS. Une brève description de leurs caractéristiques sera présentée. Par la suite il abordera les différentes manières dont la consommation d'un circuit peut être estimé. Finalement les différentes techniques pour la réduction de cette consommation seront exposées.

1.1 Consommation d'énergie dans un circuit

Pour obtenir une forte intégration pour réduire la consommation d'énergie ainsi que pour avoir une plus grande densité des fonctionnalités et des performances d'une même puce, les systèmes

CMOS ont été considérablement réduits en taille au cours des trente dernières années. Mais pour réussir à faire une estimation de la consommation d'énergie d'un circuit, il est nécessaire de comprendre les sources de consommation. Il existe quatre sources de consommation présentes dans tous les circuits électroniques intégrés composés de transistors CMOS. Le travail proposé ne tient en compte que de la consommation des circuits CMOS. Les quatre sources de consommation sont les suivantes :

- Courant de fuite
- Courant de stand-by
- Courant de court circuit
- Courant dans les capacités

1.1.1 Courant de fuite

Cette source de consommation est propre à la technologie utilisée lors de la fabrication des circuits. Elle est constituée de deux éléments, le premier étant le courant de fuite des jonctions dans les transistors. Celui-ci apparaît entre la source ou le drain à travers les diodes de polarisation inverse lorsque le transistor est éteint. Le courant de fuite de ces diodes dépend de l'aire de diffusion du drain et de la densité du courant de fuite, qui est déterminé par la technologie utilisée.

Le deuxième est le courant sous le seuil qui est le courant drain-source du transistor éteint. Il provient du courant de la diffusion des porteurs minoritaires dans le canal pour un transistor MOS (Metal Oxide Semi-conductor) opérant dans la région de faible inversion. Dans le cas d'un inverseur avec une faible tension d'entrée, le transistor PMOS est dans sa région ohmique et le transistor NMOS est bloqué, la tension de sortie est donc dans un état haut. Dans ce cas, si la tension V_{DS} entre le Drain et la Source est de 0V, il existe toujours un courant sur le canal du transistor NMOS éteint à cause du potentiel V_{DD} sur le V_{DS}. L'ordre de grandeur du courant de seuil est fonction de la température, de la tension d'alimentation, de la taille et des paramètres de fabrication pour lequel la tension d'alimentation joue un rôle très important.

Dans les technologies supérieures à 65nm, le courant de seuil est prédominant par rapport aux autres courant de fuites dans les systèmes CMOS. Ce courant est donné par l'Équation 1. Il faudra bien sûr tenir en compte dans les années qui viennent, que ce courant commence à avoir le même ordre de grandeur que le courant dynamique.

$$I_{DS} = K\left(1 - e^{-\frac{V_{DS}}{V_T}}\right) e^{\frac{(V_{GS} - V_T + \eta V_{DS})}{nV_T}}$$

Équation 1 : Courant de seuil dans un transistor CMOS

K et n sont fonctions de la technologie et ŋ est le coefficient de réduction dû à la barrière induite par le drain. La taille des transistors et la diminution de la tension de seuil augmentent le courant de fuite. Dans l'Équation 1, il est clair que cette tension fait augmenter le courant de fuite exponentiellement. Dans un circuit CMOS, les variations de la tension de seuil ainsi que de la taille des transistors lors de la fabrication du circuit feront augmenter le courant de fuite sur les transistors plus petits ou sur ceux dont la tension de seuil est inférieure à la moyenne. Il est important de noter que ce courant de fuite est devenu très important par rapport aux technologies précédentes dans lesquelles ce courant n'était important que dans les systèmes avec des temps d'inactivité trop long.

1.1.2 Courant "stand-by"

La consommation d'énergie en mode standby apparaît par exemple lorsque les deux transistors NMOS et PMOS sont saturés dans un inverseur en logique pseudo-NMOS (voir Figure 1-1), lorsque le drain d'un transistor NMOS alimente la grille d'un autre transistor NMOS dans une logique de "pass-transistor " (voir Figure 1-2) ou bien lorsque l'entrée trois états de la grille d'un transistor MOS fuit vers une valeur entre V_{dd} et la masse. La puissance de "standby" est calculée avec Équation 2.

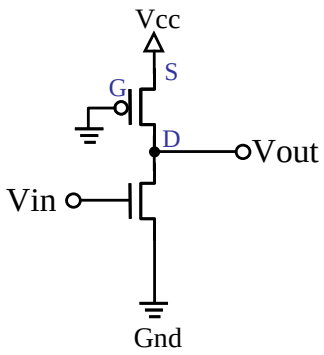


Figure 1-1 : Inverseur en logique pseudo-NMOS

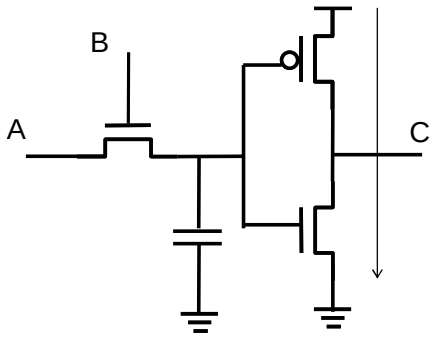


Figure 1-2 : Logique Pass-transistor

$$P = V_{DD} * I$$

Équation 2 : Puissance de Stanby

Avec V_{dd} la tension d'alimentation et I le courant DC qui passe de la source d'alimentation vers la masse.

La dissipation due à la puissance statique est la somme des dissipations de "standby" et de fuite. Les courants de fuite dans un circuit CMOS peuvent être réduits en choisissant la technologie de fabrication adéquate au système proposé. Les courants de "standby" ne sont importants que dans les systèmes CMOS de type pseudo-nMOS, dans les logiques de "pass transistor" et dans les mémoires.

1.1.3 Courant de court circuit

Dans un circuit CMOS, les réseaux de transistors nMOS et pMOS ne conduisent pas en même temps. Leur fonctionnement est complémentaire, et la sortie d'une porte est donnée soit par le transistor nMOS soit par le transistor pMOS en fonction du niveau logique souhaité. Lorsque la sortie doit basculer, les transistors nMOS et pMOS doivent passer d'un état à un autre. Comme les transistors sont complémentaires, ils devront inverser leurs états de « passant » à « bloqué » et vice versa en fonction de la nouvelle valeur d'entrée. Ce type de fonctionnement implique l'existence pendant un moment d'un court circuit entre l'alimentation et la masse lorsque les deux transistors sont dans leur région saturée.

Dans un inverseur, la consommation de court circuit est proportionnelle au temps de montée/descente de la rampe d'entrée, la charge et la taille de la grille dans les transistors. Le courant de court circuit est maximum lorsque la charge à la sortie est nulle; ce courant diminue si la charge en sortie augmente. L'estimation du courant de court circuit peut être obtenue à l'aide de quelques équations qui donnent une estimation du courant en fonction des approximations du modèle du courant

choisi. [VEE 84] présente un des premiers travaux qui a modélisé la dissipation d'énergie de court circuit dans un inverseur CMOS. [HED 87] et [HED 93] présentent des équations avec différentes précisions sans prendre en compte les simplifications de [VEE 84]. Une expression pour l'évaluation de la dissipation de court circuit, basée sur une expression de la courbe de sortie qui considère le courant à travers les deux transistors a aussi été présentée dans [BIS 96]. L'Équation 3 présentée dans [TUR 95] montre la dépendance de la dissipation de puissance de court circuit dans le circuit avec les paramètres de performance comme la taille des transistors, les temps des rampes d'entrée et de sortie et de la charge à la sortie. Cette approche permet de définir la dissipation d'énergie avec une capacité équivalente de court circuit C_{SC}.

$$P_{sc} = \eta C_{sc} V_{DD}{}^{2} f = \eta f \frac{V_{DD}{}^{4}(1-b)^{2}}{6}\left(\frac{\alpha'_{1}\beta_{p} t_{HL(i-1)}}{t_{HL(i-1)} + t_{HL(i)}} + \frac{\alpha''_{1}\beta_{N} t_{HL(i-1)}}{t_{HL(i-1)} + t_{LH(i)}}\right)$$

Équation 3 : Puissance de court circuit présentée dans [TUR 95]

Lors de l'implémentation des cellules logiques avec des transistors MOS, il est important que les tailles des grilles soient choisies de manière à obtenir des temps de montée et de descente qui soit égaux. Si cette condition et remplie, la consommation est d'un peu moins de 15% de la consommation dynamique du circuit [VEE 84]. Si, au contraire, la conception du circuit est réalisée afin d'obtenir des performances plus élevées, et que les grilles des transistors sont choisie très grandes pour alimenter des charges relativement faibles et si les temps de montée sur les entrées sont plus lents, alors la consommation d'énergie sera légèrement pénalisée.

1.1.4 Courant dans les capacités

Bien que la consommation statique dans un circuit CMOS devienne de plus en plus grande avec la forte intégration des circuits actuels, la consommation dynamique continue à être la source de consommation la plus importante dans un circuit CMOS classique. Cette consommation provient de la charge et de la décharge des capacités dans chaque nœud du circuit. La puissance dynamique P consommée dans un circuit CMOS est donnée par l'Équation 4. Avec P la puissance consommée, A le facteur d'activité, C la capacité totale de transition et F la fréquence d'horloge du circuit. Cette équation nous montre que si la capacité C est chargée et déchargée à une fréquence d'horloge F avec une tension d'alimentation V, alors la charge par cycle est de CV et la charge par seconde est de CVF. La charge étant alimentée avec une tension de V, l'énergie dissipée par cycle est de CV^2F. L'activité du système ne pouvant commuter qu'une fois par cycle, la puissance dans le circuit est ½(CV^2F). Dans un circuit synchrone, le facteur d'activité A est employé lorsqu'il utilise du 'clock gating' ou lorsque les bascules ne sont pas actives au cours de tous les cycles. Ce facteur d'activité peut prendre les

valeurs $0 \leq A \leq 1$, avec ce facteur il est possible d'obtenir une moyenne de la consommation de ce type de circuit.

$$P = ACV^2F$$

Équation 4 : Puissance dynamique dans un circuit CMOS

Dans un circuit asynchrone, la consommation n'est pas cadencée par le signal d'horloge. Il est possible d'utiliser la même équation en remplaçant la fréquence F par l'inverse du temps que l'on veut calculer. Autrement dit, pour calculer la puissance consommée par le circuit pendant un temps t, il suffit de diviser par t l'énergie E dissipée par le circuit. Le facteur d'activité dans ce type de circuit représente le nombre de transitions réalisées à la sortie de chaque porte logique. La puissance de chaque porte est ainsi donnée par l'Équation 5:

$$P = \frac{ACV^2}{t}$$

Équation 5 : Puissance consommée par une porte dans un circuit asynchrone

1.2 Estimation de l'énergie consommée

Le procédé de conception d'un circuit inclut la transformation de spécifications comportemental haut niveau à des spécifications au niveau architectural, puis à des spécifications au niveau des portes logiques. Si le concepteur arrive jusqu'au niveau transistor pour se rendre compte après simulation que la puissance consommée est trop élevée, il sera très difficile et coûteux d'effectuer les changements nécessaires pour optimiser le circuit en terme de consommation. Les nouvelles technologies ont permis l'intégration sur une puce de tout un système, donnant naissance à de nouveaux systèmes électroniques appelés Système sur puce ou SOC (System on chip). Les SOCs ont amené de nouveaux défis non seulement lors de la fabrication mais aussi en ce qui concerne la conception de tels systèmes. Pour atteindre les spécifications de conception de ces nouveaux SOCs, le développement de nouveaux environnements de co-conception a vu le jour. Ces outils ont pour objectif d'obtenir une vision globale et rapide des avantages et des inconvénients des différents types d'architectures. Pour réussir à avoir cette vue globale du système, il est nécessaire d'avoir des outils efficace et précis. Un des paramètres les plus importants dans les systèmes actuels est la consommation d'énergie. Il est indispensable d'avoir des outils très performants d'estimation de la consommation d'énergie. [CHAN 95], [RAB 96] et [BEN 97] montrent comment l'analyse et

l'optimisation très tôt dans le flot de conception ont des conséquences importantes sur le gain de consommation. Ceci a entraîné un grand effort dans le développement d'outils d'estimation d'énergie.

Il existe plusieurs façons d'aborder le problème d'estimation de la consommation d'énergie. Ce travail présente les techniques d'estimation au niveau du circuit, au niveau logique et au niveau comportemental les plus représentatives. Ces techniques sont divisées en deux grandes catégories : celles par simulation et celles non simulées.

1.2.1 Techniques par simulation

Les techniques dites par simulation sont basées, en général, sur la simulation directe du circuit ou bien par échantillonnage statistique. L'avantage de ces techniques est qu'il existe des simulateurs qui peuvent générer l'estimation d'énergie, en prenant en compte plusieurs problèmes tels que la génération d'aléas ou les temps de propagation.

1.2.1.1 Simulation directe

Les techniques de simulation directe comme celles présentées dans [QUA 89], [KAN 86] et [TYA 87] simulent les circuits avec des vecteurs d'entrée qui stimulent exhaustivement la totalité du circuit en question. L'avantage est qu'elles sont capables de gérer plusieurs types de modèles, des styles de conception différents, l'utilisation d'horloge multi-phase dans les circuits synchrones ainsi que l'utilisation de sortie à trois états. Le problème de ce type de simulation est qu'il est nécessaire de donner en entrée un vecteur représentatif pour stimuler tous les blocs du circuit, ceci n'est pas forcément facile à générer. Ce vecteur peut être énorme dans les circuits actuels et dépend aussi bien des applications ainsi que de l'environnement du système [RAJ 94]. Ceci engendre un autre problème lors de la simulation. Étant donné la taille du vecteur d'entrée et la complexité et la taille des systèmes d'aujourd'hui, le temps de calcul ainsi que la taille de la mémoire nécessaire pour effectuer cette simulation commencent à devenir excessif avec les ordinateurs actuels.

Huang a proposé un simulateur pour estimer la consommation au niveau des transistors [HUA 95]. Ce simulateur utilise des algorithmes de simulation qui génèrent des événement dans le temps (basé sur des tables de modèle de composants simplifiés, la partition du circuit et sur les itérations pas à pas non linéaires) qui diminuent le temps de calcul de deux ou trois ordres de grandeur par rapport au logiciel SPICE tout en conservant une précision de 10%. L'outil PowerMill fournit aussi des informations détaillées sur la consommation instantanée, la consommation moyenne et la consommation RMS (Root Mean Square, racine carrée de la moyenne du carré). Il calcule aussi les

courants de court circuit, les courants dans les capacités, les courants transitoires et les courants de fuite.

Il existe aussi les logiciels commerciaux tels que Verilog-XL Turbo ou ModelSim, basés sur une simulation au niveau des portes. Ce type de simulateur peut donner la consommation des circuits avec des vecteurs d'entrées générés par le concepteur. Pour réaliser l'estimation de la consommation, ces simulateurs utilisent des macro-modèles qui ont été construis pour les portes des librairies qui composent le circuit. Ces macro-modèles peuvent contenir des informations sur la consommation des portes, les délais des temps de réponse des portes et toutes information relative au fonctionnement de la porte. Une analyse détaillée du temps au niveau des portes est aussi nécessaire. Le problème de ces simulateurs est qu'ils dépendent fortement des modèles des portes logiques utilisés et de la précision des capacités décrites dans le modèle de la porte. Néanmoins, la simulation est réalisée en général trois à quatre fois plus rapidement que les simulations SPICE.

L'estimation de l'énergie par simulation dans les circuits asynchrones ne peut pas être réalisée de la même façon que dans les circuits synchrones. En effet, le manque de signal d'horloge dans ce type de circuits et les opérations qui sont réalisées de manière concurrente dans le circuit et avec son environnement rendent impossible l'obtention d'une référence de temps pour mesurer l'énergie consommée. Une autre façon d'aborder le problème est d'estimer l'énergie consommée par instruction. P. Kudva dans [KUD 94] propose de mesurer l'énergie moyenne par transition à la sortie, en effectuant une pré simulation SPICE de quelques macro-cellules. Cette technique a été utilisée dans des circuits asynchrones avec un protocole en deux phases.

1.2.1.2 Simulation hiérarchique

Les simulations hiérarchiques utilisent plusieurs simulateurs de manière hiérarchique. Les simulateurs peuvent être situés au niveau de l'architecture du circuit, au niveau des portes ou bien au niveau du circuit. Avec cette approche, il est possible d'obtenir une estimation avec un bon compromis précision/efficacité. Vanoostende dans [VAN 93] présente une méthodologie d'estimation d'énergie avec deux simulateurs hiérarchisés. Le premier réalise une simulation au niveau du circuit. Le deuxième observe l'influence de chaque nœud simulé pour en déduire la précision avec laquelle le premier simulateur doit agir. Un autre exemple de simulation hiérarchique est montré dans [GEO 94]. Dans cette méthodologie, deux simulateurs Entice et Aspen sont utilisés. Le premier réalise une analyse de l'activité du circuit et le deuxième réalise une caractérisation de la puissance au niveau des données.

1.2.1.3 Simulation Statistique

Les simulateurs statistiques utilisent l'avantage des probabilités et de la statistique pour réaliser des simulations rapides avec un facteur de confidence très élevé, en général supérieur à 95%. L'idée de tels simulateurs est de générer de manière aléatoire les vecteurs d'entrée pour simuler un circuit. [MEI 00] et [NAJ 91] utilisent les probabilités pour mesurer la consommation d'énergie de chaque porte logique. La puissance totale consommée est donnée par la somme de la consommation de chaque porte. [NAJ 91] montre qu'estimer la consommation de chaque porte avec des vecteurs d'entrée aléatoire, est très coûteux. Pour pallier ce problème, [BUR 93] présente une simulation de type monte carlo dans laquelle les vecteurs sont générés aléatoirement et la consommation est mesurée à l'aide d'un simulateur de manière globale pendant un nombre de cycle d'horloge T. La base de cette technique étant le théorème de la limite centrale qui permet de dire que lorsque T s'approche de l'infini, la densité d'échantillonnage tend vers une courbe normale. La simulation se déroule jusqu'à ce que la précision voulue soit atteinte, et en général le temps de simulation est largement inférieur à celui de [MEI 00] ou [NAJ 91].

L'Équation 6 donne le nombre d'échantillon à mesurer pour obtenir une estimation de la consommation avec une erreur de ε, une moyenne d'échantillonnage de η, un facteur de confidence 1-α et une déviation standard σ. En général un échantillonnage de 30 à 50 échantillons permet d'obtenir une moyenne de densité dans la majorité des circuits.

$$N > \left(\frac{t_{\frac{\alpha}{2}} \sigma}{\varepsilon \eta} \right)^2$$

Équation 6 : Nombre d'échantillon pour une estimation avec une erreur ε.

La simulation Monte Carlo ne peut pas être utilisée directement pour des circuits séquentiels à cause du problème transitoire initial, [CHO 95] montre le problème et présente une solution pour étendre cette simulation à des circuits séquentiels. Pour ceci, il est nécessaire de choisir correctement les états initiaux et la taille des périodes d'initialisation (warmup periods). [NAJ 95] propose une autre solution pour estimer l'énergie consommée par les circuits séquentiels. Dans ce travail certaines séquences des vecteurs d'entrée sont générées aléatoirement, ensuite les statistiques sur les sorties des verrous (latch) sont récupérées après simulation. L'avantage dans cette technique est qu'il est possible de choisir le degré de précision de l'estimation

1.2.2 Bilan des techniques par simulation

Les différentes techniques par simulation présentent des avantages qui sont principalement des résultats fiables quand à la précision de l'énergie estimée. Le grand désavantage de ce type d'estimation est qu'il est en général nécessaire de bien modéliser les circuits pour obtenir une estimation fiable. Un deuxième inconvénient est que certaines simulations nécessitent beaucoup de temps ainsi que de ressources matérielles pour réaliser la simulation.

1.2.3 Techniques par non simulation

Estimer la consommation d'énergie dans un circuit peut aussi se faire en se basant sur des méthodes par non simulation. Celles-ci consistent à obtenir des modèles de puissance pour les blocs qui constituent le circuit.

1.2.3.1 Niveau comportemental

L'idée de cette technique est d'obtenir des modèles de consommation de puissance des blocs qui composent le circuit. Ces modèles sont donnés par la librairie utilisée. La consommation est calculée par le concepteur de la librairie en faisant une simulation qui utilise généralement des données d'entrée pseudo aléatoire et considère les capacités moyennes par cycle d'horloge. Ces modèles sont inclus dans les spécifications de la librairie et peuvent être utilisés pour estimer la consommation d'un circuit qui utilise des éléments de cette librairie.

Les librairies incluent plusieurs paramètres pour les modèles de puissance. Les modèles peuvent être paramétré en fonction du nombre de bits d'entrée, pour donner ainsi la puissance consommée par une structure en fonction du nombre d'entrée. Ceci peut être le cas de structures telles que les additionneurs, les blocs de décalage, les multiplieurs et autres blocs, dans lesquels la consommation est linéaire, quadratique ou exponentielle en fonction du nombre de bits d'entrée. Les informations sur les délais, la taille et la consommation interne sont pré-calculées dans le modèle de puissance, ce qui permet aux simulateurs d'estimer la puissance des circuits.

[POW 95] présente un modèle de puissance pour des algorithmes de DSP (Digital Signal Processor) et des architectures implémentées dans des circuits dédiés. Ils analysent l'impact du style de conception et celui des entrées/sorties. La puissance est estimée en terme de paramètres haut niveau, tels que la taille des mots et le nombre d'éléments de calcul. Cette méthode permet de comparer différentes architectures et algorithmes très tôt dans le flot de conception. [LAN 93] présente une technique d'estimation de la puissance au niveau des algorithmes et des architectures. Cette

technique est basée sur des statistiques de moyenne, de variance et d'autocorrélation. Elle se différencie des outils qui estiment la puissance au niveau des portes car elle utilise les modèles de puissance des blocs haut niveau déterminés par les statistiques des données en entrée.

1.2.3.2 Estimation au niveau RTL

Plus récemment, plusieurs travaux se sont concentrés sur l'estimation de la puissance consommée au niveau RTL (Register Tranfer Level). En effet l'adoption des méthodologies de conception de circuit ASIC (Application-Specific Integrated Circuit) au niveau RTL et l'augmentation de l'intégration des SOC demandent de nos jours une précision et une efficacité sans précédent aux outils d'estimation de puissance au niveau RTL. Les outils qui estiment les performances tels que la puissance consommée, la taille du circuit et la "testabilité" à des hauts niveaux d'abstraction permettent une prise de décision adaptée lors de la conception des circuits.

Les estimations au niveau RTL peuvent être classées en trois catégories : les techniques analytiques, les techniques de macro modèles basé sur des caractérisations et les techniques d'estimation basé sur des synthèses rapides. En fonction des parties du circuit à estimer (blocs arithmétique, logique de contrôle, mémoires, réseau d'horloge et entrées/sorties) certaines techniques d'estimation sont mieux adaptées que d'autres.

Chacune de ces trois techniques présente des avantages particuliers qui seront présentés dans les paragraphes ci après.

1.2.3.2.1 Modèle de puissance analytique

Ces techniques corrèlent la consommation de puissance pour mesurer la complexité du circuit en utilisant très peu d'information des spécifications fonctionnelles. Ces techniques sont très utiles tôt dans le flot de conception. Un des premiers travaux de ce type est présenté dans [MUL 91]. L'inconvénient majeur était que le concepteur devait introduire certains paramètres comme le nombre de portes, l'activité du circuit, etc. Ces paramètres étaient estimés par le concepteur ce qui apportait une certaine incertitude sur le résultat final. Bien que ces techniques soient en général très susceptibles de produire des erreurs, elles sont très bien adaptées à certaines parties du circuit dans lesquels les paramètres sont faciles à obtenir (mémoires et réseau d'horloge). Une sous classe de ces techniques appelée approche par information théorique (information-theoric approach), estime l'activité moyenne et les facteurs de capacité pour des bloc logiques basés sur l'entropie de leurs signaux d'entrée et de sortie [MAR 95] et [GRU 00]. D'autres travaux incluent l'effet des compromis taille-délai-puissance

effectués pendant la synthèse logique [CHE 98], et la prédiction de la puissance consommée dans les interconnexions [BUY 01].

1.2.3.2.2 Macro modèles basés sur la caractérisation

L'idée est d'obtenir et de caractériser des blocs bas niveau obtenu à partir des modèles RTL. En général, ceci est réalisé avec des outils qui calculent la puissance au niveau portes ou au niveau transistor avec des séquences d'entrée dit "d'entraînement". Avec ces données un macro modèle ou un modèle de boîte noire est construit. Ce modèle décrit la consommation du bloc en fonction de quelques paramètres significatifs. [POW 90] montre les premiers travaux qui caractérisaient les macro blocs avec une valeur de puissance constante. La recherche sur cette technique s'est plutôt orientée vers la dépendance de la puissance avec les statistiques des signaux d'entrée [LAN 95], [BEN 96] et [GUP 00]. [BOG 98a] et [BOG 98b] présentent comment il est possible d'améliorer les macro modèles basés sur la caractérisation en utilisant des méthodes d'entraînement de données, en insérant des erreurs lors de la construction des modèles.

1.2.3.2.3 Technique de Synthèse rapide

Cette technique réalise une synthèse limitée de la description RTL du circuit, cette synthèse est effectuée beaucoup plus rapidement que la synthèse logique et le procédé de mappage technologique. L'idée est de mapper le circuit dans une "Meta-librairie" qui consiste d'un nombre réduit de portes (beaucoup plus réduit que la librairie de portes standard). Le fichier de description des portes (netlist) résultant est utilisé pour réaliser l'estimation au niveau des portes avec des techniques de simulation ou statistiques. [LLO 98] utilise cette technique pour décomposer plusieurs fonctions et les regrouper dans un groupe de primitive de puissance, lesquelles ont été préalablement analysées en simulation du point de vue de l'activité des entrées et des sorties pour obtenir la puissance consommée.

1.2.3.3 Conclusion

Les techniques dites par non simulation, présentent principalement l'avantage d'être très rapide par rapport aux techniques par simulation. Elles permettent de donner une estimation très tôt dans le flot de conception, ce qui permet au concepteur de changer les structures qui composent le circuit sans avoir à descendre très bas dans la conception du circuit.

L'inconvénient de ces techniques est qu'elles ne donnent pas une précision aussi élevée que les techniques par simulation. Bien que de nos jours, les techniques par non simulation délivrent des résultats très performants (erreur inférieure à 5%) qui peuvent être utilisés lors de la conception des

circuits. En conséquence, il est nécessaire définir les besoins lors de l'estimation d'énergie d'un circuit pour favoriser le temps ou la précision requise.

1.3 Techniques de réduction de la consommation d'un circuit

Au début du boom de la production des circuits intégrés dans les années 80, les efforts des concepteurs et de la recherche se sont focalisés sur les performances d'intégration et de vitesse. En effet, les concepteurs cherchaient à respecter des contraintes de surface et des performances qu'un circuit allait offrir. Une fois le circuit conçu, la question qui se posait était de savoir comment il allait être alimenté. Or l'augmentation de la consommation d'énergie dans les circuits intégrés est devenue un des problèmes principaux dans l'industrie des semi conducteurs. L'excès de dissipation d'énergie dans un circuit est responsable du réchauffement excessif dans les systèmes, et cette chaleur peut être la cause d'erreurs logicielles voire même de dégâts permanents. La consommation limite aussi la durée de la batterie dans les systèmes portables. Ceci a provoqué la nécessité de trouver de nouvelles méthodologies de conception ainsi que de développer des outils performants pour la conception basse consommation dans les circuits actuels.

Quatre sources de consommation ont été présentées dans la section 1.1. De ces quatre sources, la consommation dynamique continue à être la source la plus importante. Si un circuit est constitué de n nœuds ayant une capacité de C_i avec une tension d'alimentation V_{dd} et D_i étant la moyenne des transitions dans le nœud i, alors la puissance moyenne dans un circuit est donnée par l'Équation 7. De cette équation, trois techniques pour réduire la consommation dans un circuit intégré peuvent être distinguées. La première étant la diminution de la tension d'alimentation, la deuxième la réduction de la capacité des nœuds dans le circuit et finalement la réduction du taux d'activité. Par la suite, une brève présentation des techniques les plus utilisées est présentée. Une recherche plus approfondie sur les techniques de réduction de la consommation d'énergie peut être trouvée dans [PIG 04]

1.3.1 Réduction de la tension d'alimentation

La relation quadratique de la tension d'alimentation avec la consommation d'énergie rend la réduction de la tension d'alimentation la technique la plus efficace pour réduire l'énergie consommée dans le circuit. Si la tension est diminuée par deux, alors la puissance consommée sera réduite par quatre. Cette relation quadratique est si intéressante qu'il est parfois conseillable d'augmenter la capacité totale du circuit ou du taux d'activité pour réduire la tension d'alimentation. L'avantage de cette technique est que la réduction de la consommation se fait sur tout le circuit. Il suffit juste de réduire la tension pour obtenir une réduction d'énergie sur l'ensemble du circuit. Ce qui n'est pas

forcément le cas pour les techniques de réduction de la capacité ni pour celles du taux d'activité. Or, cette technique a aussi ses inconvénients, la vitesse de fonctionnement du circuit est la plus touchée lorsque la tension d'alimentation est réduite. En effet, les délais dans les portes augmentent considérablement lorsque la tension d'alimentation V_{dd} s'approche de la tension de seuil V_t de la technologie utilisée. Cette technique est généralement limitée à une réduction de deux ou trois fois V_t.

Il est alors évident que pour réduire la tension d'alimentation V_{dd} sans perte de performances de vitesse du circuit, il est nécessaire de réduire la tension de seuil V_t. Or, cette tension est donnée par la bibliothèque de portes logiques utilisée. Pour réduire cette tension, il est donc nécessaire d'avoir une bibliothèque spéciale. La tension minimum V_t dépend des marges de bruit réalisables ainsi que du contrôle des courants de fuites. La tension V_t optimum est fonction du gain en courant des portes CMOS dans une région faible tension ainsi que du contrôle des courants de fuites. Les courants de fuites lors de la réduction de la tension de seuil V_t sont en fait un problème auquel il faut faire attention. On peut estimer une augmentation d'un ordre de grandeur à chaque fois que cette tension change entre 80mV et 100 mV [DAV 95]. En règle générale, le courant dans le transistor bloqué ne devrait pas dépasser deux à trois ordres de grandeurs le courant du transistor passant. Ceci limite la tension V_t à plus ou moins 0,3 V.

Il faut aussi noter que les délais dans les portes peuvent changer d'un facteur 3 avec une variation de la tension V_t de +/- 0,15V pour une tension d'alimentation de 1V. Cette variation provoque un problème dans les circuits synchrones pour lesquels il est nécessaire d'adopter des techniques pour contrôler cette variation de V_t. [KOB 94] montre une technique dite d'auto ajustement de la tension de seuil qui réduit cette variation a +/- 0,05 V avec une tension V_{dd} de 1V.

Le principal problème de la variation des délais dans les portes d'un circuit synchrone est que l'horloge doit être calculée pour assurer le chemin critique le plus long. Dans un circuit synchrone, la fréquence de l'horloge est choisie en fonction du pire des cas. Or, les variations dans le délai dues aux variations de V_t lorsque cette tension a été réduite, ne sont pas forcément prises en compte. En revanche, les circuits asynchrones fonctionnent même si les délais dans les portes varient de manière dynamique. Cette propriété, les rend parfaits pour l'utilisation de cette technique. En effet, dans un circuit asynchrone il est possible de changer la tension d'alimentation V_{dd} sans avoir aucune manipulation supplémentaire.

1.3.2 Réduction de la capacité

La réduction de la capacité dans les nœuds du circuit est la deuxième technique utilisée pour réduire la consommation. La capacité dans un circuit provient dans un premier temps de la somme des

capacités d'entrée des portes logiques qui constituent le circuit. La deuxième source de capacité étant les interconnections entre portes logiques dans le circuit final.

L'estimation de la capacité des portes dans un circuit est très difficile puisqu'elle implique d'avoir à réaliser le circuit complet. Or, réduire la capacité dans le circuit tôt dans le flot de conception au niveau logique ou comportemental, devient très difficile puisque ceci impliquerait d'estimer les capacités de charge de structures qui n'ont pas encore été implémentées. Cependant, il est possible dans certain cas, de caractériser préalablement certaines structures comme celle des additionneurs, multiplieurs, mémoires etc. Il est possible, néanmoins, d'avoir des estimations sur la capacité des circuits logiques à l'aide de modèles analytique en fonction du nombre d'entrées et de sorties dans le bloc, ou bien en fonction de la complexité du circuit dans le cas des machines à états.

En ce qui concerne les interconnexions, l'estimation de la capacité de charge dans les portes est encore plus difficile car cette information n'est donnée qu'après l'étape du placement routage. Il est en effet nécessaire de connaître toutes les informations sur la manière dont le circuit est implémenté. C'est seulement après le placement routage du circuit que les distances entre les fils de connexion sont connues. Cependant, quelques techniques pour estimer ces valeurs sont proposées dans [PED 91] et [PED 89].

Une des stratégies pour réduire la capacité totale dans un circuit consiste à réduire la taille des transistors. Bien que dans plusieurs applications la réduction de la taille du circuit donne une réduction de la consommation de la puissance, dans certains cas le circuit le plus petit n'est pas celui qui consomme le moins [SAP 95]. Cette réduction a un coût puisque réduire la taille des transistors, réduit aussi le courant, ce qui réduit en conséquence la vitesse de fonctionnement. Il est peut être préférable de réduire la tension d'alimentation V_{dd} au lieu de réduire la capacité qui, elle, aussi aura un impact sur les performances du circuit. La technique de réduction de la capacité peut être utilisée lorsque la vitesse de fonctionnement n'est pas une priorité.

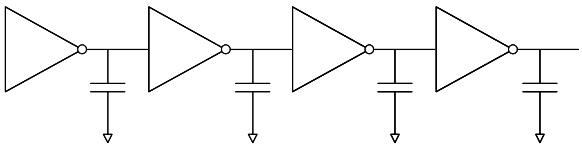


Figure 1-3 : Chaîne d'inverseurs

Puisque la capacité de charge est causée par la capacité dans la grille des transistors, il est possible de réduire la consommation en réduisant la taille des transistors. Ceci a néanmoins des conséquences sur la vitesse de fonctionnement du circuit. Le compromis entre vitesse de

fonctionnement et consommation d'énergie peut être montré en utilisant une chaîne d'inverseurs chargé uniformément comme ceux représentés dans la Figure 1-3.

La Figure 1-4 montre le délai, l'énergie et le produit énergie-delai en fonction de la contribution de la capacité des transistors dans la charge totale. La charge sera plutôt une capacité de charge pour les transistors petits, et sera plutôt une capacité due à la grille des transistors pour les transistors à grande taille. Dans le cas des transistors de faible taille, l'énergie consommée provient de la charge et de la décharge de la capacité de la charge de sortie. Les délais dans les portes sont inversement proportionnels à la taille du transistor, si la taille du transistor augmente, le produit énergie-délai est amélioré. Pour les grands transistors, les portes sont limitées par leur propre charge. Réduire la taille des transistors améliore alors le produit. Le point optimal de fonctionnement est lorsque la charge dans les transistors est la même que la charge dans les fils. [CHA 92a] montre que la réduction de la taille des transistors entraîne une réduction de la puissance consommé.

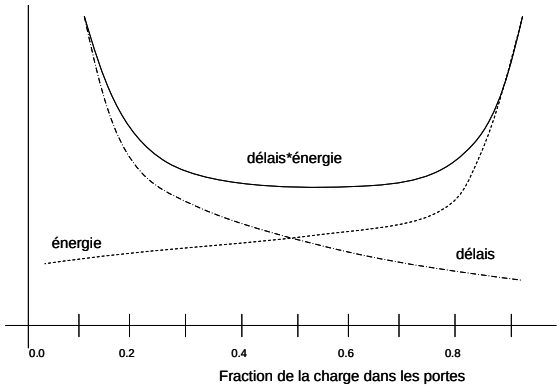


Figure 1-4 : Délais, énergie vs capacité

L'interaction entre la taille, les délais et la puissance peut être très complexe. Une façon plus astucieuse de réduire la capacité totale du circuit est de partager le plus de ressources dans le circuit. Le parallélisme est aussi une solution lorsqu'il est souhaitable de conserver le même débit de sortie en diminuant la tension d'alimentation V_{dd} [SEN 97].

1.3.3 Activité du circuit

La meilleure manière de réduire la consommation dans un circuit est sans doute de ne pas le faire fonctionner. Cette affirmation semble un peu radicale, mais c'est ainsi que le concepteur peut gagner en puissance consommée dans un circuit. En effet, moins un circuit réalise d'opérations, plus sa consommation sera réduite. Il faut donc diminuer le nombre de transitions réalisées par un circuit pour réduire l'énergie gaspillée. Il faudra faire attention dans les technologies nouvelles inférieures a 90nm, car les courants de fuite dans ces technologies commence à être très grand. Aussi, la taille des circuits

augmente considérablement et l'écart entre la consommation statique et la consommation dynamique se rétrécit de plus en plus. Dans un circuit synchrone il existe deux facteurs pour calculer l'activité du circuit. Le premier est la fréquence de l'horloge qui marque la périodicité moyenne avec laquelle les données arrivent, et le second, E qui détermine le nombre de transitions à l'arrivé des données. Pour les circuits asynchrones, l'activité du circuit peut être mesurée en nombre de transitions dans une fenêtre de temps ou bien par le nombre de transitions pour réaliser une tâche. Dans un circuit ayant n nœuds, l'énergie moyenne consommée peut être calculée avec l'Équation 7.

$$E_{moy} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{n} C_i D_i$$

Équation 7 : Énergie moyenne dans un circuit asynchrone

Avec C_i la charge du nœud, D_i le nombre de transitions, V_{dd} la tension d'alimentation et n le nombre de nœuds. Si les courants de fuites sont faibles, le circuit ne consommera de l'énergie que lorsqu'il est actif, Pour réduire le facteur d'activité D_i, il est possible d'éteindre certaines partie du circuit qui sont inactives, soit en conservant les entrées dans un état stable, soit en utilisant des technique tels que le clock gating pour les circuits synchrones. Le clock gating est une technique qui permet d'utiliser plusieurs sources d'horloges qui peuvent être activées ou désactivées. Lorsque l'horloge est désactivée, les registres de sortie n'effectuent aucune transition, ce qui réduit la consommation dynamique du circuit. [EMN 00]. [SRI 96] montre qu'il est aussi possible d'appliquer des stratégies de prédiction pour éteindre les processeurs soit en éliminant le signal d'horloge soit en éteignant complètement le processeur. La technique de prédiction proposée offre une réduction beaucoup plus importante que les techniques où le système est éteint après un certain temps d'inactivité.

1.3.3.1 Stratégies de contrôle de la consommation

Il existe plusieurs techniques pour contrôler la consommation d'énergie qui peuvent être utilisées à différents niveaux d'abstraction. Par la suite, quelques techniques sont présentées.

1.3.3.1.1 Conception du système

Au niveau de la conception du système il est possible de contrôler certaines parties du circuit pour les éteindre ou les allumer. Il est aussi possible de contrôler la tension d'alimentation de certains blocs en fonction des besoins de performance. Il est aussi possible d'utiliser des circuits adiabatiques. Dans [ATH 94] l'auteur propose de réutiliser l'énergie qui a été fournie pour la recycler à la source d'alimentation en utilisant des circuits adiabatiques. Une application plus récente [HAU 95] montre

comment il est possible de réaliser un circuit qui permet de récupérer de l'énergie avec une méthode adiabatique. Une autre approche pour réduire la consommation se situe au niveau de la compilation des programmes dans les processeurs. En effet, si le compilateur prend en compte le matériel disponible, il est possible de réaliser certaines optimisations [TIW 94]. [CHA 92b] montre comment en utilisant des transformations de type algébrique, de contrôle, opérationnelles et d'élimination de redondance, il est possible de réduire jusqu'à un ordre de grandeur la consommation du système. Les accès à la mémoire pendant l'exécution d'une instruction dans un microprocesseur ont un impact très important sur la puissance consommée. L'auteur de [WUY 94] présente une optimisation en effectuant des transformations au niveau des spécifications des signaux ou du traitement des données.

1.3.3.1.2 Synthèse comportementale

La synthèse comportementale consiste à générer un circuit au niveau transfert de registres (RTL) depuis une spécification de type comportemental. Elle génère une vue structurelle du chemin de données et une vue logique de l'unité de contrôle du circuit. Le chemin de données est le chemin que les données utilisent pour réaliser une tâche contrôlée par l'unité de contrôle à travers un ensemble d'unités fonctionnelles connectées entre elles et des unités de direction des données (multiplexeurs, bus de données). La synthèse comportementale est divisée en trois parties: allocation, affectation et ordonnancement. L'étape d'allocation détermine combien d'instances, parmi les ressources disponibles, seront utilisées. L'affectation donne le type de ressources utilisées pour chaque opération. Finalement, l'ordonnancement indique quand est ce que chaque opération sera réalisée.

A ce niveau, il est possible de réaliser quelques optimisations pour réduire la consommation du circuit. L'idée fondamentale dans ces optimisations est de réduire soit le nombre de cycle d'horloge dans un circuit synchrone ; soit d'optimiser le nombre de ressources utilisées pour exécuter une tâche. [CHA 92a] montre qu'il est intéressant d'augmenter le nombre de ressources dans le circuit, même si ce dernier augmente en taille et en conséquence en capacité totale, en réduisant la tension d'alimentation pour atteindre la vitesse initiale. Cette étude ne change pas la tension d'alimentation du circuit de manière dynamique, elle augmente tout simplement le matériel disponible pour augmenter le débit des données de sortie.

1.3.3.1.3 Synthèse logique

La synthèse logique établit une relation entre une spécification en HDL (Hardware Description Language en verilog ou VHDL) et un fichier de description en portes logiques (netlist). La synthèse logique donne en sortie un fichier de description en portes logiques (netlist) en prenant en compte l'optimisation de certains objectifs. Les spécifications d'entrés dans un logiciel de synthèse logique

peuvent être des représentations de logique à deux niveaux, des réseaux multi-niveaux booléens et des machines à états. En fonction des spécifications d'entrés, de l'implémentation cible, de la fonction objectif (taille, délai, puissance) et des modèles de délais utilisés, plusieurs techniques sont utilisées pour transformer et optimiser la description RTL initiale.

Lorsque les choix aux niveaux systèmes, architecturaux et technologiques sont opérés, la capacité commutée totale du circuit détermine la puissance du circuit. Il existe plusieurs techniques pour optimiser la consommation d'énergie lors de la synthèse logique [MON 95], [TIW 95] et [LEN 95].

1.4 Conclusion

Cette partie montre un aperçu des techniques d'estimation de l'énergie consommée par les circuits CMOS actuels. Elle présente plusieurs de ces techniques et montre les différentes méthodes d'estimation d'énergie utilisées dans le monde actuel. Il existe un nombre beaucoup plus grand de techniques et de méthodes, et ce serait impossible de toutes les dénombrer, néanmoins les grandes lignes de recherche ont été présentées. Conjointement, cette partie présente les stratégies utilisées actuellement pour réduire la consommation dans les circuits.

2 Microprocesseurs asynchrones

2.1 Histoire

Il existe un nombre considérable de microprocesseurs et de micro contrôleurs ainsi que les récents DSP (Digital Signal Processor) sur le marché. Les prédictions sur la capacité de traitement que ces dispositifs pourront atteindre sont impressionnantes. Pour arriver à ceci, les microprocesseurs ont traversé plusieurs étapes qui ont suivi l'évolution de la micro electronique dans le monde. Depuis le premier processeur 4004 de 1971 jusqu'à l'actuel Core2 Quad Processor , les choses ont beaucoup changé. Le premier processeur d'Intel le 4004, présenté sur marché le 15 novembre 1971, possédait une caractéristique unique pour son temps : l'horloge dépassait les 100kHz, disposait d'un bus de 4 bits et pouvait gérer jusqu'à 640 octets de mémoire. C'était une merveille qui pouvait réaliser une quantité de tâches énorme pour l'époque.

Peu de temps après, le 1er avril 1972, Intel annonçait une version améliorée de son prédécesseur. Il s'agissait du 8008, qui avait comme principale nouveauté un bus de 8bits, et une

mémoire adressable jusqu'à 16Ko. De plus y étaient incorporé 3500 transistors, presque le double du 4004. Il est considéré comme le prédécesseur du processeur qui a servi de cœur au premier ordinateur personnel. Ce n'est que deux ans après, qu'Intel annonçait la sortie si attendue de l'ordinateur personnel : Altair. Son nom provenait de l'une des destinations du vaisseau Enterprise de l'un des épisodes hebdomadaire de la série télévisée Star Trek. Cet ordinateur avait un coût d'environ 400 dollars à l'époque, et le processeur venait multiplier par 10 la performance de l'antérieur, grâce à sa vitesse de 2MHz (c'était la première fois que cette mesure était utilisée), avec une mémoire de 64 Ko. En quelques mois, plusieurs milliers de ce qui est considéré comme le premier ordinateur personnel ont été vendus.

La technologie des microprocesseurs et la fabrication des circuits intégrés changent constamment et de manière très rapide. Dans l'actualité des microprocesseurs, les plus complexes contiennent aux alentours de 50 millions de transistors. Les prévisions annoncent que pour 2010 les microprocesseurs les plus avancés possèderont aux alentours de 800 millions de transistors. Mais, bien que la technologie actuelle progresse, la façon de faire sera limitée de toute façon par le propre comportement des électrons qui circulent dans les transistors. En effet, quand les dimensions sont trop réduites, les effets quantiques dus à la nature ondulatoire des électrons domineront le comportement des transistors dans le circuit. Il se peut que de nouveaux dispositifs et de nouvelles conceptions de circuits soient nécessaires au fur et à mesure que les transistors approchent des tailles atomiques.

2.2 Tendance de la technologie

Depuis 35 ans, il existe une croissance exponentielle dans la technologie des semi-conducteurs, comme l'avait prédit G. Moore dans les années 70 [MOO 65]. Moore avait constaté que le nombre de transistors qui pourrait être fabriqués sur une puce augmentait à raison de 50% par an. Or, le délai d'une porte simple diminuait à raison de 13 % par an. Ceci s'est vérifié lors des 30 dernières années, et sera probablement le cas, au moins au cours de la prochaine décade.

En observant la tendance des circuits au cours du temps, la taille de la grille des transistors est passée d'une dimension de 50µm dans les années 60 à une taille de 65nm ce qui constitue une diminution de 13% par an. La taille est ainsi divisée par deux tous les 5 ans. Il faut aussi noter que la dimension des circuits augmente elle aussi. Leur taille a doublé tous les dix ans. De plus, le délai des portes décroît de 13% par an, étant ainsi divisé par deux tous les 5 ans. Le délai d'une porte simple est donc passé d'une dizaine de nano seconde a quelques dixièmes de nano seconde.

Au fur et à mesure que la technologie réduit la taille des transistors, le délai des portes élémentaires diminue. Or, la capacité d'intégration augmente, ce qui veut dire que le nombre de

transistors dans un circuit augmente aussi. En même temps, la taille de la puce a augmenté provoquant ainsi une augmentation de la distance entre les transistors. En prenant en compte le délai qui existe dans les connexions, on observe qu'en 1998, on avait le même délai de 200ps pour une porte que pour une connexion de 5mm dans la puce. Il est estimé que pour 2008, le délai des portes passera à 50ps et celui des connexions aux alentours des 12ns. Ceci implique que le rapport de 1:1 passera à 250:1.

Le délai des connexions sur les puces deviendra par conséquent, un problème très délicat. Car ces délais affectent directement la propagation du signal d'horloge sur les systèmes numériques actuels. Il est, de nos jours, très difficile de distribuer l'horloge sur tous les éléments d'une même puce. Les portes ne sont plus un problème par rapport au délai introduit par les connexions car les portes fonctionneront plus vite que le signal d'horloge dans un circuit. Il est nécessaire de considérer que le résultat obtenu d'une porte peut aussi avoir à parcourir une distance considérable pour alimenter la porte suivante.

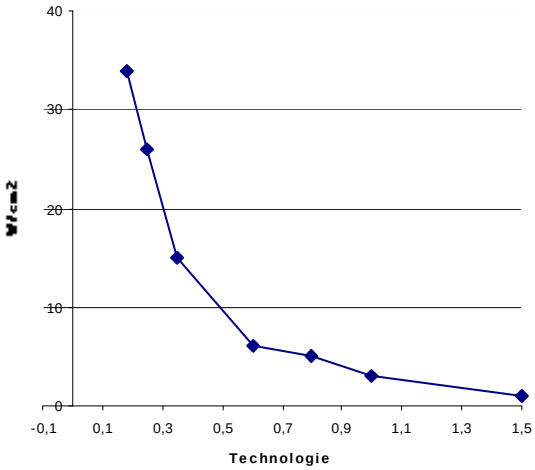


Figure 1 : Consommation par surface

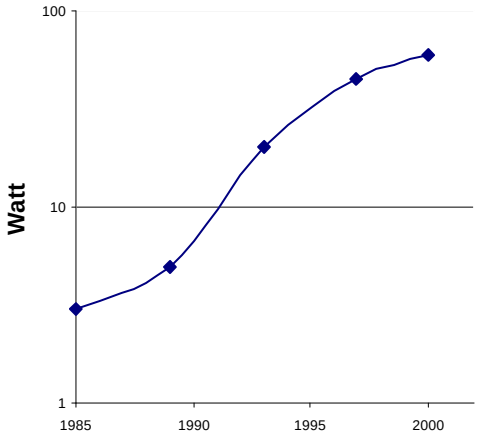


Figure 2 : Evolution de la consommation

Un autre problème qui se présente aujourd'hui est celui de la consommation des circuits actuels. En effet, l'augmentation des circuits VLSI (Very Large Scale Integration) sur des systèmes portables ou des systèmes fonctionnant avec des batteries, pose un nouveau paramètre à prendre en compte lors de la conception des circuits. L'énorme quantité de transistors intégrés sur un même circuit consomme beaucoup d'énergie. De plus, les fonctions qui ne sont pas utilisées dans les circuits synchrones (pendant un moment donné) consomment à chaque front d'horloge, ce qui augmente la consommation du circuit même s'il est inactif. La Figure 1 montre la consommation d'énergie par surface selon la technologie utilisée. La Figure 2 expose la consommation des microprocesseurs Intel du 386 au

Pentium IV. La consommation d'énergie constitue l'un des plus grands problèmes posé pour la fabrication des circuits actuels.

Ainsi, la croissance de la taille de la puce intégrant de plus en plus de transistors dans un seul circuit, augmente la consommation globale du circuit. L'autre facteur était la standardisation du voltage d'alimentation à 5V qui ne permettait pas d'alimenter les circuits avec des tensions inférieures pour réduire la consommation d'énergie. De nos jours ce voltage a été réduit a 0,85V dans les nouveaux Intel Core2 Quad Processor Q6700 @ 2.66GHz.

2.3 Microprocesseurs asynchrones

Les microprocesseurs synchrones sont les plus répandus sur le marché actuel. Toutes les entreprises qui fabriquent des microprocesseurs, sont sur la même voie. La conception de circuits synchrones a permis le développement de circuits très performants. En effet, la technologie actuelle est chaque jour plus performante en ce qui concerne la taille et la vitesse, de ce fait, les microprocesseurs actuels intègrent de plus en plus de transistors et augmentent leurs performances à chaque nouvelle version. Actuellement, la fréquence d'horloge est aux alentours de 3GHz avec une puissance qui surpasse les 50W. Or les microprocesseurs synchrones commencent à être limités par plusieurs facteurs : la très haute intégration des circuits, la vitesse de l'horloge et la consommation. L'horloge dans les circuits synchrones devient un des principaux problèmes, car de nos jours les circuits qui sont nécessaires pour la génération de l'horloge sont très compliqués et demandent une grande attention lors de la construction d'un microprocesseur. L'horloge est aussi l'une des raisons de la forte consommation d'énergie des circuits.

Plusieurs groupes de recherche ont compris que bien que les microprocesseurs synchrones soient chaque jour plus performants, ils sont très limités et dans un futur proche, il sera très difficile de pouvoir les améliorer. Pour cela d'autres solutions sont recherchées. L'asynchrone qui donne la possibilité d'éliminer l'horloge, est donc une solution très intéressante dans la conception de microprocesseurs. La conception de microprocesseurs asynchrones est présente dans la recherche depuis la fin des années 80. C'est en 1989 que le premier microprocesseur asynchrone [MAR 89a] fut conçu par l'équipe de Alain Martin à l'Institut de Technologie de Californie. Depuis ce premier microprocesseur, plusieurs ont été conçus et fabriqués. Ci-dessous, une description rapide des microprocesseurs asynchrones les plus importants qui ont été fabriqués sera faite.

2.3.1 Processeur dans la recherche

2.3.1.1 CAP

Le premier microprocesseur asynchrone est celui réalisé par l'équipe d'Alain Martin [MAR 89a]. Ce processeur a une architecture de 16 bits avec un jeu d'instructions RISC (Reduced Instruction Set Computer) simple. Il a été conçu comme un processeur simple, le jeu d'instruction n'est pas novateur et le circuit n'a pas été optimisé, il a été réalisé pour avoir un circuit fonctionnel qui avait pour but de montrer les caractéristiques de l'asynchrone. Il avait un chemin de données de 12 registres de 16 bits, 4 bus, une ALU (Arithmetic Logic Unit) et deux additionneurs. Le circuit contenait environs 20000 transistors et avait une taille de 5500λx3500λ, avec une performance de 15 MIPS (Millions Instructions Per Second) fabriqué avec une technologie de 2µm dans sa première version, puis il a été fabriqué en technologie 1,6µm. Il utilisait un protocole 4 phases avec du double rail QDI (Quasi Delay Insensitive). Le processeur contenait 2 mémoires pour les instructions et les données. Il avait trois types d'instructions, les instructions de type ALU, de type mémoire et de type PC (Program Counter).

Le microprocesseur a finalement obtenu une performance de 30MIPS (Millions Instructions Per Second) avec une consommation de 1,5W avec 1,2V, 20MIPS avec une consommation de 225mW à 5V et 5MIPS avec une consommation de 10,4mW à 2 V [MAR 89b].

2.3.1.2 Mini MIPS

Le processeur Mini MIPS a été conçu à Caltech entre les années 1995 et 1998 [MAR 97]. Ce processeur est une version asynchrone du processeur MIPSR3000. L'un des objectifs du projet était de prouver la performance des circuits asynchrones en les comparant avec leur version synchrone. Au début du projet, ce processeur a été conçu pour obtenir la vitesse d'exécution la plus rapide, car en asynchrone, il était connu que la consommation d'énergie était meilleure que pour un circuit synchrone. Ce circuit a donc été optimisé en vitesse et les concepteurs espéraient que la basse consommation allait se présenter en conséquence de l'implémentation asynchrone du processeur. Le processeur MIPS R3000 a été choisi car c'était un microprocesseur RISC classique, avec un processeur RISC 32 bits et une unité de gestion de la mémoire. Il possédait 32 registres de 32 bits, un PC et deux registres spéciaux pour la multiplication/division. Il contenait un delay-slot, trois étages de pipeline : traitement de l'adresse du PC, lecture de l'instruction et exécution de l'instruction. Or, dans les trois étages, il y avait une segmentation de pipeline très approfondie afin d'en augmenter les performances.

Ce processeur a été, dans sa première version, quatre fois plus rapide que sa version synchrone [MAR 01b]. Cette version avait une consommation d'énergie similaire à celle du R4600, et pourrait être légèrement modifié pour obtenir une consommation divisée par deux. Il a été fabriqué en technologie HP 6µm CMOS, pour une surface de 8x14 mm avec une quantité de 2M de transistors, dont 1,25M sont pour les mémoires caches. Le premier prototype visait une performance de 280MIPS à 3,3V, or il ne les a pas atteint. Les versions fabriquées avaient une performance de 150MIPS pour une consommation de 1W à 2V et 280MIPS pour une consommation de 7W à 3,3V.

2.3.1.3 AMULET

Le microprocesseur AMULET (Asynchronous Microprocessor Using Low Energy and Technology) était un projet conçu par l'université de Manchester à la fin de l'année 1990. C'était une implémentation du microprocesseur ARM et il a été développé pour mettre en évidence le potentiel de l'asynchrone pour des applications de basse consommation. Ce microprocesseur a été choisi pour démontrer la faisabilité d'un circuit complexe avec une technologie asynchrone en conservant les mêmes coûts et performances et en visant la réalisation d'un microprocesseur asynchrone commercial.

La première version de ce microprocesseur est sortie en 1993 [FUR 94], il a demandé un travail de 5 hommes/an. Il a démontré, même s'il n'a pas atteint les performances souhaitées, que la réalisation de ce type de circuit était pratique et donnait des résultats très similaires en travail de conception, en taille et en performance que pour les versions conventionnelles synchrones.

Le microprocesseur a été réalisé en utilisant les micropipelines de Sutherland [SUT 89], avec un protocole de communication de deux phases. Les résultats obtenus ne sont pas ceux qui étaient attendus, la version synchrone étant plus performante. Cette version fabriquée avec une technologie de 1µm DLM CMOS, avait une consommation de 83 mW et une performance de 9K Dhrystones [REI 84] et sa version synchrone avait une consommation de 75 mW et une performance de 14 K Dhrystones. Or, malgré le résultat, ce projet a servi à comprendre plusieurs aspects de la logique asynchrone.

La deuxième version du processeur est apparue en 1996, le AMULET 2 [FUR 98]. Cette version beaucoup plus performante que l'antérieure était fabriquée avec une technologie de 0,5µm, 3 couches de métal et comptait environ 454,000 (93,000 dans le cœur du processeur) transistors sur une puce de 6,4mm^2. Dans sa configuration la plus rapide, il avait une performance de 42Dhrystones 2,1MIPS avec une consommation de 150mW (en incluant le processeur et la mémoire cache mais en excluant les plots d'entré/sortie). Cette version était plus rapide que la version synchrone ARM710 et un peu plus lente que la version du ARM810. Cette version conservait le micropipeline de Sutherland, mais le

protocole de deux phases a été remplacé par un protocole de communication de quatre phases. Dans cette version, il y avait aussi un mécanisme de prédiction de branchement BTC (Branch Target Cache). Le BTC permet au processeur de réduire le nombre d'instructions lues en avance et réduit donc la consommation d'énergie. Une autre fonction introduite dans cette version est un mécanisme de Stop qui permet l'arrêt complet du circuit.

Une troisième version est sortie en 2000, AMULET 3 [FUR 00]. Cette nouvelle version compatible avec le code des processeurs ARM, était une version compétitive sur le marché. Et correspondait au point culminant du travail de recherche de l'université de Manchester. AMULET 3 est un microprocesseur RISC de 32 bits, fabriqué en technologie 0,35µm, il a une performance de 120MIPS avec une consommation de 155mW. Cette dernière version a prouvé que les circuits asynchrones étaient compétitifs en terme de consommation et qu'ils présentaient une interférence électromagnétique très faible. Le processeur a été intégré dans un système de contrôle appelé DRACO (Dect Radio Comunication controller)

2.3.1.4 MICA

Le micro contrôleur MICA, est un circuit QDI asynchrone de 8 bits CISC. Il a été conçu au laboratoire TIMA par le groupe CIS avec la collaboration de France Télécom R&D et ST Microelectronics en 2000 [ABR 01]. Il possède deux bancs de registres, un banc de 8 registres pour les données de 8 bits et l'autre de 8 registres de 16 bits pour les pointeurs. Il contient une unité périphérique qui peut supporter jusqu'à six ports parallèles de 8 bits, et quatre ports séries. Il contient aussi une mémoire RAM de 16Kbytes et deux mémoires ROM de 2 Kbytes.

Le processeur MICA a été testé pour une plage de tension d'alimentation de 0,6V à 3V avec une technologie de 0,25 µm. La performance du processeur pour une tension d'alimentation de 1V est de 4,3 MIPS avec une consommation de 800 µW, et pour 0,8V il consomme moins de 400 µW. Il utilise un protocole de communication de 4 phases avec un codage de n rails. Mica a été conçu en tenant compte de deux problèmes : la conception de machines à états asynchrones et la conception pour la basse consommation.

2.3.1.5 ASPRO

ASPRO [REN 98] est un microprocesseur asynchrone de 16 bits RISC à 3 étages de pipeline. La conception de ce microprocesseur a commencé à l'Ecole Nationale Supérieure de Télécommunication de Bretagne avec la collaboration de France Télécom R&D et ST Microelectronics en 1998. En 1999

l'équipe s'est associé au laboratoire TIMA. ASPRO est un microprocesseur qui décode les instructions en ordre et les termine en désordre. Il a 16 registres de 16 bits.

La conception de ce microprocesseur a été orientée dans le but d'acquérir de l'expérience dans la conception de circuits QDI asynchrones complexes utilisant des composants standard et pour prouver que les techniques de conception de circuits asynchrones pouvaient améliorer les systèmes VLSI en vitesse et en consommation d'énergie. Il utilise un protocole quatre phases avec un codage multi-rail. La synthèse a dû être partialement réalisée manuellement en utilisant des cellules standard des librairies de ST Microelectronics.

Le processeur contient 2 mémoires pour les données et pour les instructions. Il a été fabriqué avec une technologie de 0,25µm de ST Microelectronics. Et il a une performance de 24 MIPS avec une consommation de 20mW a 1V, et 140 MIPS avec une consommation de 350mW a 2,5V. Le processeur intègre trois sources d'alimentation différentes.

2.3.1.6 TITAC-2

Le processeur TITAC-2 [TAK 97] a été développé au japon au Tokyo Institute of Technologie. Ce processeur est une version asynchrone du processeur MIPS R2000. Il a été réalisé avec un model SDI (Scalable Delay Insensitive), et utilise une communication quatre phases en double rail. Il a été fabriqué avec une technologie de 0,5µm CMOS, et contient 496,367 transistors et 8,6Kbytes de mémoire cache.

La première version du TITAC-2 fonctionne correctement avec une alimentation variant de 1,5V à 6.0V, la surface ayant été chauffée à 100ºC et refroidie à –196ºC avec du nitrogène liquide. Si le processeur est alimenté avec une tension de 1,5V à 4V à une température de 20ºC, il délivre une performance de 53,3 VAX MIPS en utilisant le benchmark Dhrystones V2.1 et consomme 2,11W a 3,3V.

2.3.1.7 Lutonium

Lutonium [MAR 03] est un microcontrôleur 80c51 asynchrone conçu au California Institute of Technology. Le processeur est conçu sur un modèle QDI, visant l'efficacité d'énergie. Il est la base d'un projet de recherche qui cible les avantages de la conception asynchrone sponsorisé par DARPA. Le microprocesseur est en développement et les prévisions des performances sont avec une technologie de 0,18µm CMOS utilisant un processus TSMC SCN018 de MOSIS, de 500pJ par

instruction avec 200MIPS à une tension nominale de 1,8V. A 0,9V, ils espèrent atteindre 140pJ et 66MIPS.

La conception du microprocesseur est basée sur la théorie de Et^2. La conception utilise complètement l'habilité d'ajuster E et t avec la variation de la tension. Le Lutonium n'est pas conçu pour la basse consommation mais pour obtenir un compromis entre l'énergie consommée et le temps du cycle.

2.3.2 Processeurs commerciaux

2.3.2.1 80C51 asynchrone

Le processeur synchrone 80C51 [GAG 98] développé à l'université de Eindhoven University of technology, en collaboration avec Philips Research Laboratories, est une version asynchrone du 80C51. Ce processeur avait pour objectif de comparer les deux microprocesseurs pour montrer les avantages de la technologie asynchrone.

Ce processeur a été fabriqué avec une technologie CMOS de 0,5µm. Il possède un convertisseur DC/DC pour faire varier la tension, ce qui permet au processeur de contrôler sa consommation d'énergie. Le processeur a une performance de 4MIPS avec une consommation de 9mW à 3,3V. La version asynchrone du 80C51 divise par quatre la consommation de la version synchrone. Avec ces résultats, la technologie asynchrone a montré les bénéfices en consommation d'énergie et montra aussi que cette technologie a de très basses émissions électromagnétiques.

2.3.2.2 Data driven media processor Sharp

Le projet du Data Driven Media Processor a commencé en 1981, l'objectif du projet était de réaliser un processeur capable de répondre à certaines exigences du marché pour des applications telles que les télévisions numériques. La puissance de calcul pour ces applications n'était pas réalisable avec de simples processeurs, il a fallut se tourner vers des architectures à multiprocesseurs pour pallier le problème. La première version a été réalisée conjointement par quatre compagnies (Matsushita Electric Industrial Co., Sanyo Electric Co., Sharp Corp., et Mitsubishi Electric Co.). Les quatre ont réalisé quatre processeurs identiques et les performances d'utilisation de la plateforme qui utilisait les processeurs en mode multi processeurs ont été démontrées avec un logiciel distribué dans les quatre processeurs.

Avec cette première version, Sharp Corp. et Mitsubishi Electric on procédé a l'élaboration d'un prototype contenant cinq processeurs [KOM 89]. En 1991, Sharp Co. a réalisé la première version qui contenait tous les processeurs sur la même puce [TER 95]. Ce processeur en soi n'était pas un processeur asynchrone, mais la politique de pipeline du processeur était entièrement de type asynchrone. Ce pipeline a été choisi pour les avantages qu'il présentait en termes de consommation d'énergie et pour réduire les problèmes dus au « clock skew » dans les circuits synchrones. La première version du processeur était capable d'effectuer une dizaine de milliards de calcul de traitement de signaux par seconde, avec une consommation de 2W. Ceci a permis une réduction de la consommation de trois voire dix fois moins que les processeurs séquentiels conventionnels (Digital Signal Processor - DSP) existants sur le marché.

2.3.2.3 Flexible asynchronous microprocessor Epson 2005

La compagnie Seiko Epson a développé le premier microprocesseur asynchrone sur un substrat flexible [KAR 05] en utilisant des transistors couche mince avec du poly silicium basse température. Ce type de technologie est très attrayante puisqu'elle est de basse consommation d'énergie et permet des émissions électromagnétiques beaucoup moins élevées. Malheureusement, il existe un inconvénient à ce microprocesseur : le mauvais contrôle des caractéristiques lors de la production des circuits ; ceci étant dû aux déviations des caractéristiques générées principalement par la taille des grains des cristaux et la largeur du silicium-oxyde. Les déviations que présentait cette technologie allaient au delà des capacités des circuits synchrones.

Les avantages des circuits asynchrones en ce qui concerne la tolérance aux déviations dues aux tailles des transistors qui constituent le circuit, ont permis de concevoir ce premier microprocesseur avec une telle technologie. Ce processeur a été conçu avec une technologie LTPS (Low Temperature Poly-Silicon) TFT (Thin Film Transistor) CMOS 4µm/12µm avec deux niveaux de métal, il compte environs 32000 transistors et une mémoire de 16Moctets. C'est un processeur 8 bits avec un jeu d'instructions de 608 instructions qui inclue des instructions de multiplication et division. Le bus de données est lui synchrone et possède une interface pour relier le bus de sortie synchrone avec le cœur du processeur asynchrone.

2.4 Conclusions

Bien que la quantité de circuits asynchrones conçus jusqu'à nos jours soit assez réduite en comparaison avec les microprocesseurs synchrones, la recherche dans les microprocesseurs asynchrones est en voie de développement. L'industrie continue à fabriquer en grande quantité les

versions synchrones et l’industrie commence à peine à s’intéresser à la production de processeurs asynchrones.

Or les recherches sur la technologie asynchrone sont chaque jour plus performantes et il est possible de voir que la performance de ces circuits est supérieure à celle des circuits synchrones. La consommation des circuits asynchrones est inférieure et leur vitesse est supérieure. Cependant, le travail de recherche à effectuer avant que les circuits asynchrones ne puissent remplacer la technologie actuelle qui a déjà plusieurs années d’avance est encore énorme.

Chapitre 2

MÉTHODOLOGIE POUR LA CONCEPTION DE CIRCUITS ASYNCHRONES QDI BASSE CONSOMMATION

1 Introduction

Le chapitre 1 de ce travail a exposé la problématique du manque d'outils et de méthodologies en ce qui concerne la conception des circuits asynchrones. Ceci est un des facteurs les plus limitant qui freinent l'utilisation de ce type de circuits sur le marché actuel. Le travail proposé présente une méthodologie qui permet la génération de circuits asynchrones QDI faible consommation. Une méthode pour l'estimation de l'énergie consommée par le circuit est aussi présentée. La méthodologie a pour but de donner une estimation de l'énergie consommée par le circuit très tôt dans le flot de conception du circuit. Ceci permet au concepteur de réaliser les optimisations nécessaires avant la fin du flot, et de gagner du temps lors de la conception du circuit.

Trois outils d'automatisation pour la synthèse, l'estimation d'énergie dynamique et pour la projection technologique sont utilisés dans le flot de conception. Ces outils ont été réalisés au sein du laboratoire TIMA par le groupe de recherche CIS. Le flot de conception utilise aussi une librairie spécifique asynchrone TAL (Tima Asynchronous Library) développée aussi par le groupe CIS.

La méthodologie cible les circuits asynchrones QDI. Il existe en effet plusieurs types de circuits asynchrones, chaque type de circuit étant basé sur des hypothèses temporelles différentes.

Dans un premier temps, ce travail présente la différence entre plusieurs types de circuits asynchrones. Il montrera, par la suite, les avantages des circuits QDI par rapport aux autres circuits.

2 Présentation des différents types de circuits asynchrones

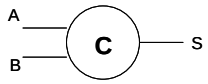
2.1 Circuits indépendants de la vitesse (DI)

Les circuits asynchrones les plus robustes sont les circuits DI (Delay Insensitve : Insensible aux délais) [UDD 86]. Ce type de circuit est celui qui ne fait aucune hypothèse temporelle. Ils ne prennent aucune hypothèse temporelle en compte pour leur fonctionnement ce qui veut dire que le circuit fonctionne indépendamment des délais dans les cellules qui le composent ainsi que des délais dus à la communication dans les fils entre cellules. Ce type de circuit utilise la notion de causalité entre les événements liés aux transitions. La notion de causalité implique que chaque transition à l'entrée d'une cellule cause une transition en sortie. En pratique, toutes les transitions en entrée contribuent au calcul de la sortie, ce qui assure le fonctionnement du circuit. En effet le fonctionnement du circuit est assuré car c'est la transition à la sortie qui acquitte la transition en entrée. Le modèle utilisé pour le délai des fils et des cellules qui composent le circuit est un modèle "non-borné".

Pour être capable de répondre à ces hypothèses temporelles, les cellules qui composent le circuit doivent attendre toutes les entrées pour fournir une sortie. Avec un modèle de délais "non-borné", l'utilisation exclusives de portes telles que les portes "ET", "OU", etc… n'est donc pas possible. En effet, ce type de portes fournit une sortie dès que l'une de ses entrées a changé. Il est impossible pour les cellules d'assurer la réception de toutes les données d'entrée pour générer la sortie. Le signal d'acquittement est généré sans avoir reçu toutes les entrées. Actuellement, il n'existe qu'une seule porte de base qui respecte ce modèle de délai. Cette porte est appelée "porte de Muller" [ref 1p2007] (Figure 2-1). La porte de Muller assure que la sortie ne change que lorsque toutes les entrées ont été reçues. Il est donc évident que la quantité d'applications qui peuvent être réalisées est très limitée [MAR 90]. La seule façon d'élargir le champ d'application étant d'utiliser des portes complexes. La réalisation de tels circuits est alors basée sur des portes standard qui sont plus complexes que celles utilisées normalement [EBE 91], [HAU 95], elles doivent en particulier avoir plusieurs sorties.

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

Table de vérité



Symbole

Figure 2-1 : Porte de Muller

2.2 Circuits Quasi Insensibles aux Délais (QDI)

Les circuits insensibles aux délais sont très contraignants. Relâcher les contraintes sur les délais est une solution pour élargir le champ d'utilisation des circuits asynchrones. Les circuits asynchrones Quasi Insensibles au Délais (QDI) introduisent pour ce faire la notion de fourche isochrone. Cette notion est la plus petite hypothèse temporelle nécessaire pour permettre la réalisation de tout circuit.

La notion de fourche isochrone est définie de la manière suivante : une fourche isochrone est une fourche dans laquelle l'information qui arrive à la fourche sera transmise avec un temps égal à tous les récepteurs connectés à chaque branche de la fourche. Une fourche étant une séparation en plusieurs branches d'un fil de connexion pour connecter la sortie d'une porte à plusieurs portes en amont. La Figure 2-2 montre le principe d'une fourche isochrone. Dans cet exemple, pour que la fourche soit isochrone, les délais Δ_1, Δ_2 et Δ_n doivent êtres égaux. Cette simple hypothèse sur le modèle des circuits QDI permet d'acquitter une seule branche de la fourche puisque l'hypothèse sur les délais des branches implique que l'information envoyée par l'émetteur arrive en même temps sur tous les récepteurs. L'introduction de cette hypothèse rend possible la conception de circuits asynchrones avec des cellules de base standard telles que les portes "ET", "OU", "OU exclusif" ou toute autre cellule à une seule sortie ce qui n'était pas possible dans le modèle insensible aux délais (DI) présenté dans la section précédente. Dans [MAR 93], A. Martin montre que l'hypothèse temporelle de fourche isochrone est l'hypothèse la plus faible à ajouter pour concevoir des circuits avec des portes à plusieurs entrées et une seule sortie.

Un autre avantage des circuits QDI est qu'ils ne nécessitent pas d'analyse de temps pour vérifier leur fonctionnement. Cet avantage rend les circuits QDI très modulaires. En effet, il est possible d'imbriquer des modules sans avoir à vérifier les délais des connexions. La seule condition qui doit être assurée entre blocs est qu'ils doivent utiliser le même protocole de communication. Ce protocole est un protocole de poignée de mains qui est utilisé pour la conception de circuits asynchrones. Le principe du protocole est d'assurer la validité des données entre l'émetteur et le récepteur ainsi que de générer un acquittement pour informer le récepteur de la réception des données.

Plusieurs protocoles sont utilisés de nos jours comme le protocole deux phases ou le protocole quatre phases.

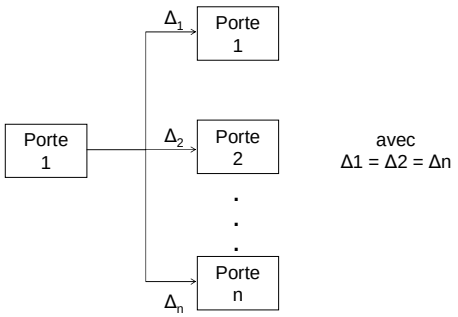


Figure 2-2 : Hypothèse temporelle de la fourche isochrone

2.3 Circuits indépendants de la vitesse (SI)

Cette catégorie de circuits asynchrones suppose que les délais dans les fils du circuit sont négligeables mais elle conserve le modèle “non-borné” pour les délais dans les portes. En pratique, elle considère toutes les fourches comme étant isochrones. Les circuits indépendants de la vitesse font plus d’hypothèses temporelles que les circuits QDI. La différence entres les circuits QDI et les circuits DI est subtile. En effet pendant plusieurs années les différences entre les deux modèles ont fait l’objet de recherches. De nos jours, il existe une équivalence entre les modèles QDI et SI, cette équivalence est montrée dans [HAU 95]. Ce travail montre comment il est possible de représenter une fourche isochrone dans un circuit SI (Figure 2-3).

Les deux modèles de circuits sont donc très similaires. La seule différence étant que les circuits indépendants de la vitesse n’identifient pas les fourches isochrones. Ce modèle considère toutes les fourches comme sensibles aux délais. Par rapport aux circuits QDI, ce modèle contient moins d’informations qui sont potentiellement nécessaires aux outils de conception.

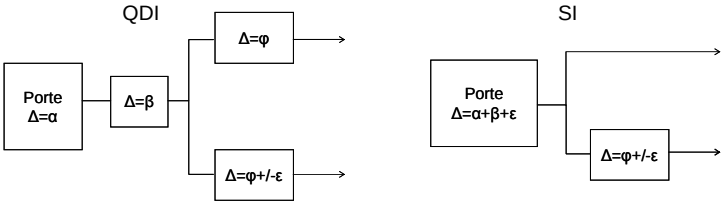


Figure 2-3 : Modèle équivalent aux circuits QDI et SI

2.4 Circuits Micropipeline

Les circuits micropipeline ont été introduits par Ivan Sutherland [SUT 89]. Cette catégorie de circuit fonctionne de manière similaire aux circuits synchrones. Ils gardent la même structure que les circuits synchrones qui est généralement composée de blocs combinatoires interconnectés par des registres. La Figure 2-4 et la Figure 2-5 montrent respectivement le principe de fonctionnement des circuits synchrones et celui des circuits micropipeline. L'horloge est donc remplacée par des circuits qui calculent et génèrent les signaux de requête et d'acquittement.

Dans un circuit synchrone, le signal d'horloge indique aux registres de prendre les données présentes à l'entrée pour les faire passer au bloc combinatoire suivant. Ces données reste dans le registre jusqu'au prochain front d'horloge. L'idée du micropipeline est de remplacer ce signal d'horloge en générant localement un front d'horloge sur chaque registre indépendamment. Le signal généré implémente le protocole de poignée de mains.

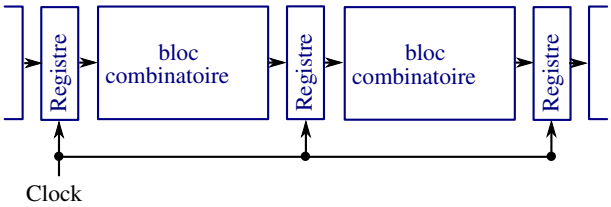


Figure 2-4 : circuit synchrone

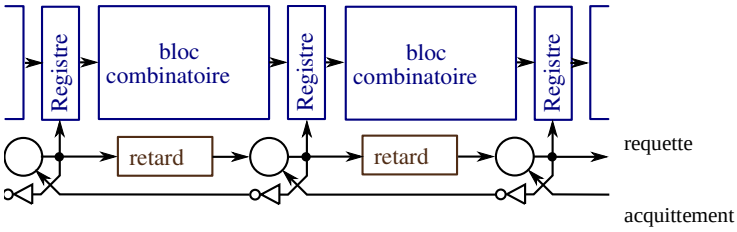


Figure 2-5 : circuit micropipeline

L'avantage des circuits micropipeline est que le chemin de donnée est entièrement concevable avec une structure synchrone. De ce fait, il est possible d'utiliser les outils disponibles sur le marché pour concevoir le circuit. Cette catégorie de circuits bénéficie de certains avantages des circuits asynchrones puisque l'arbre d'horloge est entièrement supprimé ce qui élimine tous les problèmes qu'il impose dans un circuit synchrone (moindres contraintes, moins de bruits parasites à une fréquence fixe). Néanmoins, le protocole poignée de mains est généré avec un circuit qui contient des cellules de retard ce qui implique une contrainte de vitesse pour le circuit. En effet, le retard doit être conçu pour assurer la validité des données à l'entrée du registre suivant. Le circuit doit alors

fonctionner au pire cas. Il faut alors une étape d'analyse des délais pour vérifier leur bon fonctionnement.

2.5 Circuits de Huffman

Ces circuits sont spécifiés sous la forme d'un automate (ou machine à états) fini asynchrone. Il n'y a pas d'horloge globale qui cadence l'évolution du circuit mais on introduit une borne sur les délais du calcul de l'état suivant ainsi que sur les entrées et sorties du circuit. La Figure 2-6 représente la structure d'un circuit de Huffman.

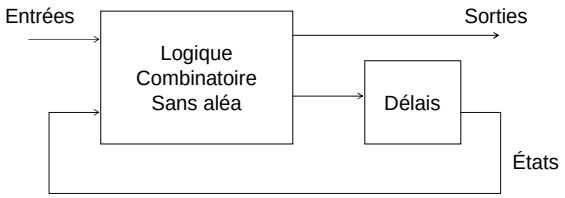


Figure 2-6 : circuit de Hauffman

La synthèse de ces circuits est décomposée en trois étapes, comme pour les circuits synchrones, sauf que ces circuits doivent être sans alea :

- minimisation de l'espace d'états
- codage binaire des états
- optimisation logique

3 Présentation du flot de conception

La conception de circuits asynchrones est très limitée de nos jours. Bien qu'il existe quelques méthodologies et techniques de synthèse [PEE 04, BAR 07, EDW 02 et BRE 07], l'automatisation de la conception de ces circuits est encore en développement. Ce travail a pour but de proposer une méthodologie automatique pour la conception de circuits asynchrones QDI. Il présente une technique d'estimation de l'énergie consommée par le circuit après avoir été synthétisé avec un outil de synthèse qui a été conçu au sein du laboratoire TIMA dans le groupe CIS. Cette méthodologie présente l'avantage d'être entièrement automatique et permet d'identifier les parties qui consomment le plus en terme d'énergie très tôt dans le flot de conception. Le flot inclut une étape de projection technologique qui utilise une librairie spécifique asynchrone développée au laboratoire TIMA au sein du groupe CIS [FOL 07]. Dans un premier temps, ce travail présentera le flot de conception. Il décrira par la suite la

méthode d'estimation de l'énergie. Finalement, il montrera les résultats obtenus pour l'estimation d'énergie d'un crypto processeur DES asynchrone.

3.1 Flot de conception

La méthodologie de conception proposée dans ce travail de thèse est représentée en détail par le flot montré dans la Figure 2-7. Il montre les étapes qui permettent de concevoir des circuits asynchrones QDI basse consommation à partir de la spécification avec le langage CHP (Communicating Hardware Processes) jusqu'au placement routage.

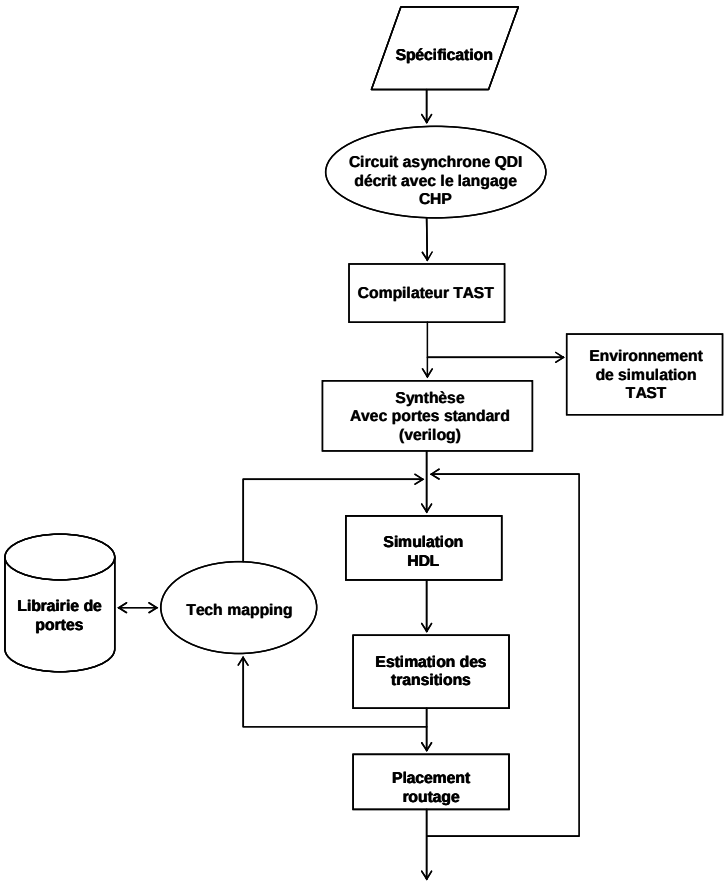


Figure 2-7 : Flot de conception

La conception du circuit asynchrone commence par la spécification du circuit en soit. Cette spécification détermine les choix d'architectures qui composeront le circuit. Le circuit est décomposé en blocs fonctionnels qui seront ensuite décrits avec le langage CHP [MAR 89c]. Le langage CHP étant un langage de haut niveau spécifique pour les circuits asynchrones. Le circuit décrit en CHP est compilé avec l'outil TAST (Tool for Asynchronous circuits SynThesis). TAST est un outil servant à la simulation et à la synthèse des circuits asynchrones qui prend en entrée un circuit décrit avec le

langage CHP. La simulation du circuit permet de vérifier si le circuit est bien fonctionnel. Il vérifie s'il n'y a pas de points bloquants lors du fonctionnement du circuit.

L'outil TAST réalise ensuite une synthèse sur des portes standard à deux entrées. La porte de Muller est rajoutée aux cellules de base. Cette synthèse est une synthèse non optimisée et sera utilisée pour effectuer une première simulation qui valide le circuit. Un vecteur d'entrée est ainsi utilisé lors de la simulation HDL du circuit. La simulation HDL est réalisée avec l'outil Modelsim. Il est donc possible de créer un benchmark en VHDL pour stimuler les différentes parties du circuit.

Lors de la simulation avec Modelsim, un outil pour estimer l'énergie consommée par le circuit peut être utilisée. Cet outil est un plug-in qui s'attache à Modelsim pour effectuer trois opérations. Tout d'abord, il est capable de compter les transitions à la sortie de chaque porte. Compter le nombre de transitions réalisées par le circuit permet d'avoir un profil d'activité du circuit pour repérer les parties les plus actives dans le circuit. L'outil d'estimation d'énergie est aussi capable de pondérer les transitions réalisées par les portes avec leur capacité en sortie. En effet, toutes les transitions des cellules qui composent le circuit, ne représentent pas la même énergie consommée. Ceci est due au fait que les cellules non pas la même charge en sortie. Lorsqu'une transition à la sortie de la porte se présente, cette dernière doit fournir le courant nécessaire pour charger la capacité qui dépendra alors de cette capacité de charge. Pondérer les transitions avec la capacité associée à chaque cellule permet d'avoir une estimation de l'énergie consommée par le circuit. Finalement l'outil rend aussi un rapport quant à la taille du circuit.

L'obtention des capacités de charge à la sortie de chaque porte peut être obtenue après l'étape de projection technologique. Avec cette étape le circuit est optimisé en utilisant la librairie TAL. TAL est une librairie asynchrone composée de portes complexes asynchrones. Durant cette étape, les portes sont sélectionnées pour qu'elles soient capables d'alimenter la charge en sortie. La vérification du circuit est réalisée à nouveau avec une simulation HDL.

Pour obtenir une estimation plus réaliste de la consommation dans le circuit, il est nécessaire de faire le placement routage du circuit. Cette étape fournit les capacités des fils d'interconnexions. La capacité en sortie des portes est ainsi calculée et les transitions des sorties sont alors pondérées. L'outil délivre en sortie une estimation de la capacité commutée pour le vecteur utilisé en entrée.

Ce flot permet la conception de circuits asynchrones. Il a l'avantage d'être entièrement automatique ce qui accélère la génération des circuits asynchrones QDI. Une estimation de l'énergie consommée est obtenue très tôt dans le flot de conception ce qui permet de comparer différentes architectures pour obtenir les performances en consommation d'énergie ou en vitesse souhaitées.

3.2 Description du circuit avec le langage CHP

Les langages de description haut niveau des circuits actuels ne permettent pas directement la conception de circuits asynchrones. Il est très difficile de les implémenter puisque les langages haut niveau tels que VHDL ou Verilog sont conçus pour décrire des circuits synchrones où tout est régi par un signal qui cadence le passage des données entre blocs. Les circuits asynchrones comme les circuits synchrones sont constitués de blocs fonctionnels qui communiquent entre eux. Mais dans les circuits asynchrones, les communications entre blocs s'effectuent avec un protocole poignée de main qui est généré entre chaque bloc. La communication ne se réalise pas à l'aide d'un signal d'horloge global. Ce type de circuit intègre beaucoup de parallélisme, chaque bloc communique entre eux par l'intermédiaire de canaux de communication qui eux-mêmes doivent assurer la génération du protocole utilisé. La description haut niveau de ce type de circuits doit alors prévoir les communications des blocs qui ont souvent un degré de parallélisme élevé.

Alain Martin dans [MAR 89c] propose le langage CHP spécifique à l'asynchrone inspiré du langage CSP [HOA 78] avec l'utilisation des commandes gardées montrées dans [WYB 97]. Ce langage permet de décrire des blocs fonctionnels communiquants entre eux. Ce langage a été repris au laboratoire TIMA dans le groupe CIS, plusieurs extensions et quelques limitations y ont été ajoutées pour répondre de manière plus adaptée à la spécification de circuits asynchrones. Ce langage est bien un langage de haut niveau qui décrit les circuits asynchrones mais reste encore très basique par rapport au langage haut niveau existant comme VHDL, Verilog et C. En effet le CHP ne propose qu'un seul type de donnée qui est le type booléen.

Le langage CHP permet cependant de décrire un circuit asynchrone d'une manière assez simple. Le circuit est décrit par blocs qui communiquent entre eux, et qui réalisent des opérations arithmétiques, logiques, de comparaison, de communication et de conversion. Les données sont représentées par des canaux multi-rails de type booléen. Les canaux peuvent être définis avec une taille allant du mono rail à un canal en n rails. Le nombre de rails correspond au nombre de valeurs que peut prendre la donnée en soi. Si la donnée est une donnée binaire en base deux, le canal pour chaque bit du signal sera un canal 1 parmi 2. Avec ce même principe, il est possible de créer des canaux 1 parmi n. Le langage CHP permet alors la création de données en quatre rails qui correspond à un canal qui transporte une donnée 1 parmi 4, qui est un des codages les plus avantageux pour les communications asynchrones puisque c'est ce type de codage de données qui fournit le meilleur rapport consommation d'énergie / taille du canal.

En effet, le codage en quatre rails nécessite quatre fils pour construire un digit de deux bits. A chaque communication, un seul des fils commute. Avec un codage en double rails (binaire) la

construction d'un digit deux bits est réalisée avec quatre fils aussi. Or lors d'une communication, il y a deux fils qui commutent. Construire des digits avec plus de rails n'optimise pas les communications en terme de consommation d'énergie et de taille du canal. Le Tableau 2-1**Erreur ! Source du renvoi introuvable.** montre les commutations nécessaires pour différents codages et la largeur nécessaire en nombre de fils par digit (noté dans le tableau [nombre de transitions|nombre de fils]). Avec un codage n rails, le nombre de transitions est le strict minimum. Cependant, le canal explose en taille puisque la taille de celui-ci est une puissance de n contrairement aux codage quatre rails et double rails qui eux sont linéaires.

	1 bit	2 bits	3 bits	4 bits	16 bits	N bits
Dual rails	1 | 2	2 | 4	3 | 6	4 | 8	16 | 32	n | 2n
Quatre rails	1 | 2	1 | 4	2 | 6	2 | 8	8 | 32	$\left[\frac{n+1}{2}\right] \mid 2n$
16 rails	1 | 2	1 | 4	1 | 8	1 | 16	4 | 64	
N rail	1 | 2	1 | 4	1 | 8	1 | 16	1 | 65536	$1 \mid 2^n$

Tableau 2-1 : (transitions|fils) pour différentes tailles du canal.

3.3 L'outil de conception TAST

Depuis quelques années, le groupe de travail CIS du laboratoire TIMA développe un environnement de conception pour les circuits asynchrones QDI [DIN 02a], [DIN 2b], [DIN 03], [FOL 07] et [BRE 07]. Cet outil est une contribution au manque d'outils de conception de circuits asynchrones dans le but d'essayer de faire évoluer la technologie des circuits asynchrones. L'objectif de cet outil est de générer des circuits asynchrones QDI décrits avec le langage de haut niveau CHP présentés dans le paragraphe précédent. TAST est constitué de plusieurs outils qui réalisent différentes étapes de la conception des circuits asynchrones QDI. La Figure 2-8 montre le flot de conception de TAST. TAST utilise un format intermédiaire basé sur les réseaux de Pétri. La partie front-end de l'outil transforme le circuit décrit en CHP au format intermédiaire en réseau de Pétri. Ce format permet la simulation et la vérification du circuit décrit en CHP.

La simulation du circuit peut être réalisée à l'aide de l'interface graphique qui permet de visualiser le réseau de Pétri généré par le compilateur TAST. La simulation est réalisée en faisant avancer des jetons dans le réseau Pétri. Le déplacement des jetons qui sont susceptibles d'avancer à la place suivante est choisi aléatoirement. Ceci permet une simulation correcte du déplacement des données dans un circuit asynchrone. Cette simulation peut aussi être réalisée par ligne de commande,

ce qui permet l'utilisation de scripts avec des fichiers de vecteurs d'entrée. Le simulateur donne en sortie un fichier avec les résultats des sorties spécifiés par le concepteur.

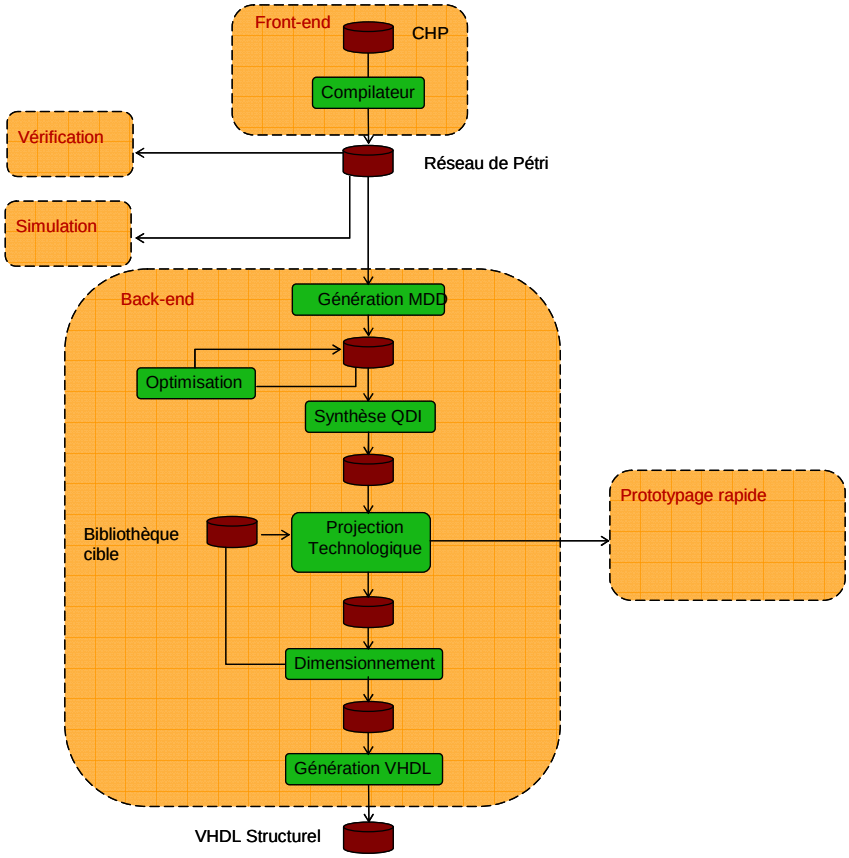


Figure 2-8 : Flot de conception TAST

3.4 Synthèse avec des portes a deux entrées

Dans un premier temps, l'outil TAST réalise une synthèse qui n'est pas liée à une technologie quelconque. La synthèse s'effectue en utilisant des cellules standard à deux entrées avec des portes "et" et "ou", ainsi qu'avec la cellule de base asynchrone, la porte de Muller. Cette synthèse permet de vérifier le fonctionnement du circuit en général. TAST délivre un fichier de description en portes logiques (netlist) en format Verilog ou VHDL avec des instances de portes standard. Cette netlist permet de simuler le circuit avec des logiciels commerciaux pour vérifier le fonctionnement du circuit. Il est alors possible d'utiliser des logiciels comme Modelsim pour simuler le circuit de façon comportementale à l'aide d'un "test bench" décrit en VHDL ou Verilog. Par la suite, ce travail montrera qu'il est possible de faire une première estimation de l'activité du circuit au niveau des portes pour optimiser les blocs les plus actifs et ainsi réduire la consommation et optimiser la vitesse des chemins de données dans le circuit. Cette étape est très intéressante puisqu'elle se présente très tôt

dans le flot de conception du circuit. Le but d'obtenir une netlist avec des portes génériques étant aussi de pouvoir choisir par la suite la bibliothèque et la technologie souhaitée pour le circuit en question.

3.5 Projection sur la librairie TAL

Après avoir réalisé la synthèse du circuit sur des portes standard à deux entrées, le circuit peut être optimisé en utilisant des cellules complexes pour en réduire la taille et la consommation. Cette étape est connue comme la projection technologique (library binding en anglais). La projection technologique consiste à transformer un réseau logique indépendant en un réseau logique dépendant d'une technologie, ce qui revient à transformer un circuit ciblant des portes génériques en un circuit composé de portes standard et complexes issues d'une bibliothèque spécifique. Pour effectuer cette étape, il est nécessaire d'utiliser une bibliothèque de cellules données, ceci permet d'utiliser un même circuit avec plusieurs technologies. Chaque bibliothèque contient un ensemble de primitives logiques qui comportent des cellules combinatoires et séquentielles. Pour chaque cellule de la librairie, les caractéristiques en terme de taille, de délai et de charge capacitive sont données et permettent d'optimiser le circuit en vitesse, en consommation ou en taille.

La seule contrainte de la projection technologique est qu'elle doit assurer que le circuit obtenu soit aussi QDI. Une étude approfondie de l'étape de projection technologique est présentée par le travail de thèse de Bertrand Folco [FOL 07].

3.5.1 Présentation de l'outil de projection

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

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

Réaliser la projection technologique permet de regrouper plusieurs cellules génériques ce qui réduit la taille ainsi que la consommation du circuit. Pour pouvoir réaliser cette étape, la méthode consiste à réaliser une couverture du réseau original par des sous réseaux qui proviennent des éléments

d'une bibliothèque. Les travaux montrés dans [FOL 07] présentent la méthodologie utilisée pour réaliser cette étape avec l'outil utilisé dans ce travail.

En pratique, la fusion de deux portes peut se faire si et seulement si la sortie de la première porte est connectée uniquement à l'une des entrées de la deuxième. Si la sortie de la première porte alimente plus d'une entrée d'un nombre n de portes en aval, la fusion n'est pas possible puisque le nœud entre la sortie et l'entrée de la porte qui suit va se perdre. La Figure 2-9 montre un cas où il est possible de réaliser cette fusion et un cas où celle-ci n'est pas possible. Dans cet exemple, les cellules du circuit montré dans la Figure 2-9.a peuvent fusionner en une cellule complexe montrée dans la Figure 2-9.b. Les deux portes "ou" O1 et O2 et la porte "et" and1 montrée dans la Figure 2-9.c ne peuvent pas être fusionnés puisque le nœud X_2 serait perdu lors de la même fusion montrée antérieurement. Pour fusionner les cellules de ce circuit, le nœud X_2 doit rester présent dans le circuit. En fonction des cellules disponibles dans la librairie cible, il est possible de réaliser d'autres fusions comme celle montrée dans la Figure 2-9.d.

La fusion de portes ne peut se faire que sur des portes combinatoires. Si un nœud du circuit est mémorisant, la fusion n'est pas possible à cause des règles de production résultante après la fusion. Cependant, il est possible de concevoir des cellules complexes à partir de portes de Muller, si la porte complexe résultante de la fusion de portes de Muller avec des portes standard conserve internement les nœuds mémorisants. La Figure 2-10 présente deux exemples de fusion de cellules de Muller avec des portes "ou".

Avec la synthèse en portes standard à deux entrées effectuée par l'outil TAST, il est donc possible de réaliser une projection technologique avec une librairie de portes complexes données. Un outil de projection technologique a été conçu au sein du groupe CIS du laboratoire TIMA. Cet outil prend la netlist en verilog du circuit synthétisé et délivre une nouvelle netlist avec les portes de la librairie choisie.

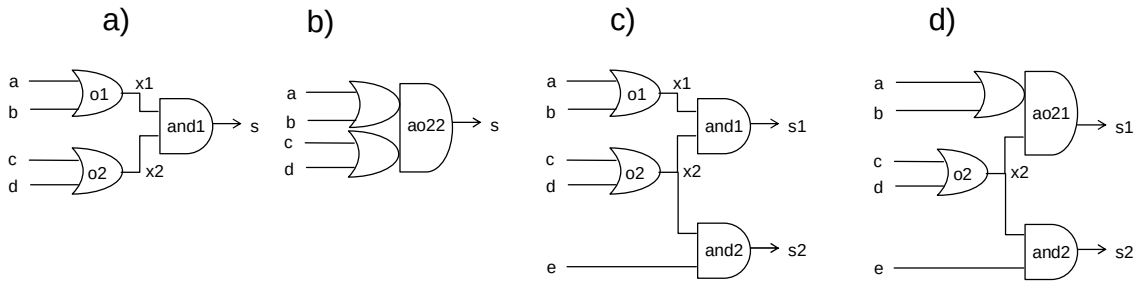


Figure 2-9 : Exemple de fusion de cellules standard en cellules complexes

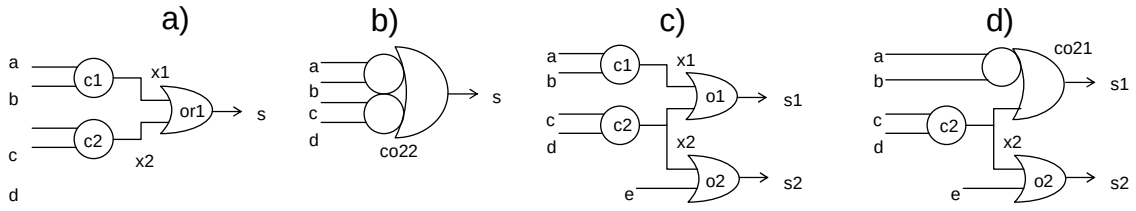


Figure 2-10 : Exemple de fusion de cellules asynchrones standard en cellules complexes

3.5.2 Présentation de la librairie TAL

Actuellement, les bibliothèques de cellules du marché ne contiennent que des portes utilisées pour les circuits synchrones. Les portes les plus courantes regroupe des portes “ou” avec des portes “et” ainsi que d’autres fonction plus complexes. Pour la synthèse de circuits asynchrones, la porte de Muller vient introduire de nouvelles portes complexes spécifiques aux circuits asynchrones. Ces portes complexes regroupent les portes de Muller et les portes standard “ou” et “et”.

Une bibliothèque spécifique asynchrone était nécessaire pour la conception automatique de ce type de circuit. La bibliothèque TAL, acronyme de “Tima Asynchronous Library” conçue au laboratoire TIMA contient un ensemble de portes complexes pour pallier au manque de librairies asynchrones. Les cellules de base étant constitué de portes de Muller et de portes “ou”. La librairie propose aussi certaines portes dissymétriques qui altèrent le fonctionnent logique des portes de Muller [RIG 02]. Il existe deux types de portes de muller dissymétrique : les portes dissymétriques négatives et le portes de muller positives. Ces portes permettent de faire basculer la sortie de la porte à un niveau logique bas ou à un niveau logique haut avec une seule des entrées, contrairement à la porte de muller standard qui nécessite l’arrivée des deux entrées pour faire basculer la sortie. Actuellement, il existe deux versions de cette librairie. La première est faite avec une technologie de 130nm de ST Microelectronics et la deuxième plus récente (TAL2) est conçue avec une technologie de 65nm de ST Microelectronics. Cette nouvelle version reprend les mêmes cellules que la première, mais a été conçue avec des objectifs légèrement différents. Tout d’abord, la bibliothèque TAL2 corrige le fonctionnement à de faibles tensions d’alimentation. En effet, la première version induisait quelques problèmes lorsque le circuit est alimenté à des tensions inférieures à 0.8V. Le deuxième objectif de la librairie TAL2 était d’introduire la possibilité d’utiliser le protocole de communication de type PCHB. Et globalement, TAL2 est de nos jours une librairie qui commence à être en phase de maturité. La bibliothèque compte 165 cellules qui permettent de réduire considérablement la taille et la consommation des circuits asynchrones synthétisés avec des portes standard à deux entrées.

La surface est réduite en moyenne d'un facteur 2 par rapport à une conception utilisant des cellules AO222. Le Tableau 2-2 et le Tableau 2-3 montrent le facteur de réduction en surface entre les cellules de la bibliothèque TAL et la bibliothèque TAL2 et les implémentations correspondantes à partir de portes standard AO222. Dans ce tableau, il est possible d'observer que plus la porte est grande en nombre d'entrées, plus le gain en surface est élevé. La nouvelle version de la bibliothèque est légèrement plus grande que la première, ceci est dû au fait que TAL2 est conçu avec des cellules statiques contrairement aux cellules dynamiques qui composent la première version. Ceci augmente le nombre de transistors dans chaque porte mais permet de réduire la tension d'alimentation jusqu'à 0.5V. La conception statique de la bibliothèque apporte aussi un gain en vitesse par rapport à la version dynamique.

Portes	Facteur de réduction en surface en fonction de la sortance		
	X1	X2	X4
M2	1.1	1.4	1.5
M3	2	2	2.1
M4	3.3	3.3	2.6
M2RB	1.2	1.2	1.4
MO22	1.9	1.3	1.3

Tableau 2-2 : Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222 pour la bibliothèque TAL

Portes	Facteur de réduction en surface en fonction de la sortance		
	X4	X18	X35
M2	1.1	0.8	1
M3	1.2	1	1.4
M4	1.6	1.1	1.6
M2RB	1	0.8	1.1

Tableau 2-3 : Facteurs de réduction de surface obtenus par rapport à des implantations à base de AO222 pour la bibliothèque TAL2

En terme de consommation, la bibliothèque TAL permet un gain moyen de 13% par rapport à la même cellule conçue avec des portes AO222. Rapporté à l'ensemble de la bibliothèque, le gain en énergie est de 11%. Le Tableau 2-4 et Tableau 2-5 montrent les différents gains en consommation des portes les plus utilisées dans les circuits asynchrones par rapport aux portes conçues avec des AO222 par rapport au cellules des librairies TAL et TAL2. Il montre aussi la consommation pour plusieurs sortances des cellules. Les gains sont donnés pour des transitions descendantes ainsi que pour des transitions montantes. Le gain est calculé avec le rapport entre la consommation de la porte TAL et la

consommation de la structure avec des portes AO222. Dans le Tableau 2-4, il est possible de voir que la consommation en énergie est supérieure par rapport aux structures avec des AO222 pour les transitions en montée dans les portes de Muller à deux entrées. En effet, pour les portes de Muller à 2 entrées, le driver de sortie est une structure à rapport. Or ce type de structure va dissiper une importante quantité de puissance lors du changement de valeur de toutes les entrées de 0 à 1 (ou de 1 à 0) en provoquant un conflit au niveau du noeud interne située entre la partie dynamique et la mémorisation (noeud X dans la Figure 2-11). Néanmoins, La consommation pour ces portes est en moyenne inférieure à la consommation avec des portes AO222 si on considère le gain en consommation de la montée et de la descente.

Portes	Gain en énergie					
	Montée			Descente		
	X1	X2	X3	X1	X2	X3
M2	1.42	1.36	1.04	0.56	0.57	0.6
M3	0.58	0.58	0.5	0.55	0.55	0.56
M2RB	1.32	1.2	1.16	0.63	0.62	0.69
MO22	1.22	1.32	1	0.85	0.87	0.84

Tableau 2-4 : Gain en énergie des cellules TAL

Portes	Gain en énergie					
	Montée			Descente		
	X4	X18	X35	X4	X18	X35
M2	0.87	0.81	0.77	0.84	0.79	0.69
M3	0.73	0.69	0.53	0.64	0.58	0.43
M2RB	0.61	0.53	0.55	0.67	0.7	0.66

Tableau 2-5 : Gain en énergie des cellules TAL2

3.5.3 Placement routage

L’étape de projection technologique fournit une netlist verilog de portes logiques associées à une technologie donnée. Avec cette netlist, il est possible d’utiliser les logiciels commerciaux pour la suite du flot de conception du circuit asynchrone.

Avec la bibliothèque de portes choisies, l’étape suivante est de placer et de router le circuit. Le placement routage est une étape qui permet de positionner et de connecter automatiquement les cellules qui sont instanciées dans la netlist du circuit. A la différence de l’étape de synthèse où le logiciel délivre un fichier qui contient les instances de plusieurs cellules présentes dans la bibliothèque choisie, l’étape de placement routage va positionner physiquement chaque cellule et va créer les fils de

connections entre toutes les cellules. Le placement routage inclut aussi les buffers nécessaires au fonctionnement correct du circuit.

Le placement routage permet alors de connaître l'emplacement physique de chaque composant du circuit. La longueur des fils de connexion entre chaque cellule peut alors être connue, ce qui permettra d'extraire les capacités dans chaque nœud du circuit. Cette étape est cruciale puisqu'elle définit le circuit en soit. C'est une étape qui présente également un problème d'optimisation difficile qui requiert une puissance de calcul importante en fonction de la taille du circuit ainsi que des contraintes imposées pour le placement routage. Ces contraintes peuvent définir la taille totale du circuit, l'emplacement des plots de connexions externes et les anneaux d'alimentation. De ce fait, le temps de calcul peut être très élevé lorsque les contraintes ne sont pas très souples. Si les contraintes sont moins restrictives, le temps de calcul diminue au détriment de son optimisation.

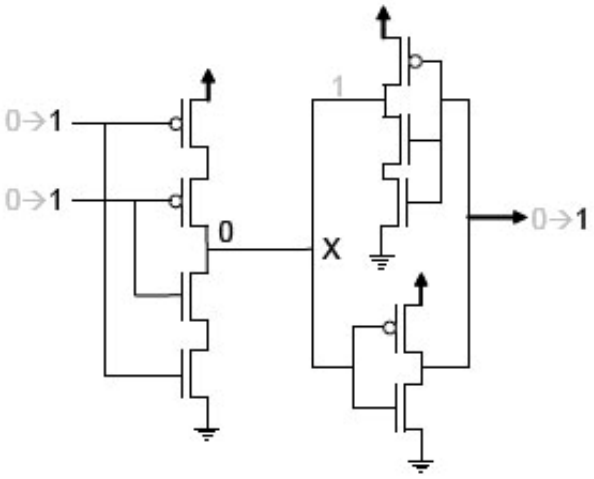


Figure 2-11 : Conflit dans une structure de Muller à 2 entrées.

4 Estimation de l'énergie par le calcul du nombre de transitions

Le chapitre 1 de ce manuscrit a présenté plusieurs techniques pour l'estimation de l'énergie dans un circuit. Chaque technique vise l'énergie à un stade donné du flot de conception. Ce travail propose une technique alternative pour estimer l'énergie des circuits asynchrones QDI. Cette technique est réalisée par simulation du circuit avec des vecteurs d'entrées, le but étant de compter le nombre de transitions à la sortie des portes logiques dans le circuit. Par la suite, ce travail montre les avantages de la technique proposée ainsi que ses limitations.

4.1 Technique de calcul du nombre de transitions dans un circuit

La section 1.1 du Chapitre 1 a exposé les sources de consommation d'énergie dans un circuit CMOS. La consommation dynamique étant la plus importante lors du fonctionnement du circuit. La consommation dynamique est principalement due à la commutation des sorties des portes d'un état haut à un état bas ou bien à la commutation d'un état bas vers un état haut. Ceci est dû à la charge et à la décharge de la capacité de sortie de la porte en question. Il est alors possible d'obtenir une estimation de l'activité et de l'énergie consommée par le circuit avec le nombre de transitions pendant un temps spécifique.

Cette analyse donne une estimation du profil de consommation avec le nombre de transitions pendant un temps donné. Par la suite, avec un rapport détaillé du nombre de transitions dans chaque partie du circuit, le concepteur aura le profil de consommation de son circuit. Avec ce profil, le concepteur identifiera les blocs qui demandent le plus d'énergie et qui pourront alors être optimisés ou bien remplacés par d'autres architectures moins consommantes. Ce premier profil donne seulement une vision globale de l'énergie consommée par le circuit. L'avantage étant que ce profil est obtenu très tôt dans le flot de conception. Le concepteur peut ainsi repérer les optimisations possibles des différentes parties du circuit au début du flot de conception sans qu'il soit nécessaire de réaliser une simulation au niveau transistor pour connaître la consommation du circuit.

La technique proposée dans ce travail consiste à réaliser une simulation à l'aide de l'outil commercial Modelsim de Mentor Graphics. Le circuit simulé doit être une netlist de portes qui peuvent être instanciées en portes basiques à deux entrées ou bien attachée à une technologie donnée, après la projection technologique. Lors de la simulation, les transitions à la sortie des portes sont comptées et enregistrées pour obtenir le profil d'activité du circuit. La Figure 2-12 montre un exemple d'un profil d'activité. À l'aide de ce profil le concepteur peut connaître les moments les plus actifs du circuit ainsi que la répartition des transitions dans le circuit. En effet, le profil d'activité peut être présenté de manière globale pour l'activité du circuit complet mais il est aussi possible de décomposer le profil par blocs du circuit. Le concepteur obtient alors une vision ponctuelle de chaque sous-bloc et est capable de définir les optimisations nécessaires pour réduire l'activité du circuit qui ont pour conséquence une diminution de l'énergie consommée dans celui-ci

4.2 Présentation de l'outil pour compter les transitions

Pour compter les transitions après la simulation du circuit avec le logiciel ModelSim, cette méthodologie utilise un logiciel qui a été développé au sein du groupe CIS du laboratoire TIMA. Ce

logiciel est un module qui s'insère au logiciel Model Sim et qui enregistre toutes les transitions de chaque porte dans la netlist du circuit. Ce module génère un rapport hiérarchique détaillé des transitions de chaque bloc du circuit. Le rapport inclut en détail toutes les cellules présentes dans le circuit avec le nombre de transitions par cellule. Les sous-blocs sont présentés hiérarchiquement en montrant bien l'activité de chaque sous bloc. La Figure 2-12 présente le profil d'activité d'un circuit après analyse par l'outil.

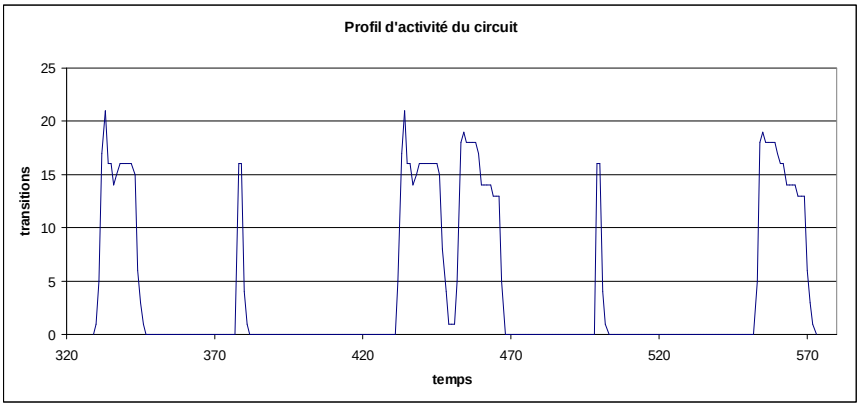


Figure 2-12 : Exemple d'un profil d'activité.

L'outil rend aussi un rapport de la taille du circuit qui est calculé avec les données des cellules qui sont instanciées dans le circuit. Ce rapport n'inclut pas les interconnexions entre cellules, les plots d'entrée et de sortie ni les éventuelles cellules de remplissage. Ces informations n'étant disponibles qu'après l'étape de placement routage du circuit où l'emplacement exact de toutes les cellules est défini et qu'il est possible de connaître la disposition des plots de connexions d'entrée et de sortie pour définir les fils d'interconnexions internes dans le circuit. Néanmoins, ce rapport permet d'avoir un ordre de grandeur de la taille du circuit. Il permet aussi d'obtenir une vision globale de la taille de tous les sous-blocs qui composent le circuit. Le rapport généré a le même format que le rapport des transitions, il présente dans un premier temps la taille totale par type de cellule et finalement la taille par sous-bloc de manière hiérarchique.

4.3 Estimation de l'énergie

Ce travail présente dans la section 1.1 du Chapitre 1, les différentes sources de consommation d'un circuit CMOS. Bien que la consommation statique de nos jours commence à être non négligeable par rapport à la consommation dynamique, cette dernière est encore la source de consommation d'énergie la plus importante. Cette consommation est principalement causée par la charge et la décharge des capacités à la sortie des portes qui commutent lors du fonctionnement du circuit.

Il est alors possible d'obtenir une estimation de l'énergie consommée par le circuit si les capacités de charge de chaque porte qui commutent sont connues. L'énergie consommée lors de la commutation d'une porte peut être calculée avec l'Équation 8. En connaissant toutes les capacités de charge des portes ainsi que leur activité, il est possible d'obtenir un profil d'activité pondéré par la charge de sortie de la porte. L'outil présenté peut fournir en lui donnant la capacité dans chaque nœud du circuit, un rapport équivalent au rapport d'activité présenté dans la section précédente. Cependant ce rapport inclut une pondération à chaque commutation qui varie en fonction de la charge dans le nœud qui a commuté. Le concepteur obtient alors un rapport des transitions pondérées avec la capacité de charge de la porte. L'énergie totale consommée par le circuit peut être calculée avec l'Équation 9. Avec V la tension d'alimentation, C_n la capacité de sortie de la porte n et K_n le nombre de transition de la porte n.

$$E = CV^2$$

Équation 8 : Énergie consommée lors d'une transition dans une porte

$$E_{totale} = V^2 \sum_{n=1} C_n K_n$$

Équation 9 : Énergie totale consommé par le circuit

4.4 Limites de l'outil

L'outil présenté calcule la consommation générée par la commutation des portes due à la charge et à la décharge de la capacité de sortie de la porte. Il ne prend pas en compte l'énergie due aux commutations internes des portes complexes puisque l'outil ne connaît pas la structure interne des portes. Dans certain cas de portes complexes, il existe des nœuds internes qui participent à la consommation du circuit. Bien entendu, la consommation interne des portes n'a pas le même ordre de grandeur que la consommation généré par la capacité de charge, mais cette consommation n'est pas prise en compte dans cette première version de l'outil.

Lors de la consommation dynamique du circuit, la troisième source de consommation d'énergie est générée par les courants de court circuit qui apparaissent pendant un court instant quand dans les circuits CMOS les deux transistors de sortie N et P conduisent en même temps. En effet, lorsque l'état de la sortie de la porte change, soit d'un niveau haut à un niveau bas, soit d'un niveau bas à un niveau haut, le transistor N et le transistor P doivent passer d'un état bloqué à un état passant, et au cours de la transition de ces états, les deux transistors se trouvent dans leur zone saturé. Ceci

génère un courant de fuite appelé “de court circuit” qui traverse les deux transistors de l’alimentation à la masse. La Figure 2-13.a montre le diagramme en transistor d’un inverseur de base. La Figure 2-13.b montre l’état des transistors lorsque l’entrée est à un état bas. Enfin, la Figure 2-13.c montre le phénomène de court circuit lors de la commutation de la sortie d’un inverseur, les deux transistors N et P sont dans leur zone saturée.

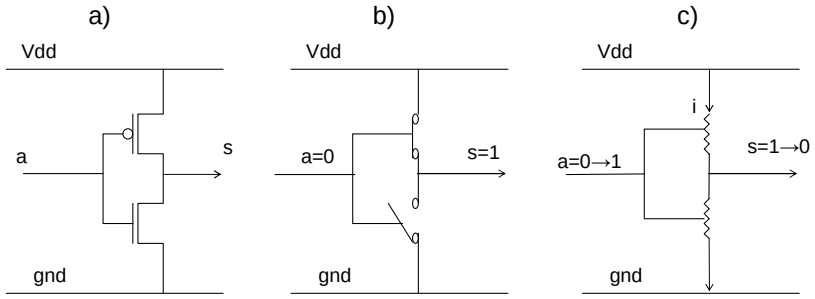


Figure 2-13 : Diagramme d’un inverseur

La consommation due à ce phénomène n’est pas prise en compte par l’outil proposé. Il existe néanmoins une façon d’aborder le problème. La puissance consommée par le courant de court circuit peut être modélisée grâce à une capacité équivalente de court circuit. Cette approche est montrée dans [TUR 95]. L’outil proposé pourrait obtenir cette capacité équivalente de court circuit avec les données des cellules de la bibliothèque sélectionnée. La capacité totale de court circuit viendrait s’ajouter à la capacité commutée préalablement calculée par l’outil pour obtenir une estimation plus correcte de la consommation du circuit analysé.

Finalement, l’outil proposé ne tient pas compte de l’énergie statique consommée par le circuit. Les prochaines versions de l’outil devront prendre en compte cette consommation qui commence à ne plus être négligeable par rapport à la consommation dynamique

5 Conclusion

Ce chapitre présente une méthodologie pour la conception de circuits asynchrones QDI basse consommation. Il montre les différents types de circuits asynchrones actuels dans un premier temps. Le flot de conception présenté permet de générer des circuits asynchrones QDI à partir du langage de haut niveau CHP. L’objectif de la méthodologie étant la conception de circuits basse consommation, le flot proposé permet d’obtenir une estimation de l’énergie consommée avec le nombre de transitions que le circuit effectue en cours de simulation. L’énergie est obtenue en pondérant les transitions avec

les capacités commutées en sortie de chaque porte. Cette capacité peut être obtenue avec les caractéristiques des cellules utilisées ou bien après l'étape de placement routage.

Le chapitre suivant présente la conception de plusieurs blocs d'un microprocesseur MIPS 32 4ksc dans sa version asynchrone. Le flot de conception présenté dans ce chapitre est utilisé pour la conception de ces circuits. Le flot permet alors d'obtenir des informations sur la consommation d'énergie pour comparer les architectures et choisir la moins consommante.

Chapitre 3

ANALYSE D'ARCHITECTURES ASYNCHRONES

Ce chapitre montre la conception de circuits asynchrones QDI. Plusieurs architectures utilisant le flot de conception montré dans le chapitre précédent seront étudiées. Le but étant de comparer ces architectures pour obtenir l'architecture la plus performante. Une brève introduction des critères de comparaison sera présentée. Les circuits conçus sont des sous blocs d'un microprocesseur MIPS32 4kc qui sera présenté. Certains choix lors de la conception de circuits asynchrones QDI seront présentés.

1 Comparaison d'architectures asynchrones.

1.1 Définition des critères de comparaison

Mesurer ou calculer les performances que ce soit en terme de consommation ou bien de vitesse est pratiquement impossible. La mesure des performances est en général une moyenne sur un ensemble de "benchmarks" qui stimulent toutes les parties du circuit. Or, dans le cas général, les processeurs sont utilisés dans des applications spécifiques, et le seul moyen d'obtenir la consommation d'un circuit pour une application précise, est de mesurer la consommation avec les logiciels qui seront utilisés par l'application donnée. Ce travail présente néanmoins trois critères de comparaison pour évaluer les performances des circuits asynchrones. Ces trois critères sont présentés par la suite.

1.1.1 Taille

Dans les années qui suivirent l'apparition des premiers circuits, la taille de ces derniers était une des contraintes majeures à leur conception. Avec la diminution de la taille des transistors, ce paramètre n'a plus eu beaucoup d'importance lorsque la microélectronique s'est focalisée sur les performances des systèmes. Actuellement, la consommation statique des circuits étant de plus en plus élevée, la taille devient à nouveau un problème. En effet, la consommation statique qui dépend de la taille du circuit va être un facteur limitant lors de la conception du circuit dans un futur très proche.

Dans un circuit asynchrone QDI, les mécanismes qui assurent l'insensibilité aux délais ont une conséquence sur la taille du circuit final. De nos jours, les circuits asynchrones sont au moins deux fois plus grand que leurs équivalents synchrones. L'un des problèmes les plus limitants lors de la conception de circuits asynchrones est par conséquent la taille que celui-ci représente par rapport aux circuits synchrones. Il est alors indispensable de prendre en compte la taille des circuits lorsqu'une évaluation d'un circuit asynchrone doit être réalisée.

Pour avoir une mesure de la taille du circuit, il est alors possible d'utiliser plusieurs critères de comparaison dans un circuit. Le premier étant le nombre de portes avec lequel le circuit est constitué. Cette approche est la plus simple à obtenir, or chaque porte est en général conçue avec une taille différente qui varie en fonction de la complexité de la porte en soit. Ceci mène au deuxième critère qui prend plutôt la taille en μm^2 pour avoir une idée plus réaliste de la taille de circuit. Néanmoins, avoir la taille en μm^2 du circuit, ne sert qu'à comparer deux circuits conçus avec la même technologie. Pour pouvoir comparer la taille de deux circuits qui sont conçus avec des technologies différentes, il est possible de reporter la taille du circuit en nombre de porte équivalente AN2LL. Cette porte est une porte standard AND à deux entrées. Avec Le nombre de portes équivalentes, il est alors possible de comparer deux circuits même s'ils sont conçus avec des technologies différentes. Ce travail présente pour chaque circuit analysé, les trois critères de comparaison :

- Nombre de cellules
- Surface en μm^2 (dans cette étude les circuits sont réalisés avec une technologie de 130nm)
- Nombre de portes AN2LL équivalentes

1.1.2 Vitesse

Dans les systèmes actuels, le nombre d'applications qui sont exécutées est de plus en plus élevé et la complexité de plus en plus grande. Les systèmes doivent assurer un traitement des données en temps réel pour de nombreuses applications. Le deuxième critère pour évaluer le fonctionnement d'un circuit est alors la vitesse à laquelle celui-ci exécute les tâches qui lui sont demandées. Un circuit

de taille très réduite qui ne consomme quasiment pas d'énergie et qui doit réaliser un traitement d'image en temps réel ne peut pas se permettre de traiter une image toutes les secondes dans un flux vidéo.

Or, les circuits asynchrones ne possèdent pas de signal d'horloge comme les circuits synchrones, il est donc plus difficile de connaître leur vitesse de fonctionnement lors de la conception du circuit, contrairement aux circuits synchrones où celle-ci peut être calculée avec la fréquence d'horloge. Dans un circuit asynchrone il existe deux manières pour estimer la vitesse de fonctionnement du circuit. La première étant en réalisant une simulation avec des vecteurs d'entrée et la deuxième en réalisant une analyse statique de temps.

Il faut aussi noter que la vitesse avec laquelle les circuits asynchrones exécutent les instructions d'un programme varie en fonction de celles-ci. Puisque pour exécuter une instruction dans un circuit asynchrone, seul les blocs qui sont nécessaires au calcul de l'instruction seront actifs, chaque instruction dans un processeur aura son chemin et par conséquent le délai d'exécution sera différent. Dans un circuit asynchrone la vitesse calculée par simulation ou bien par mesure directe sur le circuit est une vitesse moyenne qui dépend de l'application utilisée.

Dans ce travail, la vitesse en puissance de calcul du circuit asynchrone est calculée en mesurant le temps de cycle par instruction. Le temps de cycle débute lorsque le bloc reçoit toutes les données nécessaires au calcul, et se termine à la fin du protocole quatre phases. En ce qui concerne la vitesse pour le circuit synchrone, celle-ci est donnée avec la fréquence d'horloge du circuit.

1.1.3 Consommation

L'objectif de ce travail consiste à analyser la conception de circuits asynchrones QDI basse consommation, aussi le dernier critère de comparaison concerne la consommation des circuits. La comparaison de consommation d'énergie entre les circuits asynchrones et les circuits synchrones est l'un des critères les plus étudiés par la communauté scientifique qui étudie les circuits asynchrones. Ceci est dû au fait que la consommation d'énergie dans les systèmes actuels est l'un des problèmes les plus importants lors de la conception des systèmes sur puce.

Dans un circuit asynchrone, l'énergie consommée ne dépend que du type d'instructions. Les vecteurs d'entrée n'ont pas une forte incidence sur l'énergie consommée par le processeur pour une instruction donnée. A la différence des circuits synchrones, lorsqu'un processeur asynchrone doit exécuter une nouvelle instruction, toutes les entrées et sorties des blocs qui composent le processeur passe à un état invalide. Cet état est, en général, un niveau logique 0 sur les fils d'interconnexions des blocs. Lorsque l'instruction est exécutée et que les données traversent le circuit, elles font basculer les

portes qui composent le chemin de traversée. Une fois que la sortie a été calculée et qu'une nouvelle instruction est prête à être exécutée, il faut remettre les entrées et les sorties à l'état invalide. Ces deux étapes entraînent une consommation d'énergie qui ne dépend que du chemin parcouru par les données dans le circuit. Les données n'ont quasiment pas de répercussions sur la consommation d'énergie. Dans un circuit synchrone, la consommation dépend des valeurs précédentes puisque ce type de circuit n'a pas une étape de remise à zéro comme les circuits asynchrones. Si la nouvelle valeur ou si certaines parties de la nouvelle valeur ne changent pas par rapport à la précédente, alors les portes nécessaires au calcul du résultat ne consomment pas d'énergie dynamique car il n'est pas nécessaire de charger ou de décharger les capacités à la sortie des portes.

Pour donner la consommation d'énergie d'un circuit synchrone, avec une méthode par simulation, il est nécessaire d'utiliser des vecteurs d'entrée qui soient les plus réalistes possibles pour avoir une estimation de l'énergie moyenne consommée par le circuit. Des benchmarks existent déjà [SPE 08] pour faire travailler les processeurs avec certaines applications qui excitent des parties spécifiques dans le processeur.

Ce travail utilise l'énergie par instruction exécutée pour comparer les différentes architectures asynchrones qui seront montrées ainsi que pour comparer les circuits asynchrones avec leur équivalent synchrone. Cette énergie est donnée par l'Équation 3-1. Avec C_{total} étant la capacité commuté totale du circuit après simulation, N étant le nombre d'instructions exécutées et V la tension d'alimentation.

$$E_i = \frac{C_{total} V^2}{N}$$

Équation 3-1 : Énergie consommée par instruction.

1.2 Synthèse des circuits asynchrones QDI

Dans cette partie une brève explication de la synthèse des circuits asynchrones QDI sera présentée. Il existe plusieurs manières de réaliser la synthèse de circuits asynchrones, cette étude montre la synthèse de circuits asynchrones QDI en utilisant la méthode avec des Diagrammes de Décisions Multi-valué (MDD).

1.2.1 Synthèse à partir d'un diagramme de décisions multi-valué

Pour synthétiser un circuit asynchrone QDI, l'outil TAST utilise le format intermédiaire basé sur les réseaux de Pétri. TAST génère dans un premier temps le circuit avec un format en réseau de Pétri qui est obtenu à partir de la description du circuit avec le langage CHP. La deuxième étape

consiste à transformer ce réseau de Pétri en une description du circuit à l'aide d'un Diagramme de Décisions Multi-valué (MDD) [KAM 98, BAB 00]. La génération de ce Diagramme de Décisions Multi-valué est présentée dans le travail de thèse de Vivian Brégier [BRE :07]. Une fois le MDD généré, il est possible de réaliser des optimisations sur le MDD initial qui vont réduire la taille du MDD et ainsi la taille du circuit final. Un exemple d'un MDD est présenté dans la Figure 3-1. Ce diagramme implémente la fonction "et" logique entre deux opérandes A et B. L'idée de ce diagramme est de spécifier la valeur de la sortie en fonction de toutes les combinaisons possibles des entrées.

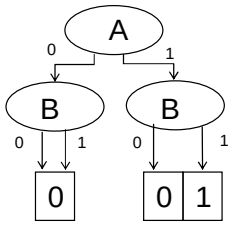


Figure 3-1 : Exemple de MDD

Avec ce diagramme de décisions multi-valué (MDD), le circuit peut être généré à l'aide de portes de Muller et de portes "ou" logiques. Les circuits générés avec cette méthode présentent une série d'étages de portes de Muller pour recevoir toutes les entrées. La sortie est alors construite avec des étages de portes "ou" logiques. Cette méthode de synthèse garantit que le circuit obtenu est QDI.

Dans l'exemple qui suit, les opérations logiques propres au langage CHP n'ont pas été utilisées (AND, OR, XOR et NOR). Le circuit est alors décrit avec une série de gardes déterministes qui calculent le résultat en fonction des entrées. Dans ce cas, il existe trois entrées sur chaque bloc qui génère un digit de deux bits (A, B et cmd). Les trois entrées utilisées se composent des deux opérandes qui seront utilisées pour réaliser l'opération logique, et de la commande qui indique le type d'opération à exécuter.

1.2.2 Exemple de synthèse sur un circuit 2 bits double rails.

Pour comprendre la manière dont sont décrit les circuits avec le langage CHP, le circuit dont la table de vérité est montrée dans la Figure 3-2, est décrit avec le langage CHP dans la Figure 3-3. Ce circuit réalise une opération AND ou une opération OR avec les deux bits d'entrées A et B. La description du circuit avec le langage CHP peut se faire de plusieurs manières. Pour pouvoir imposer la façon dont le circuit va être généré et ainsi pouvoir comparer les performances en termes de taille et de consommation d'énergie, les opérations propres au langage CHP ne sont pas utilisées. Ceci permet de montrer l'importance du choix de l'ordonnancement des entrées lors de la synthèse du circuit.

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

Figure 3-2 : Table de vérité

```
        component
ALU_LOGICAL_SLICE_ELEMENT
            port(
                cmd  : in DI MR[2][1];
                A    : in DI MR[4][1];
                B    : in DI MR[4][1];
                S    : out DI MR[4][1])

        variable cmd_var   : MR[2][1];
        variable A_var     : MR[4][1];
        variable B_var     : MR[4][1];
        begin
        [ cmd ? cmd_var,
        A ? A_var,
        B ? B_var;
          @[
          cmd_var = "0"[2] =>
    ; AND
                @[
                A_var = "0"[4] =>
                    @[
                    B_var = "0"[4] =>
                        S ! 0;
                        Break
                    B_var = "0"[4] =>
                        S ! 0;
                        Break ];
                    break
                A_var = "1"[4] =>
                    @[
                    B_var = "0"[4] =>
                        S ! 0;
                        Break
                    B_var = "1"[4] =>
                        S ! 1;
                        Break ];
                    Break ];
                Break ];

            cmd_var = "1"[2] =>
    ; OR
                @[
                A_var = "0"[4] =>
                    @[
                    B_var = "0"[4] =>
                        S ! 0;
                        Break
                    B_var = "1"[4] =>
                        S ! 1;
                        Break ];
                    Break
                A_var = "1"[4] =>
                    @[
                    B_var = "0"[4] =>
                        S ! 1;
                        Break
                    B_var = "1"[4] =>
                        S ! 1;
                        Break ];
                    Break ];
                Break ];
        Loop ];
        end;
```

Figure 3-3 : Exemple de circuit pour réaliser une opération AND et OR

Avec cette description en langage CHP, l'outil de synthèse effectue dans un premier temps un MDD qui est montré dans la Figure 3-4.a. Le MDD généré peut être optimisé et donne le MDD présenté dans la Figure 3-4.b. A partir de ce MDD, il réalise la synthèse du circuit en portes standard à deux entrées (Figure 3-5). Sur cet exemple, l'entrée cmd et l'entrée A sont traitées en premier, la racine de l'arbre étant le signal de commande (cmd), le premier opérande étant un fils de la racine et enfin, le deuxième opérande est à son tour fils du premier opérande. Dans ce circuit, la génération de la sortie commence par des portes de Muller qui reçoivent la commande et l'un des deux opérandes. Ceci est dû à la manière dont a été décrit le circuit au niveau du langage CHP. Il est possible d'inverser les entrées de façon à ce que les optimisations au niveau du MDD réduisent ses branches. Dans cet exemple, il faudrait que le MDD ait comme racine l'un des deux opérandes et que les nœuds suivants

soient constitués par le deuxième opérande ou par la commande. Finalement, le MDD génère la sortie en fonction de la valeur de la dernière entrée.

Dans le circuit présenté, ceci n'a aucune répercussion car le circuit est trop petit, il permet tout simplement d'illustrer la façon dont le circuit est conçu. Dans le cas de circuits plus grands, le choix de l'ordre des commandes est très important pour ce type de conception de circuits asynchrones QDI. Ce choix a une conséquence sur la taille du circuit, puisque si le nombre de branches du MDD est réduit, le circuit sera alors plus petit.

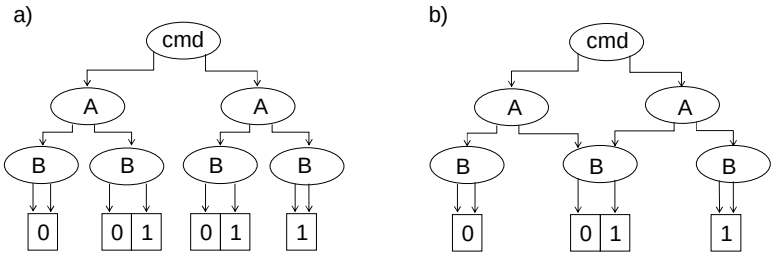


Figure 3-4 : MDD du circuit présenté et optimisation

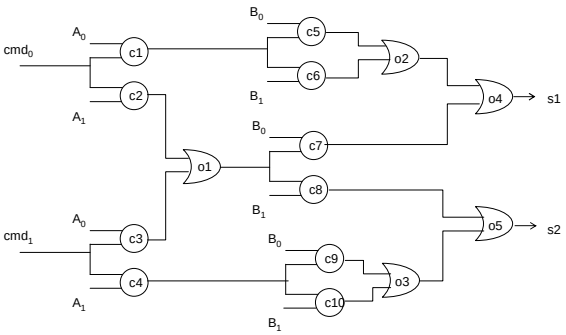


Figure 3-5 : circuit en portes standards a deux entrées

1.2.3 Implémentation du protocole quatre phases

La communication entre blocs dans cette étude utilise un protocole quatre phase. Pour assurer le protocole quatre phase, les circuits asynchrones utilisent l'équivalent d'un registre pour les circuits synchrones qui est connu comme le Half-Buffer. La Figure 3-6 montre le fonctionnement d'un circuit asynchrone QDI. Entre chaque bloc asynchrone, un Half-Buffer permet de mémoriser localement les données.

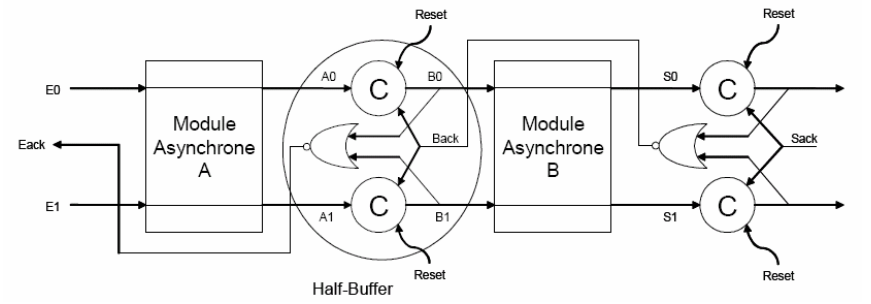


Figure 3-6 : Half-Buffer entre de blocs asynchrones QDI

1.3 Comparaison de circuits synchrone et asynchrone

Pour comparer les différences entre un circuit synchrone et son équivalent asynchrone, ce travail présente un circuit simple qui réalise quatre opérations logiques, AND, OR, XOR et NOR sur deux bits qui seront codés dans le circuit asynchrone en quatre rails.

Le code CHP montré dans l'annexe A est le code CHP de l'opérateur pour le circuit asynchrone. L'annexe B présente le code VHDL pour le circuit synchrone. Les deux circuits sont synthétisés, le circuit asynchrone avec l'outil TAST et le circuit synchrone avec l'outil Design Vision. Après l'étape de projection technologique et le placement routage des circuits, le circuit asynchrone est composé de 35 portes complexes ce qui correspond à 93 portes AN2LL, la taille du circuit étant de 941µm^2. Pour l'équivalent synchrone, le circuit compte 11 portes complexes ce qui correspond à 15 portes AN2LL et a une taille de 153µm^2.

Pour obtenir la consommation du circuit, ce dernier a été simulé avec un benchmark composé de 64 instructions qui réalisent 16 instructions de chaque opérateur logique (AND, OR, XOR et NOR). Les entrées A et B sont générées aléatoirement. Le circuit asynchrone, a une capacité commutée totale de 15,6pF ce qui correspond a une consommation de 22,46pJ avec une tension d'alimentation de 1,2V. L'énergie par instruction pour le circuit asynchrone est alors de 351fJ/instruction. Le circuit synchrone a une capacité commutée totale de 6,02pF. La consommation du circuit est alors de 8,66pJ, qui correspond à une consommation de 135.5fJ/instruction.

Les résultats obtenus pour les circuits à deux bits montrent que la consommation du circuit asynchrone est 60% plus élevée par rapport au circuit synchrone. Dans un circuit synchrone, la consommation augmente en fonction de la taille du circuit car l'arbre d'horloge se complexifie. Dans la section 1.4.2.1.3 ce travail présente comment la consommation du circuit synchrone augmente en fonction de la taille de celui-ci. Cette augmentation n'est pas linéaire contrairement aux circuits

asynchrones. Ce travail montrera alors que les circuits asynchrones consomment moins d'énergie par rapport aux circuits synchrones.

1.4 Présentation de différents blocs d'un microprocesseur

Cette partie présente deux sous blocs d'un processeur MIPS32 4KSc. La conception du processeur est de type QDI avec des canaux de données quatre rails et un protocole de communication quatre phases.

1.4.1 Présentation du processeur MIPS 32 4KSc

Le processeur MIPS32 4kc est un processeur de la famille des processeurs MIPS [MIP 07]. MIPS est un acronyme pour Microprocessor without Interlocked Pipeline Stages. Il a été développé au début des années 80 par l'équipe de John Hennessy à l'université de Standford. En 1984 John Hennessy, Robert Wall et John Moussouris fondent MIPS Computer Systems, qui sera acheté par SGI en 1992 et dont le nom changera pour MIPS Technologies. Ce processeur est actuellement très utilisé dans des systèmes embarqués tels que ; les assistants personnels plus connus sous l'acronyme PDA (Personal Digital Assistant), les routeurs Cisco, les consoles de jeux vidéo comme la Nintendo 64, la Sony Playstation et bien d'autres systèmes encore.

L'architecture MIPS est une architecture de type RISC (Reduced Instruction Set Computer). Le processeur 4KSc est un processeur 32 bits qui contient un banc de 32 registres. Le cœur 4KSc utilise le SmartMIPS ASE qui permet d'avoir de nouvelles instructions pour les applications de cryptographie. La Figure 3-7 montre le diagramme de blocs de l'architecture du processeur.

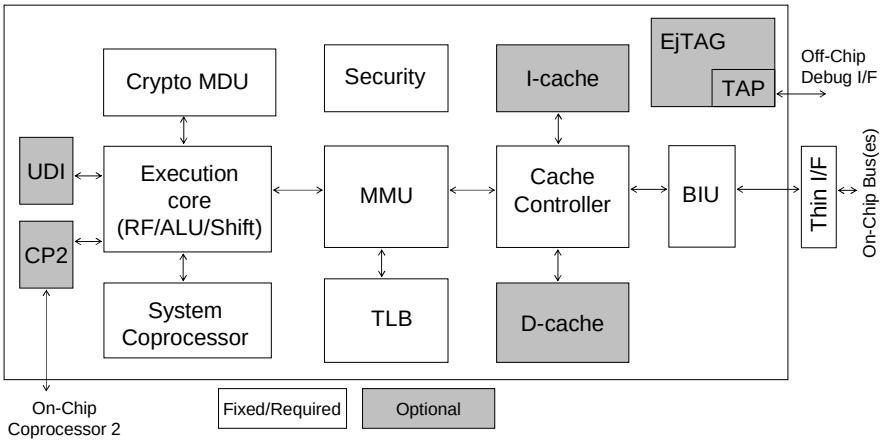


Figure 3-7 : Architecture du MIPS asynchrone

Ce travail étudie le bloc d'exécutions pour réaliser la version asynchrone du MIPS32 KSc. La Figure 3-8 montre le diagramme de blocs de l'unité d'exécution. Tout le circuit sera conçu avec des données codées en quatre rails avec un protocole de communication quatre phases. Cette unité a été entièrement spécifiée et codée avec le langage de haut niveau CHP. Pour réaliser la synthèse de ce circuit, plusieurs choix d'architecture ont été opérés. Ce travail expose par la suite, quelques considérations au niveau des architectures afin d'obtenir un circuit asynchrone QDI le moins consommant possible. Une étude approfondie de l'unité de décalage et de l'unité logique est réalisée et les résultats sont présentés dans la section suivante.

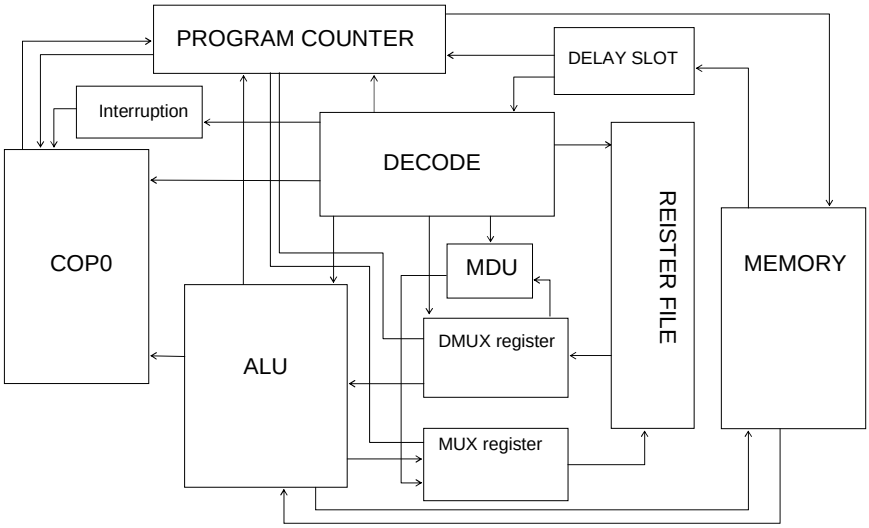


Figure 3-8 : Diagramme de blocs de l'unité d'exécution

1.4.2 Etude des architectures

Dans cette section, l'unité logique et l'unité de décalage du processeur MIPS 32 4KSc sont analysées. Le but étant de montrer plusieurs types de structures pour la conception de ces blocs. Avec l'unité logique, l'importance du choix de l'ordonnancement des entrées lors de la génération du MDD sera montrée. Pour l'unité de décalage trois différentes structures seront présentées et l'analyse montrera les avantages et les inconvénients pour chaque structure.

L'équivalent synchrone pour les deux architectures sera aussi analysé et une comparaison entre les circuits asynchrone et synchrone sera présentée.

1.4.2.1 Unité logique

L'unité logique permet l'exécution des instructions logiques AND, OR, NOR et XOR. Le bloc a été divisé en 16 sous blocs qui calculent digit par digit le résultat de l'instruction en cours. Il est

important de noter que l'unité logique réalise des opérations sur 32 bits, comme le codage des données est réalisé en quatre rails, chaque digit code 2 bits. Pour coder les 32 bits il est nécessaire d'avoir 16 digits. Les 16 sous blocs qui composent l'unité sont identiques puisque le résultat de chaque digit de toutes les instructions ne dépendent que des valeurs d'entrées par digit. Autrement dit, il n'y a pas de dépendance entre digits pour obtenir le résultat des 32 bits. La Figure 3-9 montre le diagramme de blocs de l'unité logique. Il a été décrit avec le langage haut niveau CHP et a été simulé avec le logiciel TAST. Les 16 blocs de calcul effectuent les opérations logiques sur les deux entrées quatre rails pour fournir un résultat en quatre rails. Ceux-ci réalisent donc une opération logique de deux bits, pour obtenir ainsi un résultat sur 32 bits.

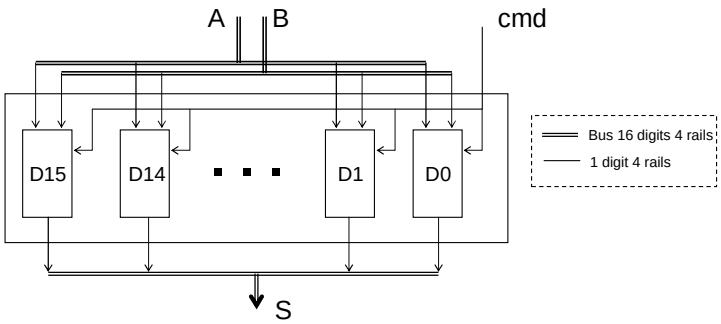


Figure 3-9 : Diagramme de blocs de l'unité logique

Pour montrer l'importance du choix de l'ordonnancement des entrées lors de la génération du MDD, ce travail présente trois versions du bloc logique. Ce bloc qui reçoit trois entrées peut être généré de six façons différentes. Néanmoins, les deux entrées des opérandes étant les mêmes en ce qui concerne la génération du résultat, le circuit peut être généré de trois façons différentes (Figure 3-10).

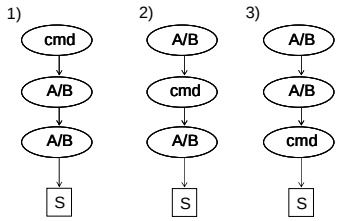


Figure 3-10 : ordonnancement des entrées pour générer le bloc logique

Ces trois versions ont été synthétisées dans un premier temps avec des portes standard à deux entrées. La taille en nombre de portes avant la projection technologique est de 1877 (2747 portes AN2LL), 1877 (2747 portes AN2LL) et 1541 (2258 portes AN2LL) portes standard à deux entrées pour les circuits 1, 2 et 3. Ceci correspond respectivement à une taille de 27749µm^2, 27749µm^2 et 22805µm^2 en utilisant une technologie de 130nm de STMicroelectronics. Les circuits 1 et 2 sont identiques puisque les combinaisons à la sortie des nœuds présents sur le MDD après la deuxième

entrée sont forcément les mêmes. En effet, le premier étage de portes de Muller doit recevoir les mêmes entrées dans les deux cas.

Pour tester le fonctionnement correct des circuits, un benchmark a été créé pour stimuler les quatre instructions réalisées par l'unité logique. Ce benchmark généré avec des entrées aléatoires est formé de 64 instructions. Il permet d'exécuter 16 instructions de chaque type d'opération.

Puisque les circuits 1 et 2 sont les mêmes, seuls les circuits 1 et 3 seront étudiés par la suite. En suivant le flot de conception proposé dans le Chapitre 2, après l'étape de projection technologique utilisant la librairie TAL, les circuits comptent 757 (1765 portes AN2LL) et 544 (1485 portes AN2LL) portes complexes et ont respectivement une taille de 17829µm^2 et 15003µm^2. Ceci représente une diminution de la taille du bloc de 35.7% pour le circuit 1 et de 34.2% pour le circuit 3 par rapport aux circuits en portes standard à 2 entrées. Le nombre de transitions est aussi calculé pour les deux versions de chaque circuit. Le résultat obtenu est une activité de 19264 et 19157 transitions pour les circuits en portes standard à deux entrées. Les circuits en portes complexes génèrent une activité de 12819 et 11528 transitions. L'étape de projection technologique permet non seulement de réduire le nombre de transitions mais aussi la taille du circuit.

Le même bloc a été synthétisé pour l'équivalent synchrone. La Figure 3-11 montre le diagramme de blocs du circuit synchrone de l'unité logique. Ce circuit a été décrit avec le langage de haut niveau VHDL, il a été simulé avec le logiciel Modelsim et synthétisé avec l'outil Design Vision. Il est composé de 247 portes complexes (311 portes AN2LL) et a une taille de 3146µm^2. Le nombre de transitions en utilisant le même benchmark que pour le circuit asynchrone est de 12129.

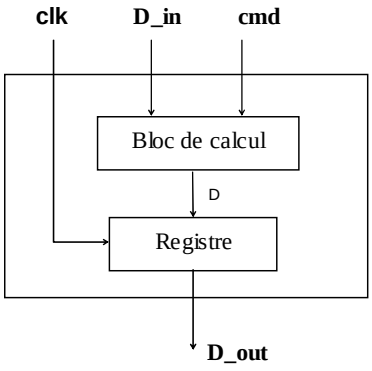


Figure 3-11 : Diagramme de bloc de l'unité logique synchrone

1.4.2.1.1 Consommation.

Pour obtenir la consommation d'énergie du circuit de l'unité logique, il est nécessaire de connaître la capacité commutée dans le circuit. Pour cela, il faut réaliser le placement routage du

circuit. Cette étape permet d'obtenir la charge à la sortie de chaque porte, celles-ci seront choisies en fonction du nombre de portes en amont. En utilisant le même benchmark, en prenant en compte la capacité de charge de chaque porte et en pondérant chaque transition de chaque porte, la capacité commutée totale des circuits est de 313,24pF et 281,52pF. En utilisant l'Équation 3-1 avec une tension d'alimentation de 1,2V et N étant le nombre d'instructions pour le benchmark donné, l'énergie consommée par instruction dans les deux circuits est de 7,04pJ/inst et 6,3pJ/inst respectivement. La vitesse d'exécution étant de 65 et 72MIPS en moyenne.

Pour l'équivalent synchrone de l'unité logique, en utilisant les mêmes vecteurs d'entrée, le circuit a une capacité totale de 910,06pF ce qui donne une consommation de 20pJ/inst. Avec l'outil Design vision en utilisant les conditions pire cas (worst) le chemin critique est de 2,66ns qui correspond à une vitesse maximale de fonctionnement de 375,9MHz, l'annexe C montre le rapport de temps généré par l'outil.

1.4.2.1.2 Circuit logique double rail

Puisque l'unité logique a été réalisée avec un protocole de communication en codage quatre rails dans les sections précédentes, il est intéressant de le comparer avec un équivalent codé en double rails. Le circuit 3 utilisé pour la version à quatre rails est utilisé par la suite. La même méthodologie est utilisée pour synthétiser et simuler le circuit obtenu. Après synthèse, et en réalisant la projection technologique, le circuit comporte 427 portes complexes (914 AN2LL). Le même vecteur d'entrée utilisé pour le circuit quatre rails est utilisé. Le circuit effectue alors 23801 transitions. Avec le placement routage et en pondérant les transitions avec les capacité de charge, le circuit a une taille de 9536 μm^2, comporte 456 portes logiques (944 AN2LL) et a une capacité commuté totale de 375,51pF ce qui correspond à une consommation de 8,44pJ par instruction.

1.4.2.1.3 Consommation d'énergie en fonction de la taille du circuit

La consommation du circuit synchrone dans l'exemple précédent qui réalise une opération logique sur 32 bits est de 20pJ/instruction. Le même circuit logique à 2bits présenté dans la section 1.3 a une consommation de 135fJ/instructions. Le circuit à 32bits ne présente pas une augmentation linéaire par rapport à la consommation du circuit à 2bits. Dans le cas du circuit asynchrone, la consommation par rapport au circuit à 2bits est linéaire. L'arbre d'horloge dans les circuits synchrones implique une augmentation qui n'est pas linéaire en fonction de la taille du circuit. Les circuits asynchrones présentent l'avantage d'avoir une consommation qui est linéaire en fonction de la taille du circuit.

Pour montrer la consommation des circuits synchrones en fonction de la taille de ces derniers, les Figure 3-12 et Figure 3-13 montrent la consommation des circuits synchrone et asynchrone qui effectuent les opérations logiques sur des entrées à 2, 8, 16, 32 et 64 bits. Dans la Figure 3-13 il est possible de voir le croisement des courbes. Le circuit asynchrone consomme moins d'énergie lorsque le nombre de bits est supérieur à 8 bits. La consommation des deux circuits en fonction du nombre de bits peuvent être modélisées par l'Équation 3-2 et l'Équation 3-3, avec E l'énergie consommée et δ le nombre de bits du circuit. L'insertion de buffers ainsi que l'augmentation de la charge dû à l'arbre d'horloge, pénalise de manière quasi quadratique l'énergie consommée par les circuits synchrones.

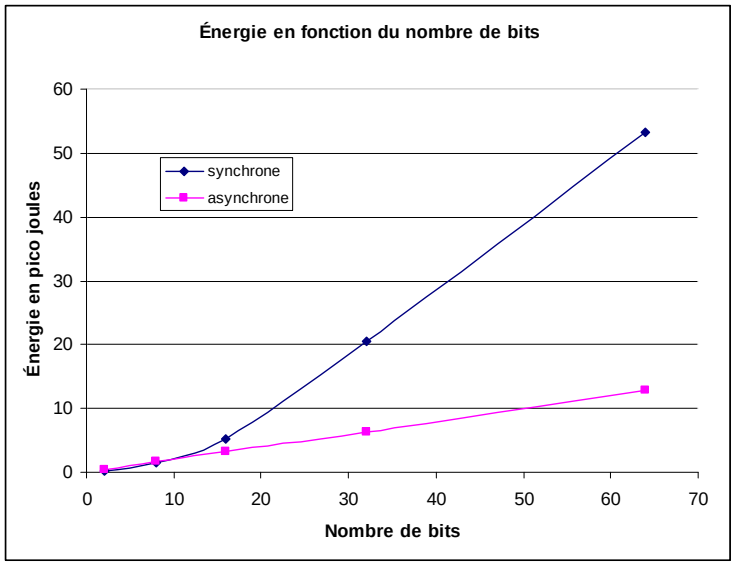


Figure 3-12 : Énergie consommée en fonction du nombre de bits

$$E = 41\times10^{-3} * \delta^{1,74}$$

Équation 3-2 : Énergie du circuit synchrone en fonction du nombre de bits

$$E = -16\times10^{-3} + 199.7\times10^{-3} * \delta$$

Équation 3-3 : Énergie du circuit asynchrone en fonction du nombre de bits

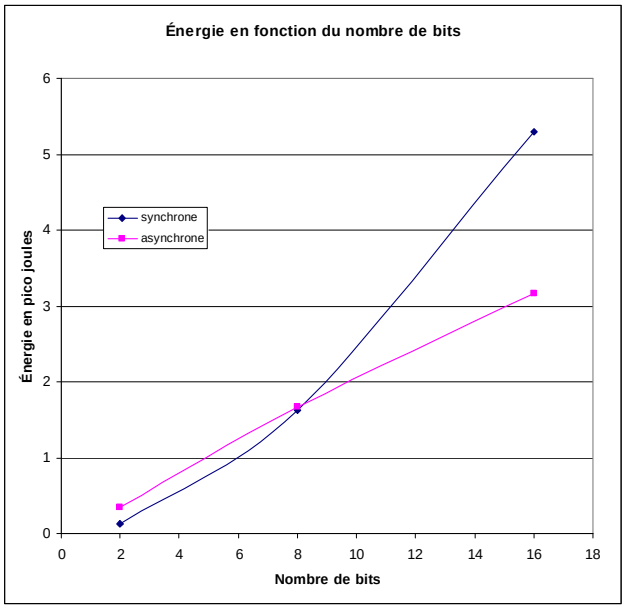


Figure 3-13 : Énergie consommée en fonction du nombre de bits

1.4.2.1.4 Capacité de charge

En comparant la taille des deux circuits synchrone et asynchrone de l'unité logique à 32 bits, le circuit asynchrone a une taille de 15003µm^2 et l'équivalent synchrone présente une taille de 3146µm^2 ceci correspond à un facteur de 4,7 fois plus grand pour le circuit asynchrone. Néanmoins, le circuit asynchrone consomme 3,1 fois moins d'énergie par rapport au circuit synchrone. Ceci est principalement dû au nombre de transitions mais surtout à la capacité totale lors de l'exécution d'une instruction. Le Tableau 3-1 montre comment sont réparties les capacités à la sortie de chaque porte qui compose les deux circuits. Pour le circuit asynchrone, la totalité des capacités est inférieure à 0,1pF. En revanche pour le circuit synchrone il existe 15 portes qui présentent une capacité très élevée. Ces portes sont utilisées lors de l'exécution de chaque instruction, ce qui explique que le circuit synchrone consomme plus que le circuit asynchrone.

nombre de portes	Capacité à la sortie de la porte en pF				
	0 -> 0,1	0,1 -> 0,5	0.5 -> 1	1 -> 2	2 -> 3
asynchrone	447	97	0	0	0
synchrone	230	2	0	13	2

Tableau 3-1 : Répartition des capacité à la sortie de chaque porte

1.4.2.1.5 Conclusions

Le Tableau 3-2 présente un récapitulatif des circuits analysés dans cette section. Cette étude montre l'importance du choix de l'ordonnancement des entrées pour réaliser la synthèse avec la

méthode par MDD. Les optimisations au niveau de la génération du MDD permettent de réduire la taille ainsi que la consommation du circuit. Dans l'exemple proposé, le choix des entrées réduit de 17,8% la taille du circuit numéro 3 par rapport au circuit numéro 1. En terme de consommation le gain étant de 10,5%.

L'étape de projection technologique permet aussi de réduire considérablement la taille du circuit. La Librairie TAL étant composée de cellules complexes asynchrones, l'optimisation du circuit en taille dans l'unité logique présentée est de 35,7% pour le circuit numéro 1 et de 34,2% pour le circuit numéro 3.

circuit	portes AN2LL	transitions	énergie/instruction en pJ
Logical V1	2747	19264	
V1 après techmap	1765		7.04
Logical V2	2747	19264	
Logical V3	2258		
V3 après techmap	1485	19157	6.3
logical sync	311	12129	20

Tableau 3-2 : récapitulatif de l'unité logique

La consommation du circuit asynchrone est de 6,3pJ/instruction pour le circuit numéro 3 à une tension d'alimentation de 1,2V. Il est possible de réduire la tension d'alimentation pour réduire la consommation d'énergie, ceci sera présenté dans le Chapitre 4. Dans l'étude proposée dans le Chapitre 4, l'alimentation minimale peut être de 0,54V, ce qui correspondrait alors à une consommation de 1,2pJ pour ce circuit. Poussé à l'extrême, cette tension pouvant être réduite jusqu'à 36mv [MEI 00], la consommation minimale que le circuit peut atteindre est alors de 5,7fJ.

Le circuit en double rail permet de réduire la taille du circuit. Dans le circuit étudié, le circuit en double rail est 36% plus petit par rapport au circuit 3 de l'unité logique. En revanche, la consommation du circuit double rail est 25,3% plus élevée que pour le circuit 3 montrée dans l'étude.

Pour l'équivalent synchrone de l'unité logique, le circuit est 4,7 fois plus petit. Or comparer la taille entre les deux blocs est très compliqué, puisqu'un circuit de cette taille n'expose pas entièrement le problème de l'arbre d'horloge. Dans un circuit plus grand, le nombre de buffers inséré augmenterait la taille du circuit. Il reste évident bien sûr que le circuit synchrone sera toujours plus petit. Néanmoins, avec la taille du circuit obtenue, l'étude montre bien que le circuit asynchrone a une consommation inférieure à celle du circuit synchrone. En effet, cette consommation est 3,17 fois plus petite. Bien entendu, un circuit plus grand en synchrone avec la génération complète de l'arbre d'horloge, viendra augmenter ce facteur.

1.4.2.2 Unité de décalage (Shift)

L'unité de décalage permet de réaliser les huit opérations de décalage suivantes : SRL, SRLV, SLL, SLLV, ROTR ROTRV, SRA et SRAV. Les quatre premières sont des opérations de décalage à gauche et à droite logique, les deux suivantes permettent d'effectuer une rotation à droite et les deux dernières réalisent un décalage à droite arithmétique qui permet de propager le bit de poids fort pour conserver le signe de la donnée. Les instructions SRL, SLL, ROTR et SRA utilisent deux registres comme opérandes. Le premier étant le registre qui contient la valeur à décaler, et le deuxième contient le nombre de bits à décaler. Les quatre autres instructions utilisent une valeur immédiate sur 5 bits pour déterminer le nombre de bits à décaler. Cette valeur est codée dans le opcode de l'instruction. Les instructions logiques de cette unité permettent de réaliser un décalage de 0 à 31 bits en insérant un zéro à gauche ou à droite en fonction de l'instruction. Avec les instructions arithmétiques, le décalage est effectué en respectant le signe de la valeur.

L'opération de décalage peut se concevoir avec deux architectures de base différentes. La première est une architecture logarithmique qui décale la donnée de 1, 2, 4, 8 et 16 bits pour décaler jusqu'à 31 bits. Cette architecture contient plusieurs blocs qui sont en série. La deuxième architecture est composée d'une machine à état qui permet de reboucler sur un seul bloc pour réaliser le décalage de 31 bits. Les deux architectures seront présentées par la suite.

1.4.2.2.1 Structure logarithmique

La structure de base pour cette architecture consiste, comme mentionné dans le paragraphe précédent, à concevoir 5 blocs qui réaliseront des décalages de 1, 2, 4, 8 et 16 bits. Le désavantage de cette architecture est le nombre de blocs nécessaires pour réaliser le décalage. Ce travail propose de rassembler quelques blocs pour avoir une structure avec moins de blocs internes, ce qui permettra d'étudier l'impact du regroupement des fonctions en un seul bloc dans les circuits asynchrones. Les deux structures choisies sont présentées par la suite.

1.4.2.2.1.1 Structure à 5 blocs

La première structure analysée est la structure de base qui contient 5 blocs. La Figure 3-14 montre le diagramme de blocs de cette architecture. L'unité reçoit une donnée sur 32 bits codée en 16 digits quatre rails. La commande est un digit codé aussi en quatre rails. Le nombre de bits à décaler est un bus de 3 digits dont deux digits sont codés en quatre rails, et le troisième est un double rails, pour obtenir ainsi les 5 bits nécessaires qui déterminent le nombre de bits à décaler.

Les blocs qui constituent l'unité de décalage, reçoivent la donnée à décaler et les commandes qui indiquent l'opération qui doit être réalisée. Les 5 blocs sont constitués de 16 sous blocs qui

réalisent le décalage. La Figure 3-15 montre la structure interne des blocs de décalage. Ce diagramme présente spécifiquement l'étage de décalage de 1 bit. Les autres étages sont similaires mais l'interconnexion des canaux de communication entre blocs est différente. Dans cet exemple, le sous bloc D_0 est relié aux sous blocs D_1 et D_15. Dans le cas de l'étage de décalage de 2 bits, le même sous bloc D_0 sera relié aux sous bloc D_2 et D_14 et ainsi de suite pour les autres étages. Chaque sous bloc reçoit un digit de la donnée et cette dernière est traitée en fonction de la commande qui provient du bloc de commande. Tous les blocs ont été décrits avec le langage CHP. Dans cette architecture, un seul Half-Buffer est utilisé à la sortie du bloc X16.

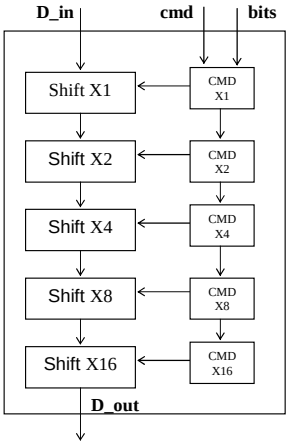


Figure 3-14 : Architecture logarithmique à 5 étages

Après synthèse, l'unité de décalage compte 2736 portes (4070 portes AN2LL) standard à deux entrées, la taille de l'unité de décalage est ainsi de 41112µm^2. Le benchmark utilisé pour la simulation consiste de 384 instructions qui réalisent les huit opérations possibles dans l'unité, avec des entrées générées aléatoirement. L'unité effectue 197559 transitions pour le benchmark donné. Dans cette architecture, l'acquittement des entrées se fait par digit. Ce travail exposera dans les sections 2.1 et 2.3, la différence et les avantages en termes de consommation entre un acquittement par digit et un acquittement par bus.

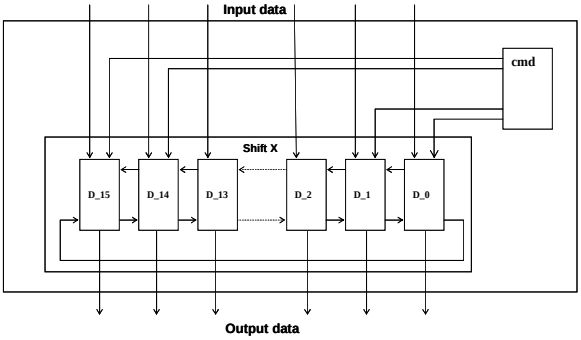


Figure 3-15 : Diagramme de bloc des blocs de décalage

En suivant le flot de conception proposé, la projection technologique utilisant les cellules de la librairie TAL et de la librairie de ST microelectronics en 130nm, le circuit se réduit à 1546 portes (3121 portes AN2LL) avec une taille de 31530µm^2. La simulation donne une activité de 129208 transitions pour les 384 instructions exécutées. Pour obtenir la consommation en joules par instruction, le circuit a été placé et routé. Dans cette étape, toutes les cellules sont choisies avec la sortance nécessaire. L'insertion de cellules de buffer est aussi réalisée pendant cette étape, le circuit compte alors 1754 portes (3381 portes AN2LL) avec une taille de 34156,56µm^2. En utilisant l'outil SOC Encounter, les capacités à la sortie de chaque porte peuvent ainsi être obtenues. Il est possible alors d'effectuer une nouvelle simulation avec l'outil Modelsim, qui avec le plugin pour compter les transitions, permet de pondérer celles-ci pour obtenir la capacité commutée totale du circuit après la simulation. Pour ce circuit, la capacité commutée étant de 2947pf. En utilisant l'Équation 3-1, l'énergie consommée par le circuit est de 4,2µJ. Le circuit consomme alors en moyenne 11pJ/instructions à une vitesse moyenne de 60MIPS

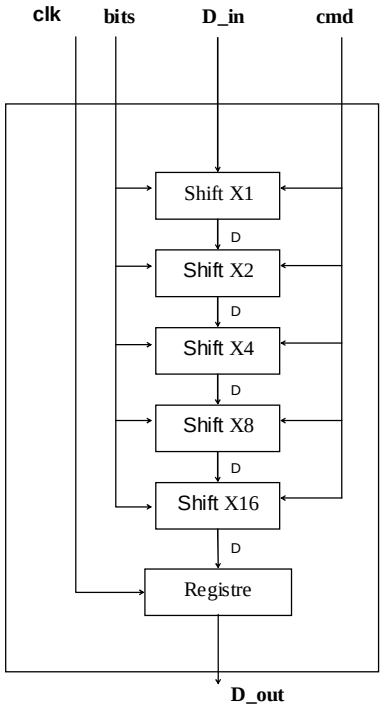


Figure 3-16 : Diagramme de blocs de l'unité de décalage synchrone

La version synchrone de l'unité de décalage est décrite avec le langage VHDL. La Figure 3-16 montre le diagramme de blocs de la version synchrone de l'unité de décalage. En synthétisant le circuit et après l'étape de projection technologique avec la librairie de ST Microelectronics en 130nm, le circuit comporte 303 (533 portes AN2LL) portes et a une taille de 5387µm^2. Une première simulation est réalisée en utilisant le même benchmark que pour la version asynchrone. Cette simulation donne

une activité en nombre de transition de 83241. Avec l'étape de placement et routage, la taille du circuit est de 5513µm^2 (545 portes AN2LL). En simulant le fichier de description en portes, le circuit a une capacité commutée totale de 4077,76pF. La consommation en joules du circuit synchrone de l'unité de décalage est alors de 15,29pJ/inst. Avec l'outil Design vision en utilisant les conditions pire cas (worst) le chemin critique est de 5,64ns qui correspond à une vitesse maximale de fonctionnement de 177,3 MHz, l'annexe C montre le rapport de temps généré par l'outil.

1.4.2.2.1.2 Structure à 3 blocs

Une deuxième architecture logarithmique a été conçue pour essayer de diminuer la consommation de l'unité de décalage. Cette deuxième structure réunit les blocs de décalage X2 et X4 en un même bloc ainsi que les bloc X8 et X16 pour obtenir un seul bloc. La Figure 3-17 montre le diagramme de blocs de l'architecture proposée. Cette architecture devrait optimiser le flux des données dans l'unité afin d'obtenir une réduction de l'activité en terme de transitions.

Étant donné que le circuit asynchrone est conçu avec des données codées en quatre rails, il a été décidé de laisser le bloc X1 comme celui dans la version à 5 étages. En effet, puisque le codage en quatre rails donne des digits de 2 bits, fusionner ce bloc avec le bloc X2 serait compliqué. Le bloc résultant doit traiter des digits en 2 bits pour faire un décalage de 1 bit, de 2 bits ou de 3 bits. En conséquence, les canaux de communication entre les sous blocs de l'étage fusionné, doivent assurer une communication de 2 bits ainsi qu'une communication de 1 bit. Ceci rendrait le circuit final plus compliqué alors qu'une fusion de blocs de décalages pairs permet une gestion plus optimisée des communications dans l'étage donné. La structure interne des étages de décalage est semblable à celles des étages montrés dans la version à 5 étages dans la Figure 3-15. Or, pour les blocs de cette nouvelle architecture, il existe plus d'un canal de communication qui relie les blocs. En effet, les structures fusionnées peuvent réaliser trois types de décalage pour chaque instruction, de 2, 4 ou 6 bits pour la structure X2-4 et 8, 16 et 24 pour la structure X8-16. Comme pour la version à 5 étages, un seul Half-Buffer est utilisé a la sortie du bloc X8-16.

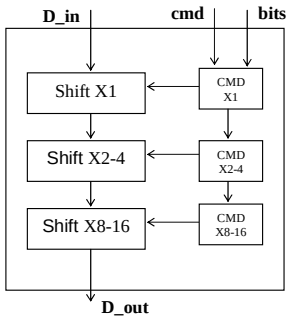


Figure 3-17 : Diagramme de blocs de l'architecture à 3 étages

De même que pour la version à 5 étages, cette architecture utilise un acquittement par digit. Le circuit synthétisé en portes standard à deux entrées comporte 3216 portes (4796 portes AN2LL) avec une taille de 48446µm^2. Avec la projection technologique qui utilise la librairie TAL et la librairie de STMicrolelectronics, le circuit est réduit à 2049 portes (3907 portes AN2LL) et a une taille de 39463,31µm^2. L'activité en transitions après simulation est de 170768 transitions. En faisant la projection technologique, les transitions avec le même benchmark sont réduites à 133036 transitions. Le placement routage de ce circuit permet d'obtenir les capacités réelles à la sortie des portes. Le circuit obtenu lors de l'étape de placement routage compte 1881 portes (3691 portes AN2LL) avec une taille de 37284µm^2. Après simulation en prenant en compte ces capacités, le circuit a une capacité commutée totale de 2,76nf qui permet d'obtenir une moyenne de 7,2pj/inst pour la consommation d'énergie du circuit a une vitesse moyenne de 45MIPS.

1.4.2.2.1.3 Simulation électrique

Pour vérifier l'estimation obtenue avec la méthodologie proposée, le circuit asynchrone à trois étages a été simulé avec l'outil NanoSim. Cet outil permet de réaliser une simulation électrique avec un fichier de description en transistor. Ce fichier de description en transistor a été généré avec l'outil Cadence après le placement routage réalisé avec SocEncounter. La simulation donne une consommation de 3,9nJ qui représente une consommation de 10,7pJ/instruction. Cette consommation est supérieure à la consommation estimée qui était de 7,2pJ/instruction. La Figure 3-18 montre un profil de courant généré par l'outil Nanosim. Ce profil montre l'exécution de deux instructions.

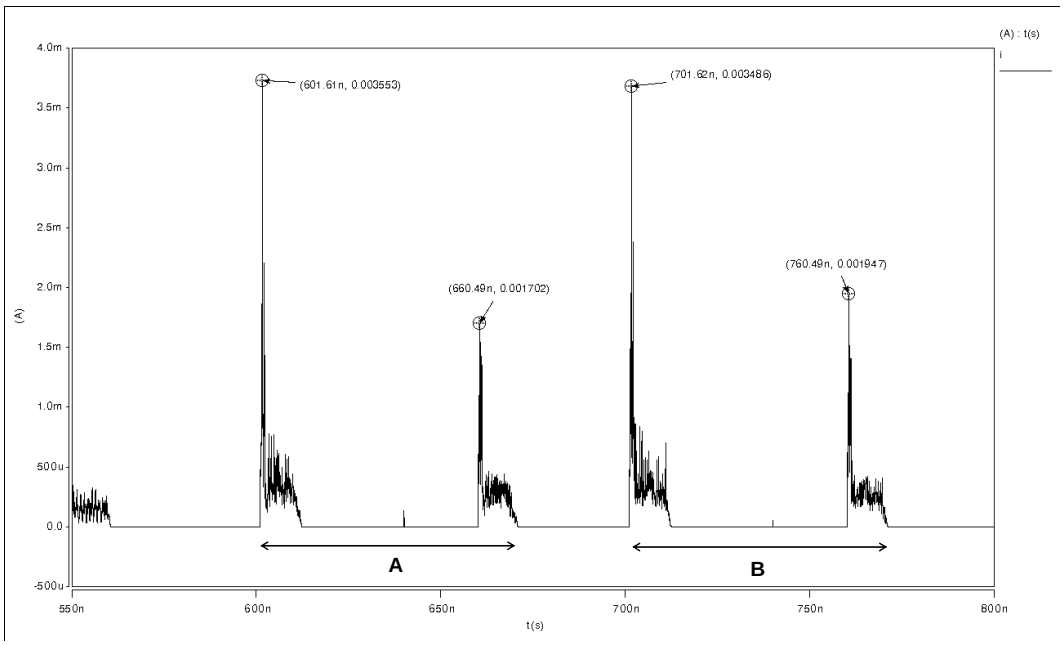


Figure 3-18 : Profil de consommation de deux instructions

La consommation de la simulation électrique est supérieure à la celle de la consommation estimée par l'outil proposé dans ce travail. Ceci est dû au fait que cette nouvelle simulation introduit un nouveau paramètre qui n'est pas pris en compte dans l'outil d'estimation d'énergie. Ce paramètre est la consommation réelle des transistors que l'outil Nanosim utilise puisque cet outil calcule la consommation des transistors et non des portes logiques comme l'effectue l'outil utilisé pour l'estimation d'énergie proposé dans ce travail.

Néanmoins, l'outil proposé dans le flot de conception permet d'avoir une estimation de la consommation du circuit analysé. Avec le profil de transitions qui est généré, le concepteur du circuit a une vision globale de l'activité du circuit dans le temps. Ceci lui permet d'optimiser les parties les plus consommantes très tôt dans le flot de conception.

1.4.2.2.1.4 Pic de courant en fonction des entrées

Il est important de voir sur le profil de courant montré dans la Figure 3-18, qu'il existe un pic de courant lors du début du calcul ainsi que lors de la phase de remise a zéro. Dans ce profil, ces deux pics de courant représentent dans un premier temps, l'arrivé des données et donc le commencement du calcul. Le deuxième pic est l'arrivée des données invalides qui commencent la mise à zéro des sorties du bloc. En pratique les entrées arrivent au fur et à mesure que le bloc précédent a généré chaque digit du bus d'entré. Dans le cas d'un circuit synchrone, toutes les entrées arrivent en même temps puisque c'est le signal d'horloge qui impose la communication entre blocs.

Pour comprendre l'importance de l'arrivée des données dans un circuit asynchrone, la Figure 3-19 montre une instruction exécutée trois fois de suite avec les mêmes entrées, mais les entrées arrivent sur des instants différents. Dans la première instruction A, les entrées arrivent en même temps ainsi que les données invalides qui débutent la remise a zéro. Dans l'instruction B, le nombre de bits à décaler et la commande arrivent en premier une à une. Par la suite, les 16 digits de la donnée arrivent un par un du digit de poids le plus fort au digit de poids le plus faible. Le pic de courant diminue mais le calcul du résultat est plus long. Pour la remise à zéro de l'instruction B, les digits invalides de la donnée arrivent en premier. Les données invalides du nombre de bits ainsi que de la commande arrivent en dernier. Cette configuration permet une légère réduction du pic de courant. Finalement dans l'instruction C, les 16 digits de la donnée arrivent en premier, puis le nombre à décaler et la commande. Ceci incrémente le pic de courant initial par rapport à l'instruction A. La remise a zéro se réalise en envoyant les données invalides sur le nombre de bits à décaler et la commande, les données invalides sur l'opérande sont envoyées à la fin. Avec cette configuration, le pic de courant sur la remise à zéro est légèrement supérieur que pour le cas B, mais inférieur au cas A. Or le temps de

remise a zéro s'incrémente avec un courant moyen inférieur aux deux cas précédent, ceci fait diminuer la consommation d'énergie nécessaire à l'exécution de l'instruction. Le Tableau 3-3 montre le récapitulatif de l'énergie consommée par chaque instruction ainsi que le temps nécessaire pour l'exécution.

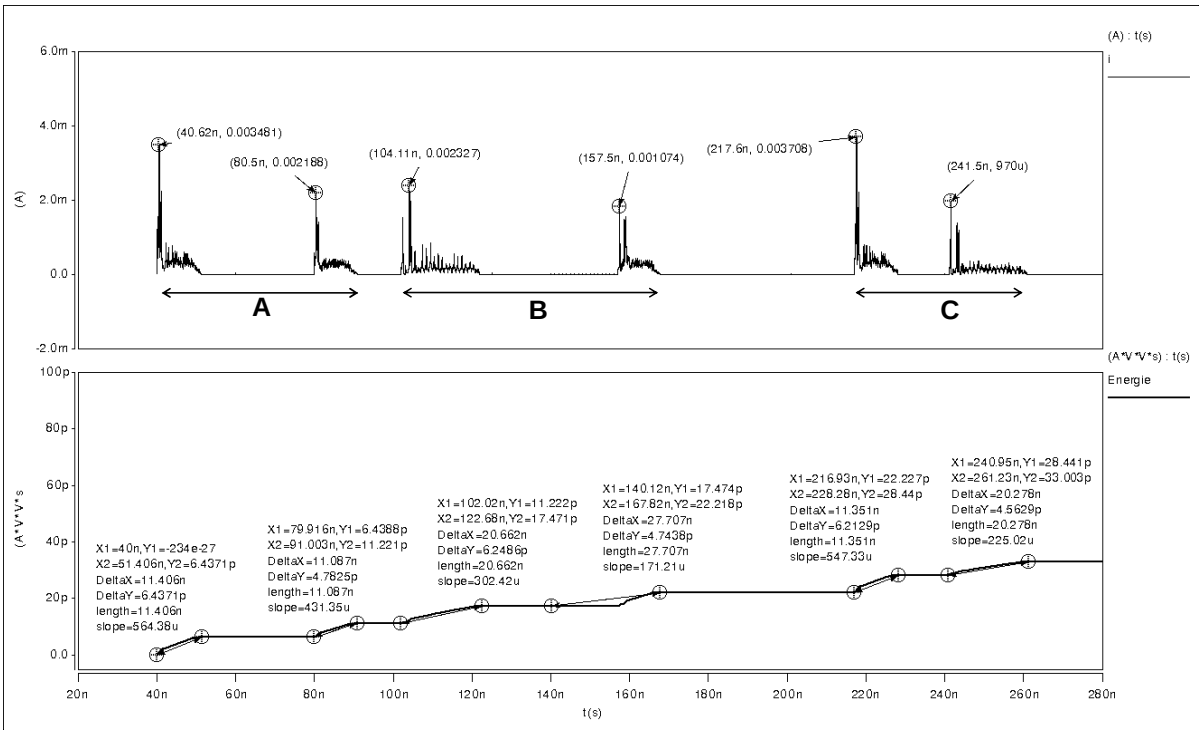


Figure 3-19 : Profil d'instruction avec entrées à temps différent.

instruction 1	calcul		remise à zéro		total	
	énergie (pJ)	temps (ns)	énergie (pJ)	temps (ns)	énergie (pJ)	temps (ns)
A	6.43	11.4	4.78	11.08	11.21	22.48
B	6.24	20.66	4.74	27.7	10.98	48.36
C	6.21	11.35	4.56	20.27	10.77	31.62

Tableau 3-3 : énergie et temps pour exécuter chaque instruction

La Figure 3-20 montre un autre exemple d'une instruction exécutée trois fois de suite avec des entrées qui arrivent à des instants différents. Dans l'instruction A, les entrées et les données invalides arrivent au même moment. Pour l'instruction B, la donnée qui va être décalée arrive en premier. La commande ainsi que le nombre de bits à décaler arrivent par la suite. Le pic de courant diminue de 28% par rapport au pic de courant dans l'instruction avec les entrées simultanées. Pour la remise à zéro, la commande et le nombre de bits présentent en premier l'état invalide. Par la suite, de façon aléatoire, chaque digit de la donnée se met à zéro. Le circuit calcule la remise à zéro en fonction des digits qui viennent de passer à zéro. Dans la Figure 3-20 il est possible de voir, pendant la remise a zéro de l'instruction B, que le circuit est inactif a certains instants. En effet il attend certains digits

pour continuer à générer l'état invalide de la sortie. Dans l'instruction C, le nombre de bit et la commande arrivent en premier pour le début de l'instruction ainsi que pour la remise à zéro. Pendant la remise à zéro, l'état invalide sur la donnée arrive du digit de poids le plus fort au digit de poids le plus faible. Le Tableau 3-4 montre le récapitulatif de l'énergie consommée par chaque instruction ainsi que le temps nécessaire pour l'exécution.

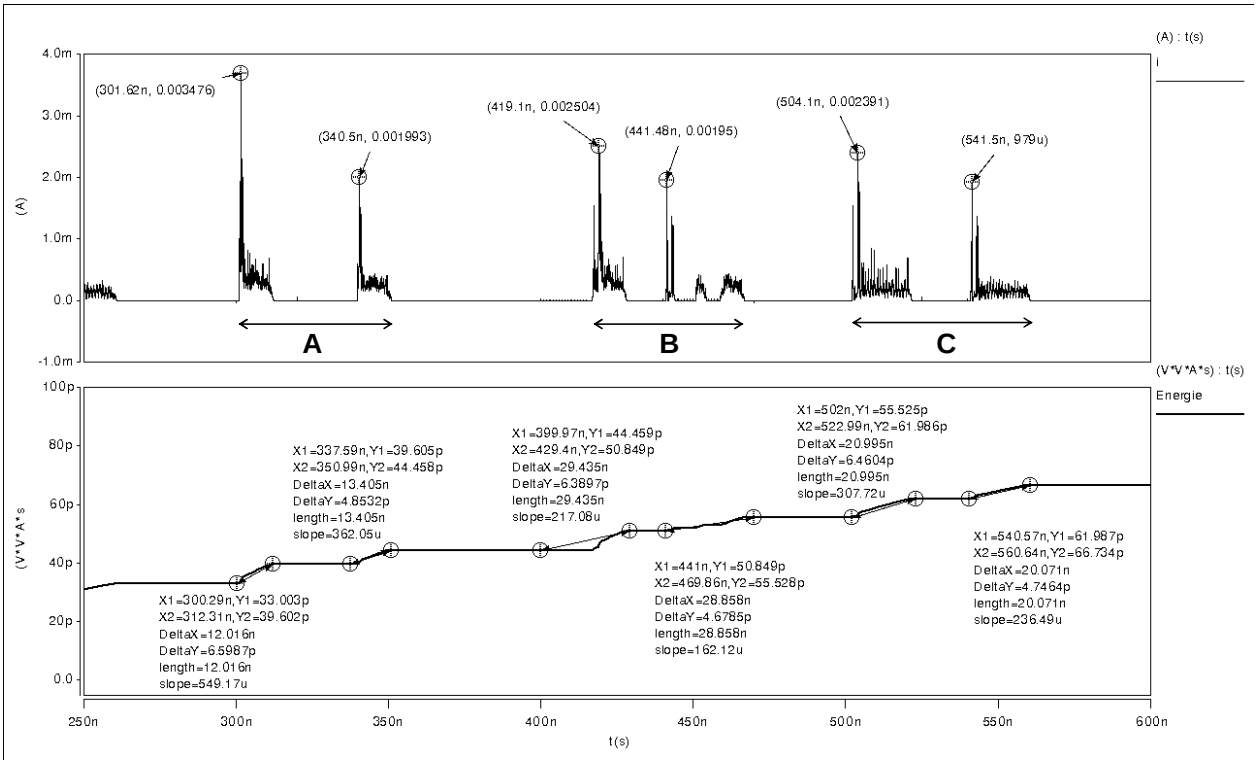


Figure 3-20 : Profil d'instruction avec entrées à temps différent.

instruction 2	calcul		remise à zéro		total	
	énergie (pJ)	temps (ns)	énergie (pJ)	temps (ns)	énergie (pJ)	temps (ns)
A	6.59	12.01	4.85	13.4	11.44	25.41
B	6.38	29.43	4.67	28.85	11.05	58.28
C	6.46	20.99	4.74	20.07	11.2	41.06

Tableau 3-4 : énergie et temps pour exécuter chaque instruction

Les deux exemples présenté montrent bien que la consommation d'énergie dépend des entrées ainsi que du temps d'arrivée. Il serait très difficile d'essayer d'optimiser le circuit pour prendre en compte ces paramètres puisque le temps de génération des sorties dans le circuit, dépend lui-même des entrées et de la fonction que le circuit doit réaliser. Néanmoins, l'énergie consommée reste assez constante dans les deux cas. Ceci est un des gros avantages des circuits asynchrones en ce qui concerne la sécurité du circuit.

1.4.2.2.2 Architecture avec machine à état.

La deuxième architecture choisie est une architecture avec une machine à états. Avec ce type d'architecture, il est possible de reboucler sur un même étage et ainsi avoir un seul étage de décalage. Ceci devrait réduire la taille du circuit puisqu'il n'utiliserait plus qu'un seul étage. Or, les commandes nécessaires aux blocs supplémentaires pour assurer le fonctionnement correct peuvent rendre ce circuit plus compliqué. Cette architecture ne contenant qu'un seul bloc pour décaler donnera des performances en termes de consommation d'énergie ainsi qu'en vitesse qui seront fonction du nombre de bits à décaler.

Ce travail présente, par la suite, une architecture qui utilise une machine à états. Il montrera le circuit proposé et les performances obtenues avec le benchmark utilisé pour les versions antérieures. Il montrera aussi l'importance des vecteurs d'entrées pour ce type d'architecture. Enfin, l'équivalent synchrone sera montré et une comparaison des deux types de circuits sera exposée.

1.4.2.2.2.1 Architecture proposée

Avec ce type de structure, reboucler sur l'étage de décalage à 1 bit peut prendre un temps considérable. Car si l'instruction à exécuter doit faire un décalage de 31 bits, la donnée qui circule dans la boucle devra passer par l'étage de décalage trente et une fois. Ceci imposera non seulement une augmentation du temps d'exécution de l'instruction, mais aussi de la consommation de cette dernière. La consommation dans une telle architecture dépend du nombre de fois que la donnée doit être traitée. Ceci est valable pour l'étage de décalage mais aussi pour tous les multiplexeurs et démultiplexeurs que la donnée devra utiliser.

L'architecture proposée dans la Figure 3-21, utilise deux étages de décalage, ceci pour accélérer l'exécution des instructions qui ont besoin de réaliser des décalages de plus de 3 bits. Cette architecture utilise alors un étage de décalage de 1 bit et un étage de décalage de 4 bits. Une machine à état contrôle le passage de la donnée à décaler lorsqu'elle traverse les étages de décalage nécessaires à son exécution. Il est important de noter que chaque bloc dans cette architecture possède un Half-Buffer en sortie, avec l'exception des blocs de calcul X1 et X4.

La Figure 3-22 montre l'architecture de la machine à états utilisée par le circuit. Les blocs CS (Current State, Etat Courant), NS (Next State, Etat suivant) et HSB (Holding State Buffer, Buffer de Maintient d'Etat) sont des half buffers qui assurent le passage de l'état suivant au bloc de calcul des sorties. Le bloc HSB permet de réaliser le protocole quatre phases. Il maintient l'état invalide sur l'entrée du bloc CS pendant que les sorties générées par le bloc de calcul sont envoyées à chaque bloc du circuit. Lorsque le circuit envoie l'acquittement de retour, le bloc HSB passe l'état suivant au bloc CS. Le bloc de calcul peut ainsi régénérer de nouvelles sorties.

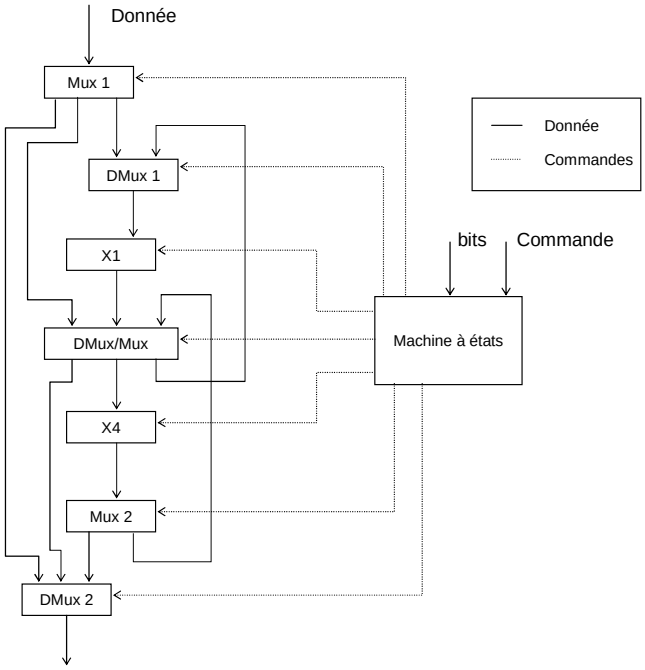


Figure 3-21 : Diagramme de blocs de l'unité de décalage avec machine à états

L'entrée ACK dans la machine à états est le signal d'acquittement du circuit. Dans cette architecture, le signal d'acquittement provient d'un bloc différent en fonction de l'état actuel. En effet, la machine à états génère les commandes nécessaires au fonctionnement du circuit. À chaque état, un ou deux blocs du circuit sont actifs. Lorsqu'ils ont généré la sortie, il est nécessaire d'acquitter la machine à états pour calculer l'état suivant ainsi que les sorties correspondantes à l'état courant. Dans la structure proposée, il existe un seul signal d'acquittement. Il faut alors regrouper tous les signaux d'acquittement qui proviennent du circuit pour en obtenir un seul. Ceci est fait avec des portes "et". Ce type de regroupement est possible car les blocs qui génèrent un acquittement sont mutuellement exclusifs entre eux. La machine à états assure cette exclusivité puisque la commande qui pourrait activer un deuxième bloc ne peut se produire qu'après avoir reçu l'acquittement du bloc précédent.

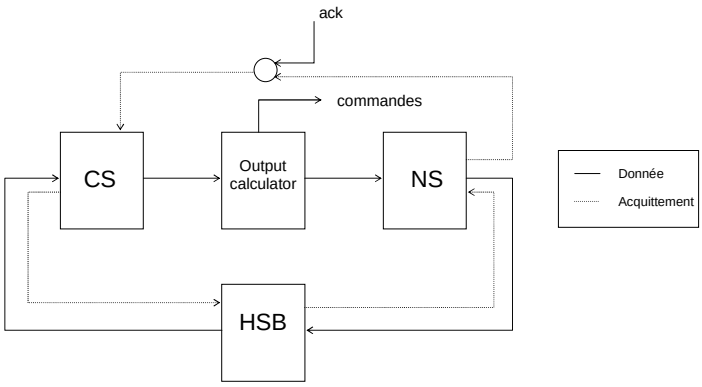


Figure 3-22 : Machine à états

1.4.2.2.2.2 Synthèse

Le circuit a été synthétisé en suivant le flot de conception proposé. La première version avec des portes standard à deux entrées a une taille de 44822µm^2 avec 2934 portes (4437 portes AN2LL). Cette structure est plus petite que l'architecture logarithmique à trois étages de décalage mais est légèrement plus grande que l'architecture à cinq étages.

Avec l'étape de projection technologique, en utilisant les mêmes librairies que pour les architectures antérieures, le circuit à une taille de 42240 µm^2 avec 2650 portes (4182 portes AN2LL).

1.4.2.2.2.3 Consommation

Dans un premier temps et pour comparer cette architecture avec les architectures précédentes, le circuit a été simulé avec le même benchmark utilisé pour les autres architectures. L'activité du circuit en nombre de transitions avant l'étape de projection technologique est de 1893259 transitions. Avec la projection technologique, l'activité est de 1815869 transitions. L'activité de ce circuit est supérieure à l'activité des circuits précédents. Ceci est dû à l'augmentation du nombre de blocs internes qui doivent assurer le protocole de communication quatre phases avec les commandes qui proviennent de la machine à états. En d'autres termes, cette version contient un nombre supérieur de half buffers qui entraînent une augmentation de l'activité du circuit.

En effectuant le placement routage du circuit, le nombre de portes augmente à 2902 (4734 portes AN2LL)et a une taille de 47822 µm^2. L'augmentation est due à l'insertion des buffers nécessaires. La simulation effectue 2125853 transitions. En utilisant les capacités de charge réelles du circuit lors de la simulation, le circuit a une capacité commutée totale de 38241,88pF qui correspond a une consommation en énergie de 55,07µJ, soit 143,4pJ/instruction. Le Tableau 3-5 montre le pourcentage de la capacité commuté totale pour chaque élément du circuit. Il expose clairement que le bloc qui calcule les états ainsi que les commandes propres au fonctionnement du circuit, est le bloc le plus consommant. Ce bloc représente plus du 50% de la consommation totale du bloc logique.

Bloc	Capacité commuté	%
mux1_inst	616.76pF	1.61%
dmux1_inst	1338.23pF	3.50%
shift_x1_inst	1349.77pF	3.53%
dm1_inst	4430.80pF	11.59%
shift_x4_inst	3389.99pF	8.86%
mux2_inst	2762.24pF	7.22%
dm2_inst	537.61pF	1.41%
state_machine_inst	22318.94pF	58.36%

Tableau 3-5 : Capacité commuté par bloc

La consommation du circuit avec cette architecture de machine à états, dépend du nombre de bits à décaler. Une simulation exhaustive pour calculer la consommation en énergie par bit à décaler a été réalisée. Le benchmark utilisé permet de décaler de 0 à 31 bits avec des entrées aléatoires. La Figure 3-23 montre la consommation de cette architecture en fonction du nombre de bits à décaler. La vitesse de fonctionnement de l'unité de décalage avec la structure à machine à états dépend également du nombre de bits a décaler. La Figure 3-24 montre la vitesse de calcul en fonction du nombre de bits à décaler.

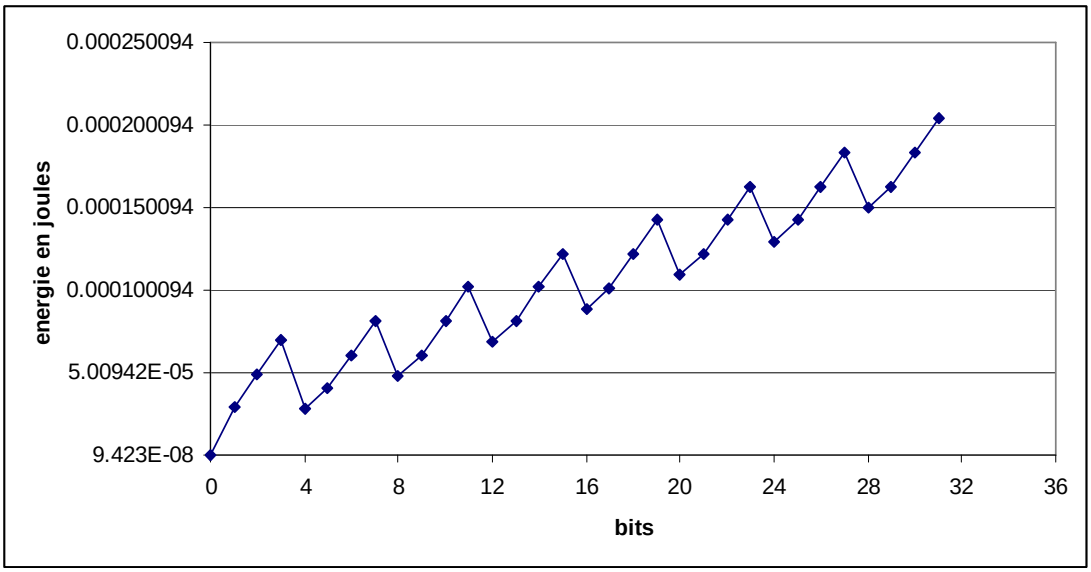


Figure 3-23 : Consommation par nombre de bits a décaler.

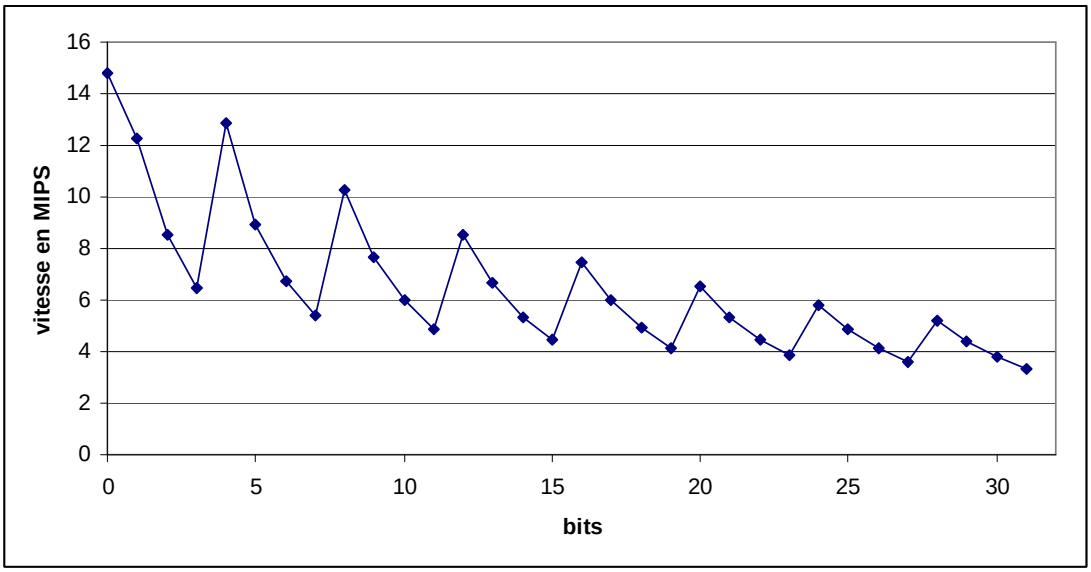


Figure 3-24 : vitesse de calcul en fonction des bits à décaler

1.4.2.2.3 Conclusions de l'unité de décalage

Le Tableau 3-6 présente un récapitulatif des différentes architectures de l'unité de décalage. Le regroupement des fonctions dans un circuit permet de réduire la consommation d'énergie. Dans l'architecture proposée, la consommation de l'unité de décalage a été réduite de 34,5%. Or la taille du circuit dans la version à trois étages a augmenté de 13,4%. La consommation statique qui n'est pas prise en compte augmentera en conséquence. Le temps de calcul dans la version à trois étages diminue puisque les données traversent moins de portes logiques.

L'équivalent synchrone de l'unité de décalage consomme 28% plus d'énergie que le circuit asynchrone. Il est néanmoins 83% plus petit que le circuit asynchrone. Dans ce cas comme pour l'unité logique, la taille des circuits est difficilement comparable puisque les buffers qui sont insérés à cause de l'arbre d'horloge, ne sont pas représentatifs pour un circuit de cette taille.

L'utilisation d'une machine à états dans un circuit asynchrone est très pénalisant. La structure choisie dans cette étude présente plusieurs inconvénients en terme de vitesse et d'énergie. Il existe néanmoins d'autres structures pour réaliser une machine à état dans un circuit asynchrone. Ces structures sont étudiées actuellement dans les travaux de thèse de Khaled Alsayeg.

Pour obtenir le circuit le moins consommant il est nécessaire de regrouper au maximum les blocs de calcul dans le circuit. Minimiser l'insertion de half-buffer est crucial pour réduire la consommation. Dans les deux architectures logarithmiques, les half-buffers se trouvent à la sortie du circuit tandis que dans l'architecture avec la machine à états, chaque bloc interne de l'architecture possède un half-buffer.

circuit	portes AN2LL	transitions	énergie/instruction en pJ
Shift 5 étages	4070	197559	
5 étages après Techmap	3121	129208	
5 étages après Placement routage	3381		11
Shift 3 étages	4796	170768	
3 étages après Techmap	3691	133036	
3 étages après Placement routage	3907		7.2
Shift avec machine à états	4437	1893259	
machine à états après Techmap	4182	1815869	
machine à états après Placement routage	4734		143
Shift synchrone	533	83241	15.29

Tableau 3-6 : Récapitulatif de l'unité de décalage

1.4.2.3 Comparaison de la consommation en fonction des entrées

L'étude des architectures du bloc de décalage et du bloc logique a permis de comparer la consommation entre les circuits asynchrones et leurs équivalents synchrones. Or cette consommation a été obtenue en réalisant une simulation avec des vecteurs d'entrée aléatoires qui donnent en conséquence une consommation moyenne des circuits proposés.

Dans un circuit asynchrone, la consommation d'énergie est quasiment indépendante des entrées. Ceci n'est pas le cas dans les circuits synchrones. En effet, dans les circuits synchrones la consommation varie en fonction des vecteurs d'entrée, et plus précisément de la quantité de bits qui changent entre deux données. Le nombre de bits changent entre données est couramment connue comme la distance de Hamming [HAM 50]. Pour montrer l'importance de la variation de la consommation dans les circuits synchrones et asynchrones, une étude exhaustive a été réalisée. Ceci dans le but de mesurer la consommation d'un circuit en fonction de la distance de Hamming.

L'unité de décalage à 5 étages après placement routage pour les versions synchrone et asynchrone est simulée en faisant varier la distance de Hamming. La Figure 3-25 et la Figure 3-26 montrent le nombre de transitions obtenu pour le circuit asynchrone et le circuit synchrone respectivement lorsque la distance de Hamming varie entre 0 et 32.

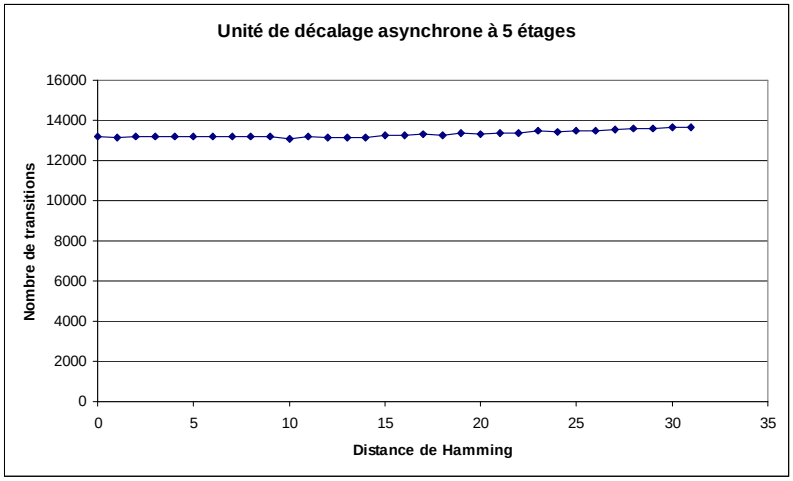


Figure 3-25 : Transitions en fonction de la distance de Hamming pour le circuit asynchrone

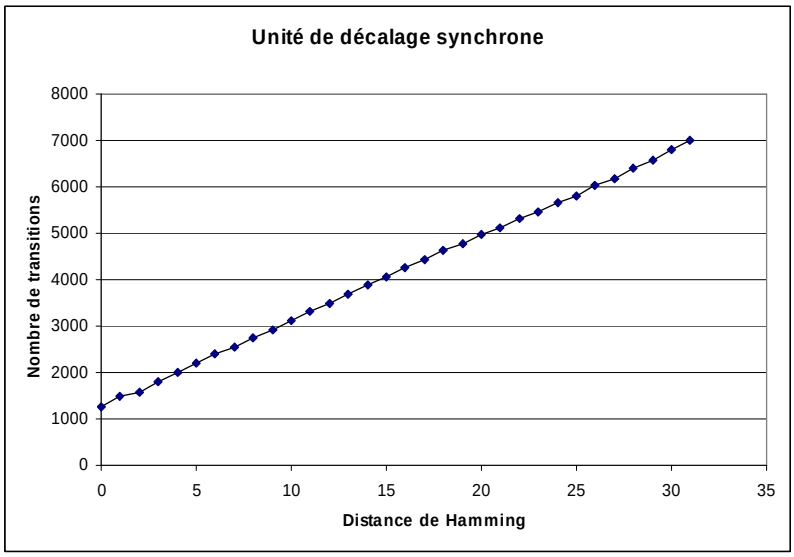
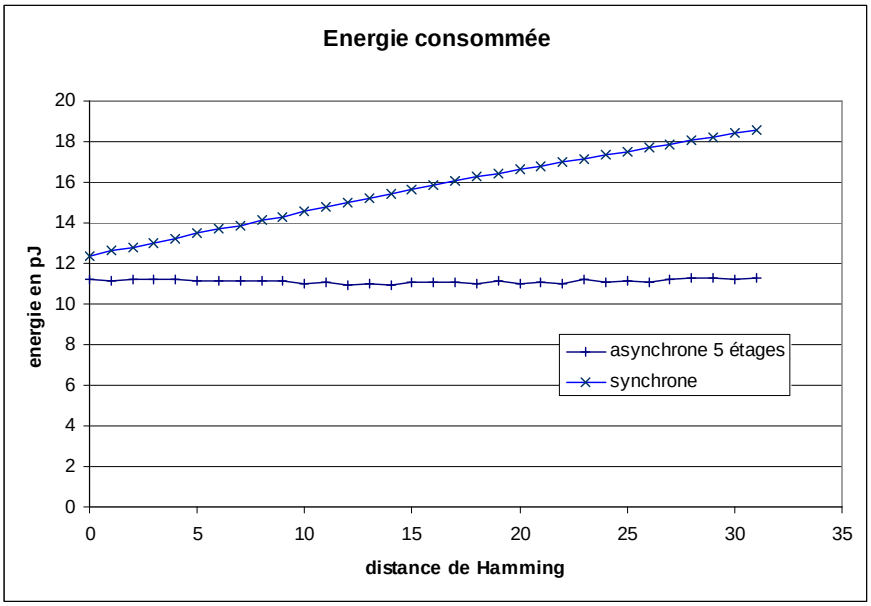


Figure 3-26 : Transitions en fonction de la distance de Hamming pour le circuit synchrone

La Figure 3-27 présente l'énergie totale consommée par les deux circuits synchrone et asynchrone en fonction de la distance de Hamming. Cette figure permet d'observer que la consommation du circuit asynchrone est pratiquement constante. La variation en consommation en fonction de la distance de Hamming est presque nulle. Contrairement au circuit asynchrone, la consommation de l'équivalent synchrone du circuit varie en fonction de la distance de Hamming. Cette variation est non négligeable puisqu'elle peut augmenter jusqu'à 33% par rapport à la consommation minimale.

Un autre avantage des circuits asynchrones est montré dans la Figure 3-28. Le pic de transitions par calcul est dans le meilleur des cas deux fois plus important dans le circuit synchrone, mais ce pic peut être jusqu'à trois fois plus important. Ceci représente des pics de courant qui pénalisent fortement la consommation dans un circuit synchrone. Il est important de noter que le pic de transitions présenté dans cette figure, est le pic de transitions dans le pire des cas pour le circuit asynchrone. Les données arrivent en même temps.



g

Figure 3-27 : Energie consommée en fonction de la distance de Hamming

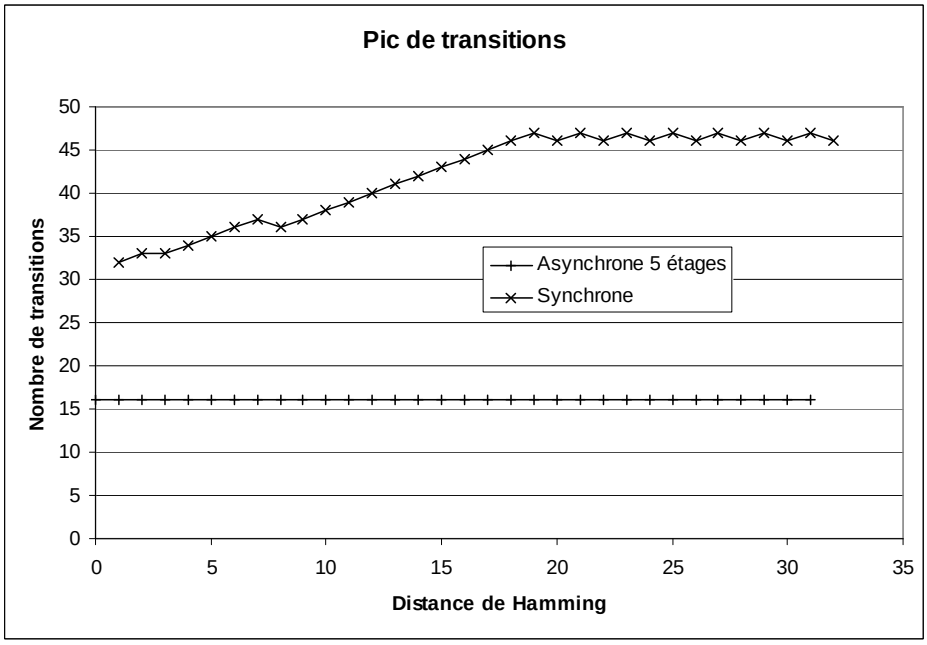


Figure 3-28 : Pic de transitions en fonction de la distance de Hamming

2 Analyse d'architectures pour réduire la consommation

Par la suite, ce travail exposera les avantages et les inconvénients de certains choix d'architectures qui doivent être effectués lors de la conception d'un circuit asynchrone. Il analysera, dans un premier temps, la génération du signal d'acquittement nécessaire à la communication entre

blocs d'un circuit asynchrone. Dans un deuxième temps, il montrera comment il est possible d'éliminer certains signaux de commandes en utilisant des entrées et des sorties actives ou passives. Finalement, une analyse de la composition des blocs de démultiplexeurs en terme de probabilité sera faite.

2.1 Génération du signal d'acquittement.

Le signal d'acquittement dans un circuit asynchrone est la base de la communication entre blocs du circuit. Ce signal permet dans ce type de circuit, avec un protocole de communication quatre phases, d'informer le bloc précédent de la réception des données envoyées par celui-ci. Dans un bus de communication de n bits, le bloc émetteur doit attendre la réception de tous les bits qui composent la donnée. Le bloc récepteur reçoit chaque bit théoriquement en même temps. Cependant, dans un circuit asynchrone, la génération de chaque bit de la donnée envoyée par l'émetteur n'est jamais faite en même temps ; à la différence des circuits synchrones qui eux envoient une donnée à chaque front d'horloge. Dans un circuit asynchrone les bits de la donnée sont calculés en fonction du temps nécessaire à chaque bit et sont alors envoyés au fur et à mesure de leur génération.

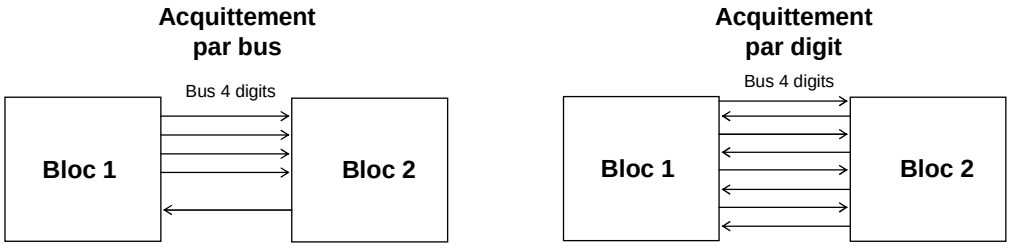


Figure 3-29 : Génération de l'acquittement

L'acquittement du bus de données peut être généré lorsque toutes les entrées ont été reçues avec un signal d'acquittement global. Mais il est aussi possible d'acquitter chaque bit indépendamment (ou chaque digit pour une communication quatre rails). La Figure 3-29 expose les deux types d'acquittements possibles dans un circuit asynchrone. D'après cet exemple, l'acquittement par bus paraît plus optimisé puisque le nombre de fils d'interconnexions entre blocs est inférieur au nombre de fils dans l'acquittement par digit. En effet, dans un circuit qui a des données en 32 bits, le signal d'acquittement augmente considérablement le nombre de fils d'interconnexions. Ceci entraîne également des problèmes de bruit dus à la communication puisque les transitions des fils augmentent en fonction du nombre de fils. Dans l'exemple montré, l'acquittement par bus génère, lors d'une communication, 10 transitions avec un protocole quatre phases, les bits étant codé en double rails ou en quatre rails. Le même exemple avec un acquittement par digit effectue 16 transitions. L'activité due à la communication est également augmentée. Une communication d'une donnée sur 32 bits nécessite

66 transitions sur les fils d'interconnexion avec l'acquittement par bus, et 128 pour un acquittement par digit.

2.2 Acquittement par bus

Cette première approche semble logique, or, elle ne tient pas compte de la génération en soit du signal d'acquittement. En effet, afin de générer le signal d'acquittement, le bloc récepteur doit avoir reçu tous les bits de la donnée pour pouvoir construire le signal d'acquittement. La Figure 3-30 montre la génération typique du signal d'acquittement dans un circuit asynchrone avec un protocole quatre phases en WCHB (Weak Condition Half Buffer). Le bloc 1 génère l'entrée du bloc 2. Par la suite, ce dernier calcule la sortie sur les n digits. L'acquittement est généré en regroupant les quatre rails de chaque digit avec des portes "OU". Ce regroupement est possible puisque les quatre rails sont mutuellement exclusifs. Les sorties de toutes ces portes sont ensuite regroupées par des portes de Muller. Le bloc "Réseau de portes de Muller" réalise cette tache et délivre le signal d'acquittement. Ce réseau assure que toutes les sorties du bloc ont été générées et qu'il est alors possible d'acquitter le bus d'entrée.

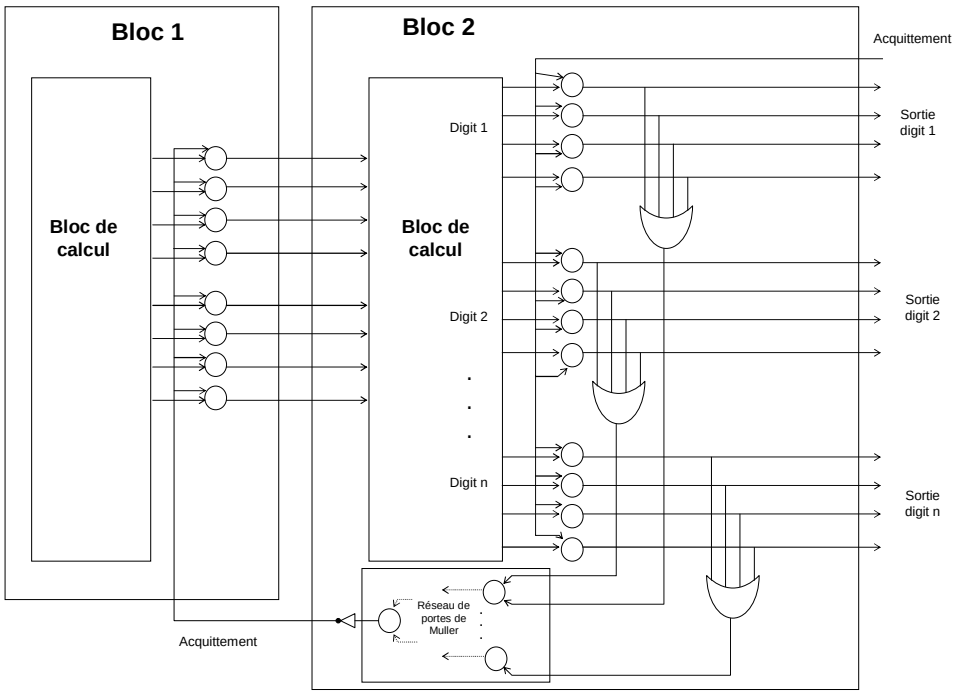


Figure 3-30 : Acquittement par bus

Deux nouveaux problèmes apparaissent avec cette architecture. Le premier est le réseau de portes de Muller en soit. Ce réseau incrémente la taille du bloc et en conséquence, le nombre de commutations dans les portes. Pour un circuit sur 32 bits, il est nécessaire d'utiliser 17 portes de

Muller à deux entrées et un inverseur puisque le signal d'acquittement est actif à l'état haut. Avec la librairie TAL, cette partie du circuit peut être réduite à 5 portes de Muller à quatre entrées et un inverseur. La consommation d'énergie est néanmoins non négligeable puisque ce circuit sera présent sur tous les blocs du circuit.

Le deuxième problème est que le signal d'acquittement généré avec cette méthode doit alimenter toutes les portes de Muller à la sortie du bloc qui vient d'envoyer la donnée. Pour un circuit à 32 bits, le signal devra charger 64 portes de Muller. Ceci est impossible avec l'inverseur qui se trouve à la sortie du réseau de portes de Muller. Il est alors nécessaire d'introduire des buffers pour assurer l'envoi de ce signal. Les buffers sont introduits au cours de l'étape de placement routage.

2.3 Acquittement par digit

Pour réduire la consommation de ce circuit, la génération de l'acquittement par digit est probablement plus conseillée. La Figure 3-31 montre le circuit équivalent avec un acquittement par digit. Avec cette nouvelle structure, le circuit a été diminué en taille et l'activité est ainsi réduite. Chaque signal d'acquittement doit alimenter quatre portes de Muller ce qui est possible avec une porte à faible sortance, ce qui réduit ainsi la capacité de charge dans le nœud d'acquittement. Aucun buffer n'est introduit dans l'étape de placement routage puisque la sortance des portes est suffisante.

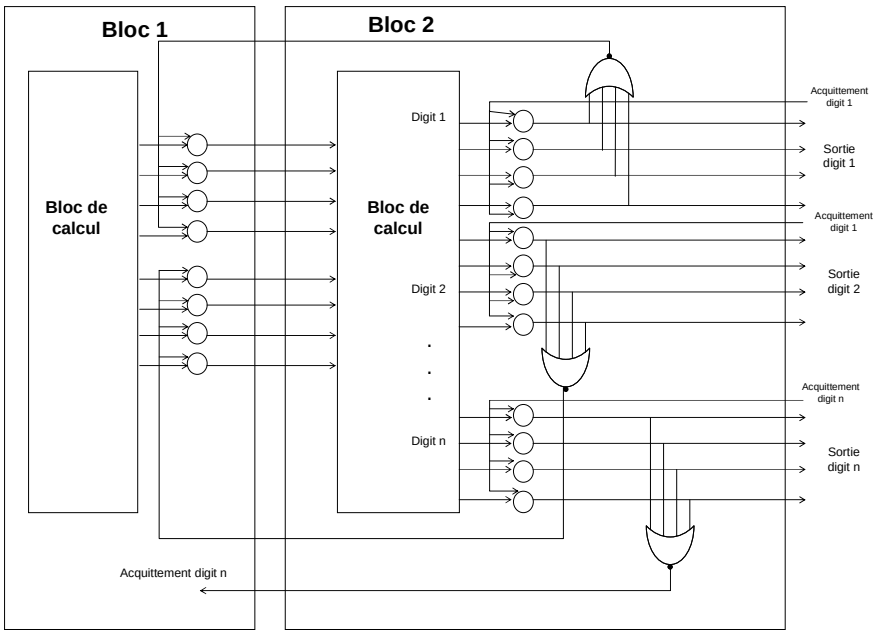


Figure 3-31 : Acquittement par digit

2.3.1 Unité logique

Pour comparer la consommation entre les deux architectures, l'acquittement de l'unité logique présentée dans la section 1.4.2.1 a été modifié pour permettre un acquittement par bus. Pour comparer la consommation des deux versions, il a fallu ajouter un circuit qui envoie les données. Les portes de Muller qui sont présentes à la sortie du bloc qui envoient les données ont été alors incluses dans le circuit à simuler. Ceci permet d'obtenir une vision plus réaliste de la consommation résultant de la commutation des portes, mais aussi celle provenant de l'insertion des buffers nécessaires pour assurer la propagation du signal d'acquittement. Le circuit compte alors 770 portes (1814 portes AN2LL) et une taille de 18330 μm^2 après l'étape de placement routage. L'activité est estimée à 19768 transitions avec une capacité commutée totale de 497,57pF ce qui donne une consommation de 716pj avec le benchmark utilisé précédemment dans l'unité logique. Ceci correspond alors à 11,2pj/instructions. Il faut noter que cette consommation inclut aussi la consommation de la sortie du bloc précédent.

La simulation de l'unité logique précédemment étudiée, avec un acquittement par digit, est alors faite en incluant aussi l'étape de sortie du bloc précédent. Cette nouvelle simulation donne une activité de 17247 transitions, la taille du circuit est alors de 18.135μm^2. La capacité commutée totale étant de 377,37pF, la consommation totale d'énergie est alors de 543,4pJ ce qui génère une consommation de 8,49pJ/instructions.

2.3.2 Conclusions

Le signal d'acquittement dans un circuit asynchrone, permet de synchroniser la communication de manière locale. Dans un bus de données, l'acquittement de cette donnée peut se faire de manière globale avec un signal qui acquitte tout le bus qui fournit la donnée. Or cet acquittement peut aussi se faire par digit de la donnée en soi.

L'étude présentée montre que l'acquittement par digit, dans l'unité logique, présente deux avantages. Le premier étant la consommation du circuit, qui pour l'exemple montré représente un gain de 24,2% par rapport a un acquittement par bus. Le deuxième avantage est que la taille du circuit avec un acquittement par digit est légèrement inférieure à la taille du circuit avec un acquittement par bus.

Ce résultat n'est pas généralisable à tous les circuits asynchrones. La génération du signal d'acquittement peut être plus complexe dans certains cas et un acquittement par bus serait plus intéressant en consommation et surface. Cette section montre qu'en fonction du type d'acquittement, le circuit aura une consommation plus ou moins élevée et qu'il est nécessaire de prendre en compte ce choix lors de la conception de circuits asynchrones.

2.4 Suppression des signaux de commande.

2.4.1 Présentation du problème

Lorsque les blocs d'un circuit asynchrone sont conçus, il est habituellement nécessaire d'avoir deux types de signaux d'entrée : les entrées dites opérande et les entrées de contrôle ou de commande. Les premières sont les données avec lesquelles il faudra réaliser certaines fonctions ou bien les diriger vers certaines sorties dans le cas de multiplexeurs ou démultiplexeurs. Le deuxième type d'entrée sont les commandes qui proviennent d'un bloc de contrôle pour indiquer au bloc en question quel type de fonction il faut effectuer. La Figure 3-32.a expose ce type de conception.

Dans le cas où le bloc "Fonction 2" de cet exemple représente un démultiplexeur, le bloc de contrôle indique, par l'intermédiaire de la commande, vers quelle sortie l'entrée devra être dirigée. En analysant le comportement d'un démultiplexeur, il est possible de voir que les blocs "Fonction 3" et "Fonction 4" sont exclusifs puisque le multiplexeur envoie la donnée à un seul des deux blocs. Dans ce type de conception, le bloc qui transmet des données possède des sorties qui sont actives et le récepteur est formé d'entrées passives. Une sortie active signifie que c'est elle qui commence le protocole quatre phases de la communication asynchrone. Elle passe de l'état invalide à un état valide pour envoyer la donnée au bloc suivant. Par la suite, elle attend le signal d'acquittement de la donnée envoyée pour remettre la sortie à un état invalide. Le récepteur est toujours en train d'attendre l'entrée pour qu'à son tour il génère les signaux de sortie et d'acquittement.

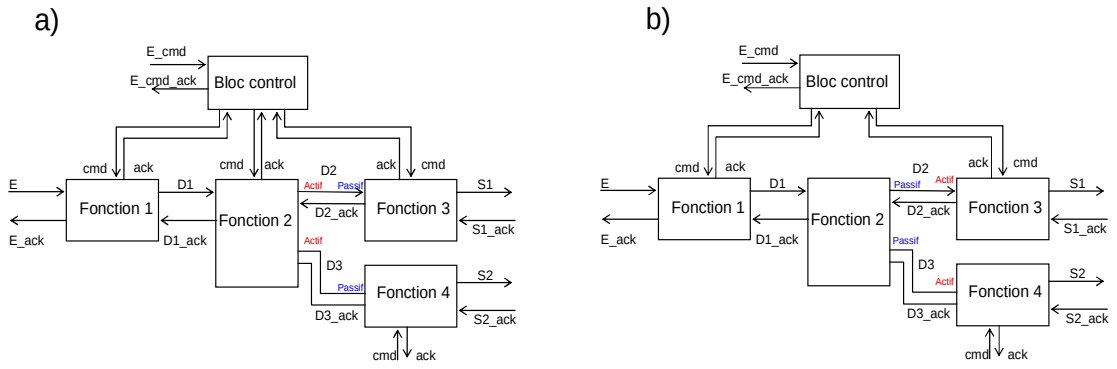


Figure 3-32 : Exemple d'une structure asynchrone

Pour optimiser en taille ainsi qu'en consommation le circuit présenté dans la Figure 3-32.a, une simple modification peut être réalisée. Il suffit d'inverser la nature de la sortie du bloc 2 ainsi que celles des entrées des blocs 3 et 4 comme le montre la Figure 3-32.b. Cette modification permet de supprimer la commande du bloc 2 ce qui implique une réduction en complexité du bloc de contrôle puisque grâce à cette modification, ce bloc n'aura plus à calculer et à générer cette commande. Dans

cet exemple, les commandes générées aux blocs 3 ou 4, restent les mêmes. Ce sera à ces deux blocs de réaliser la demande des données d'entrées lorsqu'ils en auront besoin. Ceci est réalisé par l'intermédiaire du signal d'acquittement qui va alors initier le protocole de communication quatre phases.

2.4.2 Démultiplexeur

La Figure 3-33 montre le diagramme en porte d'un démultiplexeur 1 bit double rails avec et sans commande. Dans le circuit sans commande, la taille est pratiquement réduite d'un facteur 2 ainsi que la consommation du circuit.

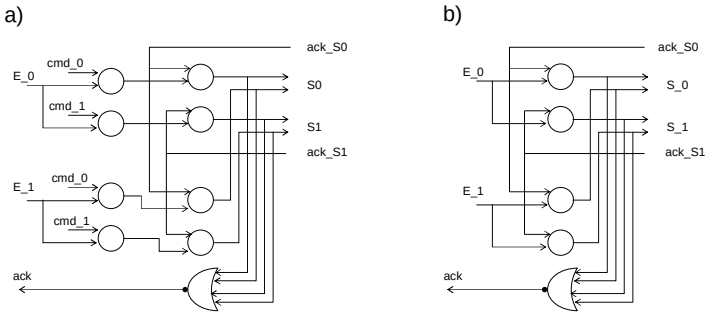


Figure 3-33 : Demultiplexeur avec et sans commande

2.4.3 Conclusion

Supprimer les signaux de commande dans les blocs, permet de réduire la taille du circuit, puisque le bloc reçoit moins d'entrées, ce qui se traduit par une diminution de portes de Muller dans les circuits asynchrones. Parallèlement, la complexité du bloc qui génère la commande diminue puisqu'il doit générer un signal en moins. Ceci réduit en conséquence la consommation en énergie du circuit.

2.5 Structures équilibrées et non équilibrées

Le but de cette étude est de pouvoir comparer deux structures de démultiplexeur sans commande. La première étant un démultiplexeur classique équilibré (on ne privilégie aucune des sortie), la deuxième est un démultiplexeur déséquilibré dans lequel l'une des sorties est privilégiée. Ceci est intéressant lorsque l'une des sorties est utilisée plus fréquemment. La deuxième partie de l'étude consiste à comparer les structures en 2 et 4 rails.

Cette étude cherche à évaluer la consommation des démultiplexeurs ainsi que la taille en nombre de transistors que coûteraient de telles implémentations. Dans ce but, l'outil réalisé par Joao Fragoso au sein du groupe CIS du laboratoire TIMA sera utilisé. Avec cet outil, il est possible de calculer la consommation d'un circuit en fonction des probabilités sur les entrées.

2.5.1 Problématique

L'objectif est d'optimiser la consommation du processeur MIPS présenté dans la section 1.4.1 en améliorant l'architecture des parties les plus sollicitées. Pour cela, en étudiant l'architecture, il ressort que dans l'ALU les opérandes qui y sont envoyées passent toujours par des démultiplexeurs qui ensuite seront envoyés au sous bloc correspondant. La Figure 3-34 montre une version simplifiée de la structure interne de l'ALU. Elle présente simplement le chemin que suit l'opérande 1 pour arriver au bloc de calcul nécessaire. Les blocs de calcul demandent l'opérande au démultiplexeur lorsqu'ils en ont besoin. Le démultiplexeur n'a donc pas de commande. Ce démultiplexeur sans commande en double rails est illustré dans la Figure 3-35.

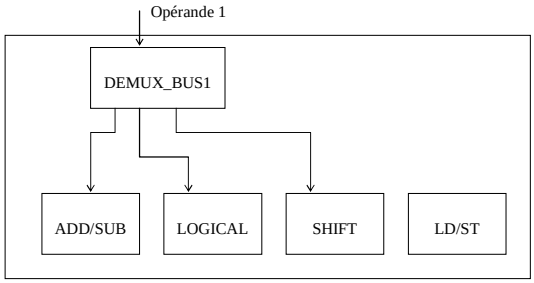


Figure 3-34 : Détail du chemin des opérandes dans l'alu

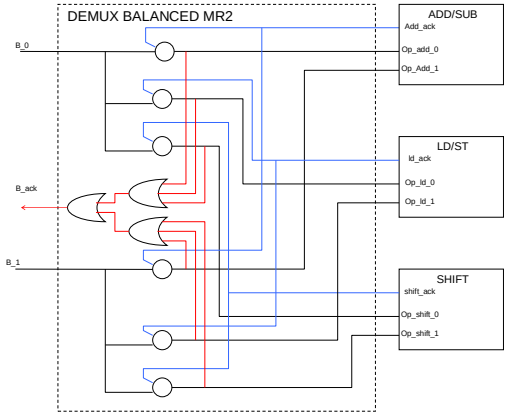


Figure 3-35 : DÉMULTIPLEXEUR équilibré double rails

[CME 91] montre l'utilisation des instructions en fonction du benchmark SPEC. Le Tableau 3-7 présente les résultats de cette étude. L'instruction NOP étant exécutée comme un SLL avec

destination le registre 0, elle est donc additionnée aux instructions de décalage. Les instructions CTL étant les instructions de contrôle (saut conditionné). Quelques considérations concernant l'utilisation des instructions dans le MIPS réalisé par l'équipe de travail doivent être prises en compte. Les instructions de LD/ST (Load/Store) utilisent le bloc ADD/SUB pour calculer l'adresse. L'instruction NOP ne sera pas implémentée dans ce MIPS de la même façon que dans le MIPS commercial. En effet, puisque cette instruction a pour but de ne rien faire, elle ne sera pas passée à l'ALU. Le décodeur du MIPS vérifie lorsqu'une instruction SLL se présente si le registre destination est le registre 0, si tel est le cas il annulera l'instruction puisque ceci correspond à une instruction NOP. Avec ces considérations, le Tableau 3-8 montre le nombre d'instructions par bloc.

	gcc1.35	espresso	li	eqntott
ld/st	220	521	1322	205
	21.2765957	18.6537773	23.3734088	16.6215318
Ctl	177.62	423.01	887	343.07
	17.1779497	15.1453634	15.6824611	27.8163361
nop	148	341	1316	181
	14.3133462	12.2090942	23.2673267	14.6755963
add/sub	301	799	1232	490
	27	28.27	20.45	39.44
logical	21.71	330	134	2.37
	2.09961315	11.8152524	2.36916549	0.19216112
shift	69.7	254	39.21	2.04
	6.74081238	9.09416398	0.69324611	0.16540451
shift + nop	217.7	595	1355.21	183.04
	21.0541586	21.3032581	23.9605728	14.8410009
Total instructions	1034	2793	5656	1233.34
in benchmark				

Tableau 3-7 : utilisation des instructions en fonction des benchmarks

nop not executed in ALU				
add/sub block	698.62	1743.01	3441	1038.07
	88.43%	74.90%	95.21%	99.58%
shift block	69.7	254	39.21	2.04
	8.82%	10.92%	1.08%	0.20%
logical block	21.71	330	134	2.37
	2.75%	14.18%	3.71%	0.23%
Total instructions	790.03	2327.01	3614.21	1042.48

Tableau 3-8 : nombre d'instructions par bloc

De cette étude il est possible de conclure que l'utilisation du bloc ADD/SUB dépasse les 75% d'utilisation. Pour certains benchmarks, l'utilisation de l'unité ADD/SUB peut même être de 99,58%.

Le reste concerne des instructions utilisant les blocs LOGICAL et SHIFT. De ce fait, il serait envisageable de réaliser un démultiplexeur qui privilégierait le chemin de données vers le bloc ADD/SUB.

La Figure 3-36 montre le démultiplexeur déséquilibré. Dans ce démultiplexeur, la donnée B (B_0 et B_1, puisqu'elle est codée en double rails) arrive sur 2 portes de Muller. Ceci réduit la charge d'entrée par rapport à celle du démultiplexeur équilibré puisque cette dernière est de 3 portes de Muller. Il faudra bien sûr prendre en compte le fait que lorsque les blocs LD/ST ou SHIFT sont utilisés, la consommation du circuit sera supérieure à celle du démultiplexeur équilibré.

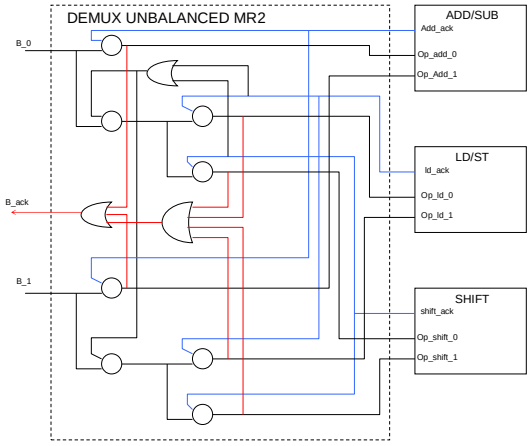


Figure 3-36 : Démultiplexeur déséquilibré en dual rail

Il est évident, d'après les schémas des deux circuits, que le démultiplexeur équilibré aura une consommation qui ne dépendra pas de la sortie utilisée. Au contraire, le démultiplexeur déséquilibré aura une consommation différente en fonction de la sortie utilisée. Comme il a été mentionné dans les paragraphes antérieurs, si la sortie ADD/SUB est utilisée, la consommation sera inférieure aux sorties LD/ST et SHIFT. Pour pouvoir comparer les deux démultiplexeur, il faudra estimer la consommation en fonction de la probabilité d'utilisation des sorties pour le démultiplexeur déséquilibré. Ceci est réalisé avec l'outil développé par Joao Fragoso pour obtenir l'estimation de la consommation d'énergie en fonction de la probabilité dans les entrées. L'outil utilisé rend une consommation équivalente en nombre d'inverseurs de base.

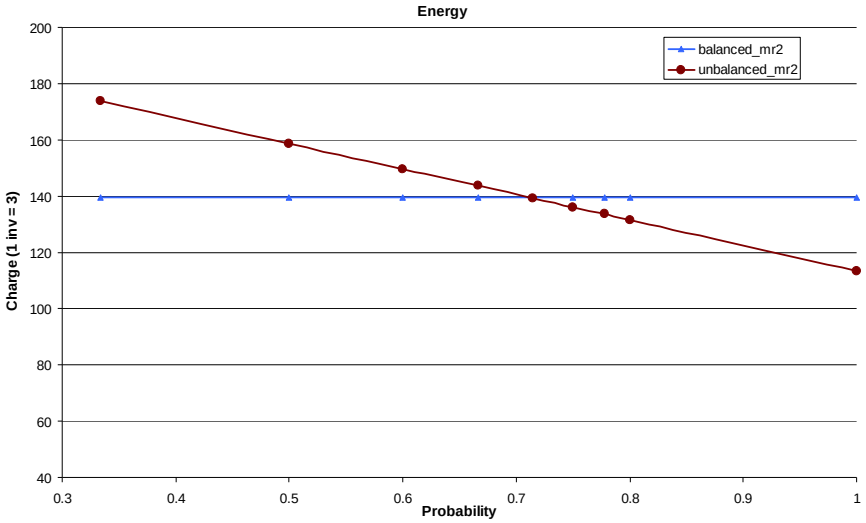


Figure 3-37 : consommation démultiplexeur dual rail

La probabilité sur l'entrée B n'influence pas la consommation, elle a été choisie de 50% pour B_0 et 50% pour B_1. L'énergie consommée en nombre d'inverseurs en fonction des probabilités est représentée dans la Figure 3-37. Cette figure montre que pour une probabilité supérieure a 71% sur la sortie ADD/SUB, il serait plus avantageux d'utiliser cette architecture.

La même étude est réalisée pour un démultiplexeur en 4 rails. La Figure 3-38 et la Figure 3-39 montrent les circuits correspondants au démultiplexeur équilibré et déséquilibré en 4 rails.

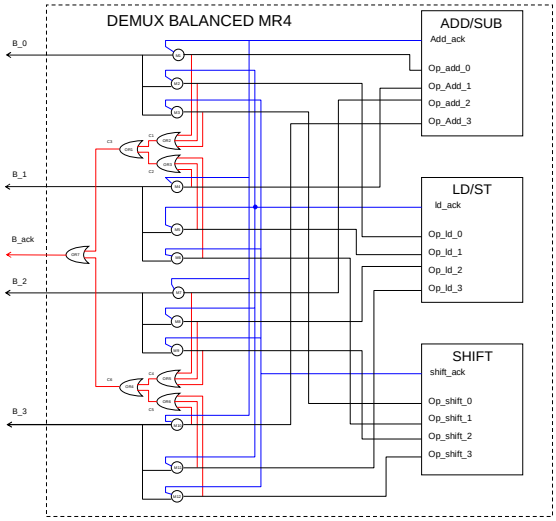


Figure 3-38 : Démultiplexeur équilibré 4 rail

Avec le même outil utilisé précédemment, l'analyse avec les probabilités est effectuée. La Figure 3-40 montre la consommation obtenue pour cette version ainsi que pour la version double rails et présente aussi une troisième version d'un démultiplexeur double rails 2 bits pour comparer ce

dernier avec le démultiplexeur quatre rails. La consommation du circuit en quatre rails est légèrement plus élevée que pour la version en double rails. Ceci est normal puisque avec un codage en quatre rails, 2 bits sont codés. Pour comparer les deux versions, il est nécessaire de construire un démultiplexeur double rails qui gère 2 bits. La consommation de ce circuit est légèrement plus élevée que la consommation que représentent deux circuits double rails. Ceci est dû au fait que pour générer le signal d'acquittement avec le circuit 2 bits en double rails, il est nécessaire d'ajouter une porte de Muller. Il est possible de voir la consommation de cette dernière version dans la Figure 3-40.

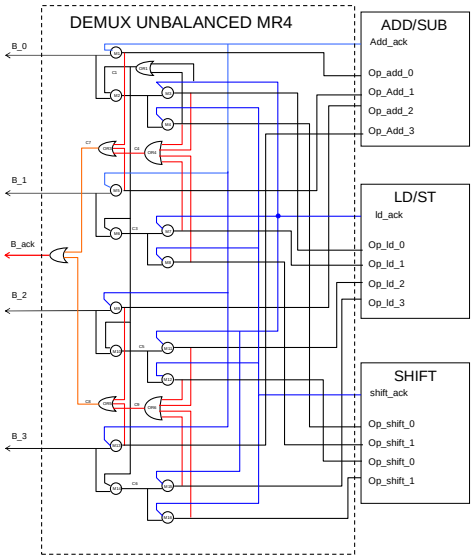


Figure 3-39 : Démultiplexeur déséquilibré 4 rails

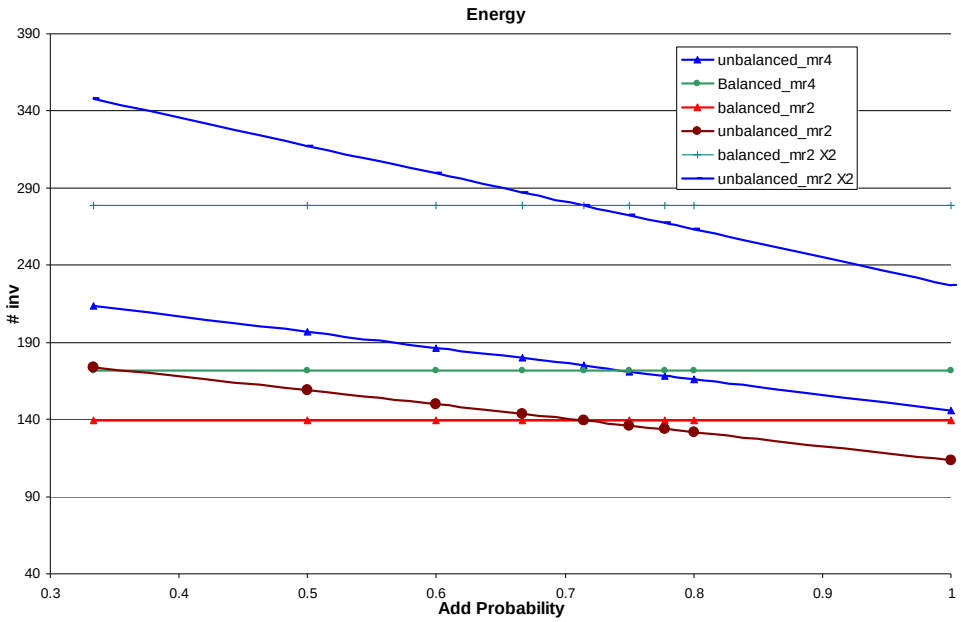


Figure 3-40 : consommation des version 2 rails, 4 rails et 2x2 rails

2.5.2 Conclusions

Avec cette étude, il est possible de voir que déséquilibrer certains blocs, lorsque les données traversent avec plus de fréquence certains chemins, diminue la consommation. Elle montre bien que la consommation avec un codage quatre rails est inférieure à la consommation du codage double rails. Les résultats montrent néanmoins, que le codage en quatre rails augmente la probabilité minimum qui est à atteindre afin d'obtenir une réduction de la consommation. Cette augmentation n'est toutefois pas considérable.

Le désavantage de cette technique est qu'elle implique de réaliser une étude approfondie de l'utilisation des chemins de données dans le circuit. Pour un processeur, cette tâche peut être relativement facile en utilisant les benchmarks existants. Mais pour une application plus spécifique et même pour un processeur qui sera utilisé dans des applications spécifiques, ceci peut être difficile à réaliser.

3 Conclusions

Ce chapitre présente l'analyse de plusieurs architectures d'un microprocesseur asynchrone. La méthodologie présentée dans le chapitre 2 est utilisée pour la conception et pour l'estimation d'énergie des circuits présentés. Il montre dans un premier temps les critères de comparaison utilisés. Une brève présentation de la méthodologie pour la synthèse des circuits asynchrones est fournie. Dans l'étude réalisée, il est possible de voir que la façon dont est décrit le circuit avec le langage de haut niveau CHP, affecte la taille ainsi que la consommation du circuit. En effet l'ordonnancement des entrées a un impact sur les performances du circuit synthétisé.

Une des contributions la plus importante de ce travail est la comparaison des circuits asynchrones avec les circuits synchrones. Cette comparaison est réalisée très tôt dans le flot de conception. La méthodologie présentée permet de réaliser une estimation de l'énergie consommée dans les circuits asynchrones ainsi que dans les circuits synchrones. La fiabilité des résultats obtenus par l'outil a été vérifiée avec des simulations électriques.

Il est possible de voir que les circuits asynchrones ont une consommation inférieure à celle de leur équivalent synchrone, le grand désavantage étant leur taille. Or dans l'étude présentée, les efforts de synthèse des circuits asynchrones n'est pas comparable à ceux des outils de synthèse des circuits synchrones. Ces outils ont plusieurs années d'avance par rapport à l'outil utilisé dans ce travail. Des algorithmes plus agressifs lors de la synthèse devraient optimiser la taille des circuits asynchrones, notamment dans le domaine de l'optimisation logique et de la projection technologique.

La bibliothèque de cellules asynchrone utilisée est aussi cruciale pour diminuer la taille ainsi que la consommation d'énergie. Ce travail utilise la bibliothèque TAL développée au sein du laboratoire TIMA. Or cette bibliothèque est à peine dans le stade de développement. Une nouvelle version de cette bibliothèque est actuellement en conception. Les nouvelles cellules devraient à leur tour réduire la taille du circuit.

En ce qui concerne la vitesse de fonctionnement, les circuits synchrones présentés ont de meilleures performances que les circuits asynchrones. Or la vitesse obtenue dans les circuits synchrones ne prend pas en compte les marges utilisées pour garantir le fonctionnement du circuit en ce qui concerne la fréquence d'horloge maximale de fonctionnement. Il faut aussi prendre en compte que la fréquence maximale de fonctionnement des circuits synchrones sera la fréquence maximale de fonctionnement du bloc le plus lent du circuit complet. Ceci n'étant pas le cas dans les circuits asynchrones puisqu'ils ne dépendent pas de ce signal d'horloge. Chaque instruction dans un processeur asynchrone aura une vitesse d'exécution qui sera toujours le meilleur cas. Finalement les nouvelles cellules de la librairie asynchrone devraient aussi améliorer ce paramètre. Malgré toutes les contraintes liées au manque de maturité des outils asynchrones, les circuits asynchrones présentés dans ce chapitre montrent des performances supérieures en terme d'énergie consommée.

Le signal d'acquittement peut aussi être généré de façon à obtenir une meilleure consommation. Deux manières de générer ce signal sont présentées, chacune donnant des performances différentes. Les entrées et sorties des blocs asynchrones peuvent être conçues de façon active ou passive. Avec des entrées actives, il est possible d'éliminer certains signaux de commande qui permetent de réduire la taille ainsi que la consommation. Finalement, ce chapitre montre l'importance du chemin de données utilisée par les opérandes des instructions.

Chapitre 4

ASSERVISSEMENT DE LA TENSION D'ALIMENTATION POUR REDUIRE LA PUISSANCE CONSOMMEE

Ce chapitre présente la méthode DVS (Dynamic Voltage Scaling, Variation Dynamique de Tension). La première partie du chapitre abordera les principes de base du DVS, puis les limitations de la méthode. L'implémentation d'un système de contrôle à boucle fermée sera présentée par la suite, et finalement il abordera les résultats et conclusions.

1 Principe du DVS

La technique de DVS est une technique qui a déjà fait ses preuves et qui consiste à réduire la consommation d'énergie dans un circuit CMOS en réduisant la tension d'alimentation. Plusieurs travaux peuvent être trouvés sur l'étude de cette technique [FLA 01b, GOV 95, GRU 00, LOR 01, MART 01, PER 98, WEI 94]. Les idées présentées par Weiser et al. [WEI 94] et celles de Govil et al. [GOV 95] forment les bases de ces algorithmes : le temps d'inactivité du processeur doit être minimisé en réduisant la vitesse de ce dernier. L'énergie consommée par le circuit peut être donnée par l'Équation 4 comme mentionné dans le Chapitre 1.

$$E \propto C * V^2$$

Équation 4 : Énergie consommé par un circuit CMOS

C étant la capacité totale commutée et V la tension d'alimentation [BUR 95]. Pour réduire l'énergie consommée par le circuit, il est possible de diminuer la capacité totale du circuit ou bien la tension d'alimentation en utilisant les techniques abordées dans le Chapitre 1. Ce travail propose un système de contrôle en boucle fermée pour contrôler la tension d'alimentation du circuit. En effet, grâce à la relation quadratique entre l'énergie consommée et la tension d'alimentation, une faible diminution de la tension d'alimentation, entraîne une réduction non négligeable de l'énergie consommée. La réduction de la tension est réalisée par la technique de DVS. Cette technique permet effectivement d'obtenir une réduction de 80% de la puissance consommée [PER 98].

Mais cette réduction de la consommation a un coût. En effet réduire la tension d'alimentation réduit par la même la vitesse du circuit. L'Équation 5, avec V la tension d'alimentation et c une constante qui dépend du procédé de fabrication, montre que réduire la tension d'alimentation réduit la vitesse maximale à laquelle le circuit peut fonctionner.

$$F_{\max} \propto \frac{V-c}{V}$$

Équation 5 : Fréquence maximale de fonctionnement

Cette technique est cependant très intéressante car il est possible de choisir une tension d'alimentation pour que la vitesse soit suffisante pour atteindre les dates d'échéance des applications qui sont à exécuter. Il est alors nécessaire que le système utilise le profil de l'application pour contrôler l'énergie nécessaire en contrôlant la tension d'alimentation [FLA 01]. Pour réussir un tel contrôle, une interaction entre le matériel, le logiciel et le système d'exploitation est nécessaire. La Figure 4-1 montre une application exécutée dans le processeur dans une fenêtre de temps T. Cette fenêtre est le temps maximum dans lequel la tâche peut être exécutée. L'énergie consommée est supérieure lorsque aucun contrôle n'est utilisé, mais le temps d'exécution augmente lorsque le contrôle est utilisé.

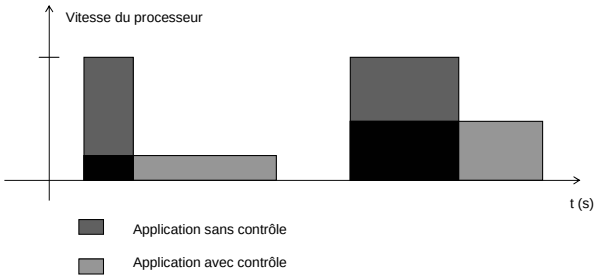


Figure 4-1 : Application avec et sans contrôle DVS

Dans les processeurs nouveaux qui utilisent la technique de DVS, la tension d'alimentation est limitée par une plage de tension qui va de la valeur maximum à la moitié de cette valeur (V_{dd} à $V_{dd}/2$).

Le Tableau 4-1 montre quelques valeurs de tension d'alimentation ainsi que les performances associées pour 3 processeurs commerciaux. La valeur limite inférieure de la tension d'alimentation est en général donnée par la tolérance au bruit des circuits lorsque la tension est réduite. Ceci est vrai pour les circuits où la technique DVS est appliquée sans faire attention à la conception du circuit qui devra être capable de fonctionner sur une plage de tension. Or, il est bien connu que les circuits CMOS peuvent être utilisés à des tensions beaucoup plus basses, et peuvent descendre jusqu'à moins de 200mV. Des recherches ont été réalisées qui montrent que certains circuits peuvent opérer à leur tension de seuil [MIY 02]. Une autre étude montre que la tension minimale d'alimentation dans un inverseur CMOS est de 36mV [MEI 00]. Plusieurs produits commerciaux utilisent des tensions d'alimentation proche de la tension de seuil pour des applications à très basse consommation [MOL 99].

	Plage de tension	Plage de fréquence
IBM PowerPC405LP [ref 17a]	1.0V-1.8V	152MHz-380MHz
TransMeta Crusoe TM58000 [ref 5a]	0.8V-1.3V	300MHz-1GHz
Intel XScale PXA261 [ref 16a]	1.1V-1.43V	200MHz-400MHz

Tableau 4-1 : Plage de tension et fréquence d'utilisation pour quelque processeurs commerciaux

2 Synchrone et Asynchrone

Les processeurs sont des cibles idéales pour utiliser/appliquer la technique de DVS, et ceci est dû au fait qu'ils sont très souvent inactifs. Il existe deux manières de concevoir un processeur de nos jours, la première étant les processeurs synchrones qui dominent le marché actuel, et la deuxième étant les nouveaux processeurs asynchrones.

Les processeurs synchrones ont besoin d'un signal d'horloge pour contrôler le système et pour cadencer les tâches réalisées dans le cœur du processeur. Si la technique de DVS est utilisée dans ce type de processeurs et quand la tension d'alimentation est changée, il est aussi nécessaire de changer la fréquence du signal d'horloge. Il faut alors faire attention à la synchronisation avec la PLL (Phase Locked Loop).

Les processeurs asynchrones deviennent de plus en plus populaires. Ils sont très modulaires et sont, en soit, des processeurs basse consommation. Ce type de processeur n'a pas besoin du signal

d'horloge, tous les blocs qui composent le circuit possèdent un contrôle local à l'aide d'un protocole de type poignée en main. Ceci permet de changer la tension d'alimentation sans avoir à changer le signal d'horloge. Avec ce type de processeur, la taille et la complexité du circuit de contrôle pour réaliser la technique DVS sont réduites.

Ces propriétés rendent les processeurs asynchrones des candidats idéaux pour l'utilisation de la technique DVS [MAR 00]. La réduction de la complexité du circuit de contrôle fait que le gain obtenu par cette technique, est supérieur au gain obtenu dans les circuits synchrones. Le travail présenté par la suite cible l'utilisation de processeurs asynchrones, cependant les concepts qui seront abordés ainsi que la méthode décrite, peuvent être aussi utilisés dans un processeur synchrone.

3 Contributions

De nos jours, la complexité des instructions et le temps d'accès à la mémoire à travers plusieurs niveaux de cache, rendent très difficile de prédire le temps exact d'exécution d'un programme ainsi que la vitesse réelle du processeur. En général, les circuits qui utilisent la technique DVS, assument le pire cas d'exécution d'une tâche, et calculent la tension d'alimentation en fonction de cette hypothèse. La conséquence de cette hypothèse est que le processeur travaille beaucoup plus vite que la vitesse nécessaire pour réaliser la tâche en question.

Ce travail présente la conception d'un co-processeur pour la réalisation de la technique DVS dans un processeur asynchrone. Le co-processeur contrôle le régulateur de tension DC/DC pour changer la tension d'alimentation du processeur. Dans le cas des processeurs synchrones, le co-processeur changera aussi la fréquence en fonction de la tension d'alimentation générée. Le co-processeur reçoit en entrée un signal qui indique la fin d'une instruction exécutée par le processeur et la consigne de la vitesse de fonctionnement nécessaire calculée par le système d'exploitation du processeur. Le régulateur est contrôlé dès que le co-processeur a calculé la vitesse réelle du processeur. Cette vitesse est calculée en temps réel par le co-processeur, et donne une valeur plus exacte de la vitesse moyenne du processeur. Avec cette vitesse, le processeur peut être alimenté avec une tension plus basse que celle qui aurait été calculée par une technique DVS ordinaire.

La technique DVS pour réduire la consommation d'un circuit n'est pas nouvelle en soit, elle a déjà été utilisée dans de nombreuses applications. Ce travail présente trois nouvelles contributions à cette technique abordées par la suite :

- Modélisation du contrôle de puissance avec un système de contrôle.
- Description d'un co-processeur qui permet l'utilisation d'un contrôle à boucle fermée pour contrôler le régulateur DC/DC afin de répondre aux besoins de vitesse du système.

- Modélisation d'un système DVS qui calcule la vitesse en temps réel du processeur.

4 Présentation du système

Contrôler la puissance consommée par un processeur est une tâche difficile. L'utilisation de la technique DVS permet de réduire la consommation du circuit. Or il est difficile d'obtenir la tension idéale d'alimentation pour avoir la consommation minimale ainsi que pour respecter les temps de calcul maximum nécessaire lors de l'exécution d'une application. Pour obtenir la tension d'alimentation, le contrôle d'un régulateur DC/DC est nécessaire. Il existe néanmoins deux types de système de contrôle : à boucle ouverte et à boucle fermée. Dans les systèmes à boucle ouverte, le contrôleur envoie une commande qui dans le cas du contrôle DVS demande une tension d'alimentation. Cependant, le système n'est pas capable de vérifier si le résultat obtenu avec cette commande satisfait les besoins du système. Avec un contrôle à boucle fermée, le contrôleur observe les performances du système et change les commandes pour assurer le fonctionnement correct du système.

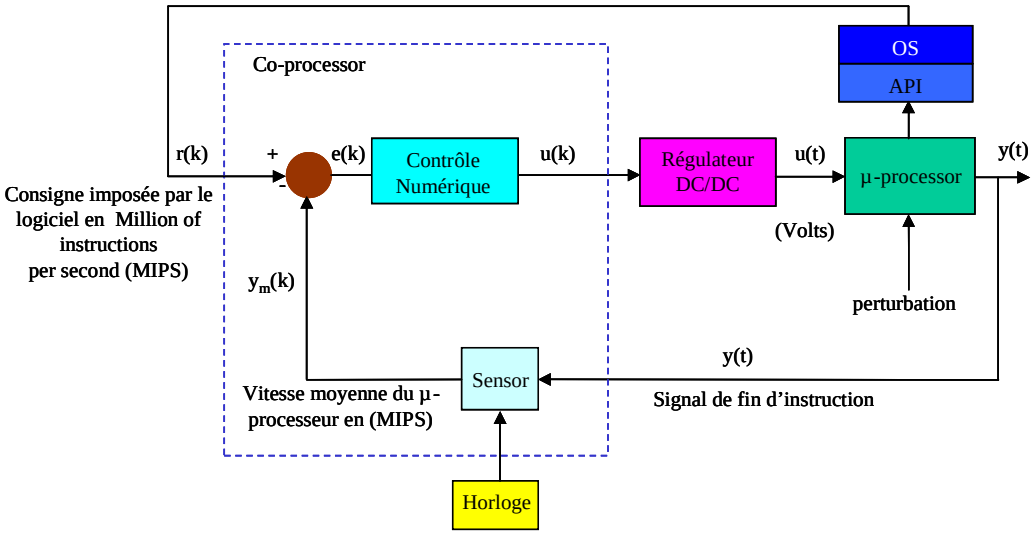


Figure 4-2 : Diagramme de bloc du système

Avec un contrôle à boucle fermée, il est possible de réaliser un contrôle très performant pour obtenir la tension idéale tout en respectant les spécifications et besoins de l'application qui est exécutée par le processeur. La Figure 4-2 présente le système proposé. Le contrôle est réalisé en obtenant deux informations fournies par le processeur. La première étant la consigne donnée par l'application qui est exécutée pour connaître la vitesse à laquelle elle doit être exécutée. Cette consigne peut être gérée par l'intermédiaire d'une API (Application Programming Interface) ou du système

d'exploitation (OS Operating system). La vitesse nécessaire peut être insérée dans le code lors de la compilation du programme, ou bien elle peut être calculée en temps réel par le système d'exploitation. La deuxième information est un signal de fin d'instruction donné par le processeur qui permet au co-processeur de calculer la vitesse réelle d'exécution du processeur. Avec ces deux informations, il est possible d'obtenir un profil de l'exécution des tâches dans le temps. Ce profil permet donc d'appliquer un contrôle très précis pour répondre aux dates d'échéance des tâches exécutées par le processeur.

Le co-processeur présenté dans la Figure 4-2 intègre un compteur d'instructions et une horloge pour avoir une base de temps. Avec le signal de fin d'instruction envoyé par le processeur, le co-processeur calcule la vitesse moyenne en MIPS (Million Instructions Per Second, Millions d'Instructions Par Secondes) du processeur en temps réel. Cette vitesse est une vitesse moyenne dans une période d'exécution donnée par la période de l'horloge du co-processeur. La vitesse est alors comparée avec la consigne donnée par le processeur. Cette comparaison est effectuée en réalisant une soustraction entre la consigne et la vitesse calculée ce qui permet d'obtenir l'erreur. Elle est ainsi appliquée à l'entrée du contrôleur numérique qui génère alors le signal de contrôle du régulateur DC/DC.

4.1 Calcul de la vitesse

Le processeur exécute les instructions d'une application une par une. Le co-processeur reçoit alors par l'intermédiaire du signal de fin d'instruction, un flux d'instructions exécutées. La quantité d'instructions exécutées pendant une période de temps donnée est la vitesse moyenne avec laquelle le processeur travaille.

Dans un processeur synchrone, la vitesse réelle d'exécution des instructions n'est pas nécessairement la vitesse du signal d'horloge. En effet, dans un processeur synchrone, certaines instructions prennent plusieurs cycles d'horloge pour être exécutées. Le temps d'accès à la mémoire oblige d'autres instructions à être exécutées dans plusieurs cycles d'horloge, les instructions telles que les " wait on interupt " (attendre interruption), rendent l'estimation de la vitesse d'exécution très difficile. En ce qui concerne les processeurs asynchrones, ils exécutent les instructions avec le temps exact nécessaire. Ce temps est en général différent pour chaque type d'instruction. Il est donc impossible de prédire la vitesse d'exécution. Dans les deux cas, synchrone ou asynchrone, il est nécessaire d'intégrer un composant matériel pour calculer la vitesse moyenne du processeur. Dans ce travail, la vitesse moyenne est calculée à l'aide d'une base de temps donnée par l'horloge interne du co-processeur.

Lors de l'exécution d'une application, deux phénomènes qui font varier la vitesse calculée peuvent se présenter. Le premier est dû à l'architecture du processeur. Comme présenté dans le paragraphe précèdent, une architecture asynchrone réalise chaque instruction avec un temps différent. Ce phénomène est modélisé par l'insertion d'une perturbation dans le processeur (voir Figure 4-2). L'avantage du système de contrôle proposé est qu'il est capable de compenser ces variations. Le deuxième phénomène survient lorsque la tension (et la fréquence pour les processeurs synchrones) est en train de changer. Pendant ce laps de temps, la vitesse initiale du processeur S_1 change au fur et à mesure que la tension (fréquence) varie (Figure 4-3). Lorsque le front montant du signal d'horloge du co-processeur arrive, la vitesse du processeur est calculée. Cette vitesse est calculée avec le nombre d'instructions ayant été comptées pendant la période de l'horloge. La vitesse réelle étant S_2, la vitesse calculée sera inférieure ou supérieure puisque le nombre d'instructions exécutée durant la période d'horloge ne correspond pas au nombre d'instructions que le processeur peut exécuter à cette nouvelle vitesse. Cette erreur de calcul peut perturber le contrôle du régulateur DC/DC.

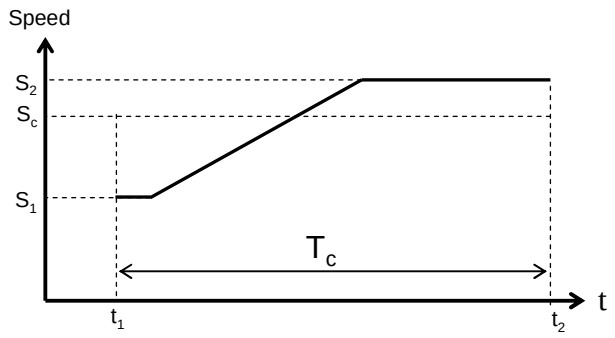


Figure 4-3 : Evolution de la vitesse pendant un changement de tension

4.2 Contrôle du régulateur DC/DC

Le contrôle du régulateur DC/DC peut être réalisé de plusieurs manières. Les approches les plus classiques étant un contrôle de type proportionnel, dérivatif ou intégrateur. Ce type de contrôle peut être utilisé séparément ou en utilisant deux ou les trois en même temps. Il est aussi possible de contrôler le système avec de la logique floue (fuzzy logic). La logique floue présente l'avantage de ne pas avoir besoin de réaliser une analyse de modélisation du système à contrôler. Par la suite, une brève explication des deux types de contrôle est abordée.

4.2.1 Contrôle avec logique Floue

Le contrôle flou utilise la logique floue pour calculer les commandes nécessaires aux contrôles d'un système à boucle fermée. La logique floue a été conçue par Lotfi Zadeth [ZAD 65] dans les

années 60. Elle traite les problèmes de la même manière que le raisonnement humain. Elle est très adaptée aux problèmes pour lesquels obtenir une modélisation mathématique du système est très compliqué voire impossible. Ceci à cause du manque d'informations des variables qui composent le système ou tout simplement parce que le système est trop complexe. Un autre avantage de la logique floue est qu'il est aussi possible de contrôler de la même façon des systèmes linéaires que des systèmes non linéaires. Il est néanmoins important de noter que l'utilisation de la logique floue ne vise pas à réaliser des résultats aussi précis que les techniques de contrôle traditionnelles, mais elle donne en général des résultats acceptables dans un temps très court.

4.2.2 Contrôle PID

Le contrôle PID est la technique la plus utilisée jusqu'à présent. Il est composé de trois types de contrôle : proportionnel, intégratif et dérivatif. Avec ce type de contrôle, il est possible d'obtenir un contrôle très précis, et il existe plusieurs logiciels pour le développement et la modélisation du système [MAT 07] [PID 07]. Le diagramme de bloc basique d'un système à contrôle PID est montré dans la Figure 4-4.

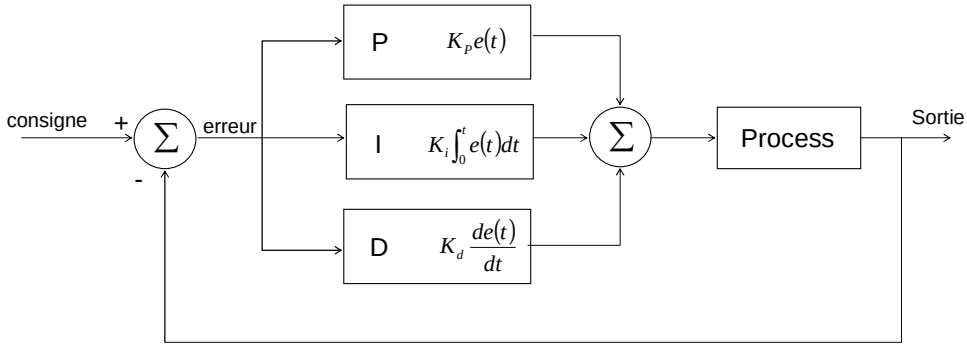


Figure 4-4 : diagramme de bloc d'un contrôle PID

4.3 Paramètres du contrôle choisi

Bien que la recherche dans les contrôles à logique floue soit actuellement dans une phase de maturité, pour cette étude un contrôle de type PID a été choisi puisque les performances de ce type de contrôle peuvent être très bien paramétrées. En effet, il est possible de contrôler le temps de réponse, le surpassement et les oscillations dans un contrôle de ce type. Le calcul des paramètres du contrôleur dépend principalement de trois éléments, le premier étant le processeur cible, le deuxième étant la fréquence d'horloge du co-processeur et le dernier étant le régulateur DC/DC qui va être utilisé. Pour chaque processeur, il est nécessaire de caractériser le système pour pouvoir calculer les paramètres du contrôleur.

4.4 Fréquence du signal d'horloge du co-processeur

La période du signal d'horloge est très importante car elle détermine la précision de calcul de la vitesse moyenne du processeur. Elle détermine aussi la vitesse de réponse du système. Le co-processeur mémorise le nombre d'instructions exécutées et calcule la vitesse moyenne du processeur à chaque front montant du signal d'horloge. Si la période du signal d'horloge est trop courte, le calcul de la vitesse moyenne sera effectué à une fréquence élevée, et par conséquent le temps de réponse du système sera plus élevé. Or la précision de la vitesse moyenne calculée diminue si la période de l'horloge diminue. D'un autre coté, si la période de l'horloge augmente, la précision de la vitesse moyenne calculée augmente, mais le temps de réponse du système diminue.

Il n'est pas possible de trop augmenter la fréquence d'horloge même si l'on veut un temps de réponse élevé. Pour donner un exemple, si le processeur est à une vitesse de 1MIPS, chaque instruction s'exécute en moyenne toutes les 1µs. En fixant une fréquence d'horloge a 32kHz, la vitesse du processeur sera calculée toutes les 31,25 instructions en moyenne. Avec cette fréquence, le régulateur DC/DC est commandé toutes les 31,25µs. Pour obtenir un temps de réponse plus rapide, il faut donc augmenter la fréquence d'horloge. Dans l'exemple précèdent, si l'on augmente la fréquence à 320kHz, le régulateur est commandé toutes les 3,125µs mais le calcul de la vitesse du processeur sera effectué toutes les 3,125 instructions. Il est clair que l'exactitude du calcul diminue si la fréquence augmente. Ceci peut aussi entraîner des oscillations dans le système. Il est indispensable de bien choisir les paramètres du contrôleur pour avoir le meilleur gain en consommation. La fréquence d'horloge choisie dans cette étude sera montrée dans la section 5.4.

5 Modélisation et simulation du système

Le système a été modélisé avec le langage VHDL pour les parties numériques, et avec le langage VHDL-AMS et Spice pour les parties analogiques. La simulation a été réalisée avec le simulateur Modelsim de Mentor qui permet la simulation mixte des parties numériques et analogiques.

Le système proposé est montré dans la Figure 4-5. Il est composé de trois blocs principaux. Le premier en VHDL-AMS modélise le processeur, le deuxième est le régulateur DC/DC qui alimente le processeur et qui peut être contrôlé par le co-processeur. Ce dernier constitue le troisième bloc du système modélisé en VHDL et VHDL-AMS. Par la suite le travail présentera en détail chaque bloc du système.

5.1 Processeur

Ce travail cherche à montrer les bénéfices de la technique DVS avec le contrôle en boucle fermée. La manière idéale pour modéliser le processeur est d'avoir le modèle au niveau transistors et de simuler le système avec des vecteurs d'entrée qui reproduisent l'exécution d'une application ainsi que le système d'exploitation dans le processeur. Ceci étant une approche très lourde, deux niveaux d'abstraction ont été pris en compte dans cette étude pour modéliser et, de cette façon, alléger le modèle du processeur. Les deux niveaux étant au niveau logique et au niveau électrique. Le niveau logique fait référence à la génération du signal de fin d'instructions qui servira à compter les instructions exécutées au cours d'une période et ainsi calculer la vitesse moyenne du processeur. Le niveau électrique est utilisé pour calculer la consommation d'énergie du processeur en soit.

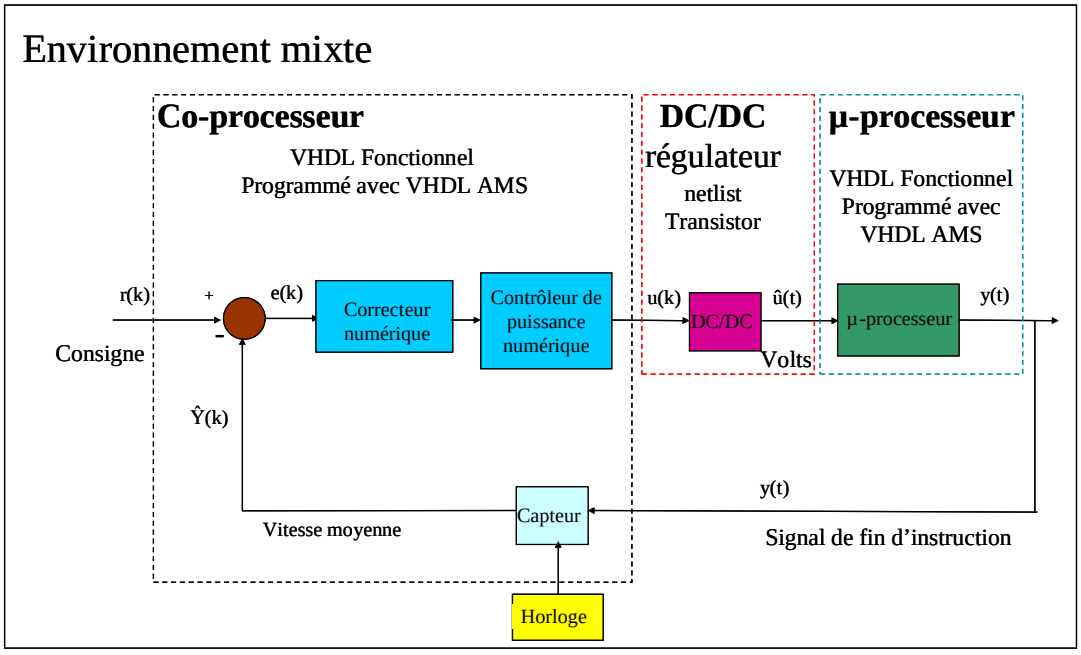


Figure 4-5 : Système proposé

5.1.1 Niveau logique

Le signal de fin d'instructions peut être réalisé avec un modèle simple qui génère une fréquence en fonction de la tension d'alimentation. Les processeurs qui existent dans la recherche et dans le commerce donnent en général les spécifications de performance du processeur en fonction de la tension d'alimentation. Il est donc facile de créer un modèle avec les données du constructeur. Bien sûr, la vitesse d'exécution sera une vitesse moyenne, et ne prend pas en compte les possibles variations du temps d'exécution des instructions. Ce modèle sera utilisé dans un premier temps pour valider le système proposé. Cependant, il est possible de modéliser la génération du signal de fin d'instruction avec un modèle plus complexe qui donnerait une fréquence variable en fonction de la tension

d'alimentation, ainsi que d'autres critères qui pourraient être introduits dans le modèle. Ces critères peuvent être des critères qui prennent en compte le type d'application qui est exécuté dans le processeur. En effet selon le type d'application, les instructions les plus utilisées sont différentes, et lorsque les instructions sont plutôt du type accès à la mémoire, le temps d'exécution entre instructions aura tendance à être beaucoup plus inégal.

5.1.2 Niveau électrique

Au niveau électrique, il est possible d'avoir un modèle complet du processeur comme mentionné dans les paragraphes précédents. Or, ce modèle est souvent très difficile à obtenir si le processeur qui va être utilisé est un processeur du commerce. De plus, la simulation de tels modèles requiert des ressources de calcul très importantes, et en général les temps de simulation sont très grands. Pour valider la méthode proposée, il est possible d'utiliser un modèle beaucoup plus simple qui consiste à modéliser le processeur avec une résistance qui varie en fonction de la tension d'alimentation. Ce modèle est très simple, mais il donne une bonne idée du courant consommé par le processeur en fonction de la tension d'alimentation. Modéliser ainsi le processeur est assez simple puisque les constructeurs fournissent en général la consommation du processeur en fonction de leur tension d'alimentation.

Ce travail utilise les caractéristiques et performance du processeur Luthonium asynchrone réalisé par A. Martin et al. [MAR 03] pour simuler et valider la méthode proposée.

5.2 Régulateur DC/DC

Le choix du régulateur DC/DC est très important pour le contrôle de la tension d'alimentation. Plusieurs critères doivent être pris en considération, tout d'abord l'efficacité du régulateur. En effet le régulateur doit être un régulateur qui puisse changer sa tension de sortie avec une tension d'entrée fixe. Les régulateurs ont une efficacité de l'ordre de 98% lorsqu'ils doivent fournir une tension fixe. Mais quand ils fournissent des tensions variables, leur rendement peut descendre jusqu'à 40%. Le rendement d'un régulateur DC/DC correspond au pourcentage d'énergie qu'il fournit en sortie par rapport à l'énergie présente sur son entrée. Un régulateur qui a un rendement de 100% ne consomme pas d'énergie tandis qu'un régulateur qui a un rendement de 60%, fournit 60% de l'énergie réelle de la source d'alimentation (batterie, secteur) et en consomme 40%. Un autre critère est la manière avec laquelle le régulateur va être contrôlé. Dans le système proposé c'est le co-processeur qui génère la commande du régulateur. En conséquence, plus cette commande est complexe, plus le co-processeur consommera d'énergie pour la générer. Enfin, un autre critère à prendre en compte, est le temps de réponse du régulateur. Le but étant de contrôler le régulateur à une fréquence de 32kHz donnée par

l'horloge du co-processeur, le régulateur doit être capable de changer sa tension de sortie à une vitesse qui soit supérieure à la fréquence de l'horloge du co-processeur. Si le régulateur prend plus de temps pour changer la tension, le système risque d'osciller.

Deux types de régulateurs ont été choisis dans ce travail. Le premier est un régulateur qui peut changer sa tension de sortie de manière continue. Ce régulateur est capable de fournir la tension d'alimentation exacte pour que le processeur exécute les instructions dans le temps maximum permis par l'échéance de l'application en cours. Le deuxième est un régulateur qui donne une tension de sortie par palier. Ce régulateur ne donne pas la tension exacte requise.

5.2.1 Pompe de charge

Le premier régulateur choisi est un régulateur qui est réalisé avec une pompe de charge. La Figure 4-6 montre le schéma de la pompe de charge. L'idée de cette pompe de charge est de faire varier la tension de la capacité de charge à travers les transistors N et P pour ainsi obtenir une tension variable dans la grille du transistor T1. Le transistor T1 donne une tension sur son drain qui est fonction de sa tension de grille. Cette pompe de charge est alors contrôlée avec une commande sur deux bits. Les deux bits de contrôle étant connecté respectivement sur les grilles des transistors P et N. Si la tension d'entrée sur la grille du transistor P est à un état logique bas, la tension de la capacité de charge augmente et la tension de sortie du régulateur diminue. Si au contraire, la tension de la grille du transistor N est à un niveau logique haut, la tension de la capacité de charge diminue et la tension de sortie du régulateur augmente. Si les deux transistors N et P sont bloqués, la tension de sortie ne change pas. Il est important dans ce type de régulateur que les transistors N et P ne soient pas passants en même temps, car ceci provoquerait un court circuit.

La pompe de charge a été décrite avec un fichier de description en transistor (netlist) avec une technologie HCMOS9 (130nm) de STMicroelectronics. Ce régulateur a le désavantage d'avoir une efficacité très faible dans les faibles tensions.

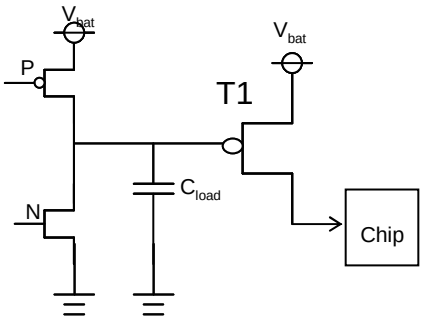


Figure 4-6 : schéma de la pompe de charge

5.2.2 Régulateur par paliers

Le deuxième type de régulateur est un régulateur par paliers. Ces régulateurs ne donnent pas une tension continue en sortie, ils sont conçus pour fournir un certain nombre de tensions fixes. Ces régulateurs ont un rendement très grand qui peut aller au-delà de 85%. Avec cette solution, la tension de sortie ne pourra pas être la tension exacte nécessaire pour alimenter le processeur. La tension de sortie devra toujours être supérieure à la tension voulue. Le processeur exécute les instructions à une vitesse plus rapide et finit alors l'application courante avant la date d'échéance. La consommation d'énergie n'est pas optimum de cette manière, mais le rendement de ces régulateurs est très intéressant par rapport à la pompe de charge montrée précédemment.

Le système proposé utilise deux régulateurs par paliers. La différence étant le nombre de tensions de sortie que le régulateur peut générer. Le premier ayant une commande de 4 bits pour générer 16 tensions de sorties différentes. Le deuxième régulateur est commandé avec 2 bits qui permettent de générer 4 tensions de sortie différentes.

Les deux régulateurs ont été modélisés avec VHDL-AMS. Le régulateur 2 bits génère les quatre tensions de sortie suivantes : 0,4V, 0,666V, 0,933V et 1,2V. Pour les tensions du régulateur 4 bits, la première tension commence à 0,543V avec un pas de 0,043V ; ceux-ci fournissent en tout 15 tensions. Un palier spécial a été crée à 0,4V lorsque le processeur est inactif.

5.3 Profil de l'application

L'exécution d'une application dans un processeur est constituée d'une série d'instructions qui sont exécutées une par une. Il existe deux types d'applications qui peuvent se présenter dans un processeur. Tout d'abord les tâches propres au fonctionnement du programme qui est dans le processeur. Puis, les tâches qui arrivent lors d'une interruption. Ces interruptions peuvent être dues à des événements externes comme la réception de données ou des signaux qui demandent l'exécution d'une application spécifique. Les interruptions peuvent aussi provenir des modules internes du processeur. Ceci est très courant pour les modules tels que les compteurs internes (timer), la fin de la transmission d'une donnée, l'écriture dans la mémoire, etc.

Les tâches propres à l'application qui s'exécute dans le processeur peuvent avoir une date d'échéance mais elles peuvent aussi ne pas être contrainte en temps d'exécution. Plusieurs profils d'application peuvent donc être observés dans le fonctionnement d'un processeur. Ce travail présente quelques cas typiques des profils d'application qui peuvent se produire.

Le premier cas survient lorsque l'application du processeur n'a pas de contrainte de temps. Dans ce cas, le processeur peut travailler à la plus petite vitesse disponible. Le contrôle pour ce type d'application n'est pas nécessaire puisque le processeur exécute toujours les instructions à la même vitesse. Or, ce cas n'est pas très commun, et, en général, il existe au moins une source d'interruption dans le processeur. Ces interruptions ont généralement une date d'échéance puisqu'elles risquent de se reproduire soit de manière sporadique soit de manière cyclique. Un profil d'application plus commun regroupe le premier profil ou les tâches s'exécutent en un temps non borné, avec l'insertion des tâches générées par les interruptions. De manière plus générale, le processeur exécute des tâches sporadiques qui ont une date d'échéance avec des tâches générées par des interruptions internes ou externes qui elles aussi sont bornées dans le temps.

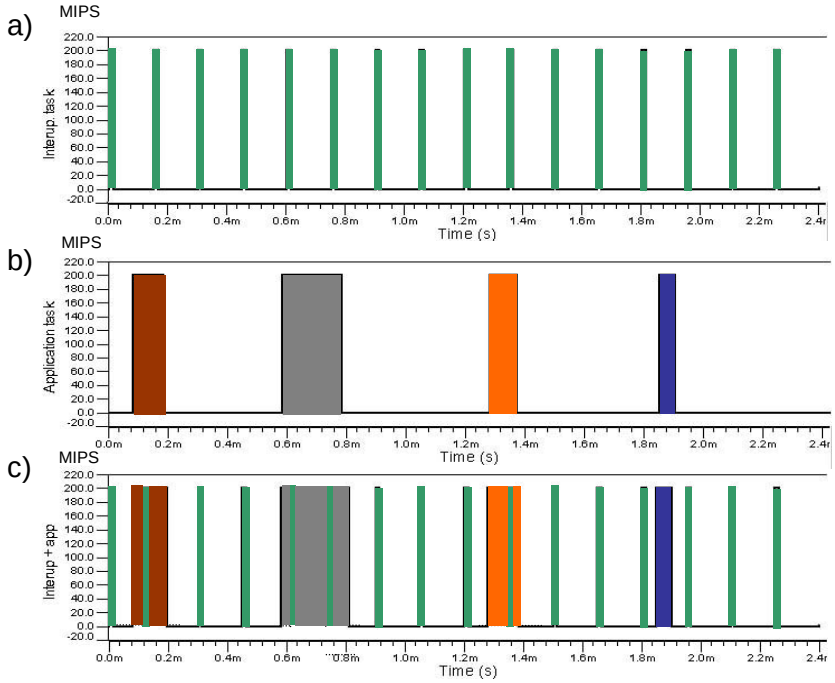


Figure 4-7 : Exemple de profil d'application

Pour valider le système, un exemple d'un profil d'application est présenté dans la Figure 4-7. Ce profil représente le fonctionnement typique d'un nœud dans un réseau de capteur. Il est composé d'une tâche qui provient d'une interruption externe périodique et de l'application propre au nœud. Ce type de profil réduit généralement l'efficacité de la technique DVS dans les processeurs synchrones à cause de la synchronisation avec la PLL. La tâche cyclique dans la Figure 4-7.a représente l'arrivée de données qui proviennent du capteur du nœud. Ces données arrivent de manière périodique et le processeur doit alors recevoir, traiter et envoyer ses résultats avant que la donnée suivante ne soit arrivée. En même temps, le processeur exécute des tâches propres au fonctionnement comme c'est le cas des tâches du système d'exploitation. Le profil dans la Figure 4-7.b représente ces tâches. Lorsque le processeur travaille de manière normale, le profil des tâches exécutées est représenté dans la Figure

4-7.c. Ce profil montre clairement que les tâches qui sont exécutées dans le processeur sont interrompues chaque fois que les données arrivent. Le processeur arrête l'exécution des tâches en cours pour répondre à l'interruption. Ce profil considère que le processeur travaille à la vitesse maximale de 200MIPS. Les échéances des tâches dans ce profil seront données par le début de la tâche suivante, ceci constitue un cas idéal pour les tâches sporadiques, mais représente le cas réel pour les tâches cycliques.

La consigne que reçoit le co-processeur varie en fonction de la manière dont le système d'exploitation gère les dates d'échéance. En effet, pour calculer la vitesse à laquelle le processeur doit exécuter les instructions, les nouvelles tâches doivent soit donner leur date d'échéance avec le nombre d'instructions à exécuter (dans le pire cas) soit donner elles-mêmes la consigne de vitesse nécessaire à leur exécution. Dans le premier cas, le système d'exploitation calcule la vitesse moyenne nécessaire à l'application. Chaque fois qu'une application se met à la queue, une nouvelle consigne est calculée. Dans le deuxième cas, le système d'exploitation doit tout simplement additionner les consignes au fur et à mesure qu'elles arrivent. En utilisant un ordonnancement d'application tel que le EDF (Earliest Deadline First) [ZHU], le système d'exploitation connaît exactement la vitesse à laquelle le processeur doit exécuter les instructions pour répondre aux échéances des applications. Quand une tâche est finie, la vitesse d'exécution nécessaire à cette tâche est soustraite et la nouvelle consigne est calculée.

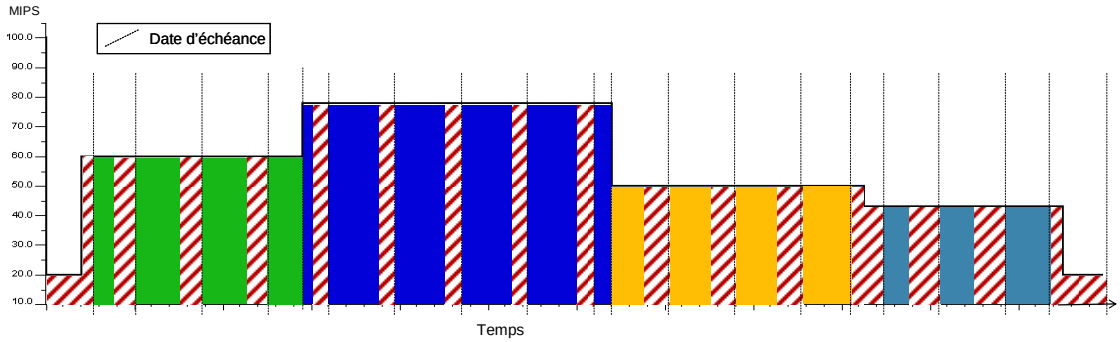


Figure 4-8 : Consigne calculée.

Avec un régulateur DC/DC qui fournit une tension en sortie continue, la vitesse du processeur suit la consigne donnée au co-processeur. La Figure 4-8 montre la consigne qui a été calculée par le système d'exploitation à partir du profil montré dans la Figure 4-7.c. Les polygones hachurés représentent les tâches périodiques, les autres polygones correspondent aux tâches sporadiques. Cette consigne est la consigne optimale, puisqu'elle considère que la tension d'alimentation du processeur peut être générée de façon continue et qu'elle peut prendre toutes les valeurs possibles. La tension dans ce cas sera une image de la consigne. Cela suppose aussi que le temps de réponse du régulateur est très petit. Dans un cas réel, le fait de changer la tension n'est pas instantanée, en conséquence la

vitesse du processeur ne change pas de façon immédiate. Ce temps doit être compensé par le système d'exploitation.

Avec un régulateur par palier, la tension ne peut pas être la tension exacte nécessaire pour alimenter le processeur. C'est pourquoi la vitesse du processeur ne peut pas être la vitesse requise par l'application pour répondre au temps maximal disponible. Dans ce type de régulateur, la tension générée doit forcement être plus élevée pour assurer que la date d'échéance soit respectée. Ceci fait que le processeur travaille plus vite et consomme plus d'énergie que dans le cas idéal où la tension est exacte.

Il est possible néanmoins de contrôler dynamiquement la tension d'alimentation pour la réduire lorsque le nombre d'instructions permet de changer la tension d'alimentation à une tension plus basse. En effet, le processeur qui utilise ce type de régulateur DC/DC, commence à travailler à une vitesse supérieure à la vitesse nécessaire idéale. Après un certain temps, les instructions qui n'ont pas été exécutées, permettent de réduire la vitesse du processeur tout en respectant la date d'échéance.

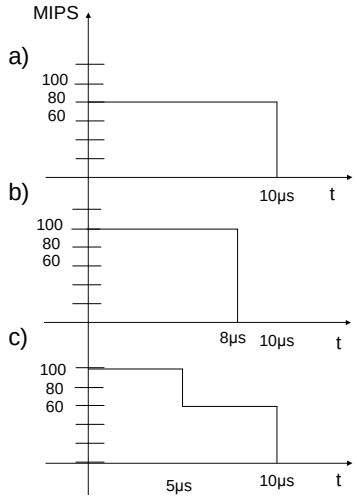


Figure 4-9 : Exemple de contrôle dynamique

La Figure 4-9 montre un exemple d'une tâche qui a besoin d'une vitesse de 80MIPS pendant 10µs pour être exécutée à temps. Si le régulateur ne peut que fournir deux tensions pour que le processeur travaille à 100MIPS et à 60MIPS, il sera nécessaire de travailler à une vitesse de 100MIPS au début de l'exécution de la tâche. La Figure 4-9a présente le cas idéal lorsque la tension fournie correspond à une vitesse de 80MIPS. La Figure 4-9.b montre le profil de la consigne nécessaire pour cette application lorsque le régulateur ne peut fournir qu'une tension qui fait travailler le processeur à une vitesse de 100MIPS. La Figure 4-9.c montre comment il est possible de changer la consigne pour réduire la vitesse du processeur une fois que les instructions restantes peuvent être exécutées à une vitesse inférieure tout en respectant la date limite d'exécution. Avec ce type de contrôle, il est possible

d'optimiser l'énergie consommée par le processeur qui utilise des régulateurs DC/DC par palier. Le contrôle est bien sûr plus compliqué et consomme plus d'énergie que le contrôle d'un régulateur à tension continue. Or, ce contrôle est très intéressant, l'étude exposée dans ce chapitre montrera que les résultats obtenus avec cette politique de contrôle sont très attractifs.

5.4 Co-processeur

Le co-processeur proposé est composé de deux parties. La première est un compteur d'instructions qui s'incrémente à chaque front montant du signal de fin d'instructions envoyé par le processeur. Une horloge de 32kHz commande le calcul de la vitesse moyenne du processeur. L'idée d'utiliser une horloge à basse fréquence est d'obtenir une faible consommation du système en soi, mais qui soit assez rapide pour répondre aux changements de la consigne ainsi qu'aux changements de la vitesse du processeur. La variation de la vitesse se traduit par un changement de l'erreur commise entre la vitesse moyenne calculée et la consigne due aux variations du temps d'exécution des instructions.

La deuxième partie est le contrôle en soit. Dans le système proposé, ce contrôle est un contrôle proportionnel qui a rendu des résultats satisfaisants pour réduire la consommation du processeur avec le profil d'application choisi. Trois types de contrôle proportionnel ont été conçus pour commander les trois régulateurs DC/DC utilisés dans cette étude.

Le premier contrôle génère une sortie sur deux bits pour contrôler le régulateur à tension continue. Le contrôle génère un signal de deux bits avec une durée variable qui dépend de l'erreur en entrée. Cette impulsion contrôle alors le régulateur en lui indiquant d'augmenter ou de diminuer sa tension de sortie. Si le temps de l'impulsion est grand, le régulateur baisse ou augmente fortement la tension de sortie. Si cette impulsion est courte, la tension diminue ou augmente finement. La Figure 4-10 montre un exemple de la génération du signal de commande pour le régulateur à tension continue. Dans le premier front montant de l'horloge, l'erreur entre la consigne et la vitesse calculée est de 90MIPS, le contrôle génère une impulsion avec un temps assez grand pour augmenter la tension de sortie. Quand l'impulsion de contrôle est finie, l'erreur est encore de 20MIPS. Il faut noter que l'erreur est calculée au front montant de l'horloge, il faudra attendre le front d'horloge suivant pour calculer l'erreur et produire une nouvelle commande. Dans cet exemple, lorsque le nouveau front d'horloge arrive, l'erreur n'est plus que de 5MIPS, le contrôle génère alors une nouvelle commande qui sera plus courte que la précédente et ainsi de suite jusqu'à ce que l'erreur soit nulle.

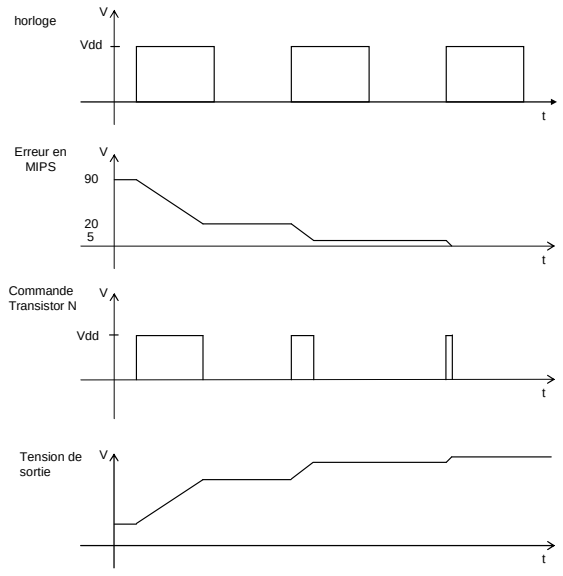


Figure 4-10 : Exemple de la commande pour le régulateur a tension continue

Dans cet exemple, l'erreur est toujours positive et la commande générée est envoyée au transistor N de la pompe de charge. Si l'erreur est négative, le contrôle se déroule de la même manière, mais ce sera le transistor P de la pompe de charge qui recevra la commande pour diminuer la tension d'alimentation.

Dans le cas des deux autres contrôles implémentés, le co-processeur génère un signal de respectivement deux et quatre bits qui contrôlent chaque régulateur. Le calcul de la commande de sortie, et donc de la tension désirée, est effectué par le contrôle proportionnel en fonction de l'erreur d'entrée.

6 Résultats obtenus

6.1 Régulateur à tension continue

Le régulateur composé de la pompe de charge montrée dans la Figure 4-6 est utilisé avec le profil d'application montré dans la Figure 4-7. Avec la consigne montrée dans la Figure 4-11.a, le régulateur délivre la tension montrée dans la Figure 4-11.b. Finalement, la Figure 4-11.c montre la vitesse de fonctionnement obtenue. La consigne est générée statiquement car le régulateur donne une tension continue qui fournira la vitesse exacte nécessaire pour répondre aux besoins de l'application exécutée. Il est possible de voir dans la Figure 4-11.b comment la tension Vdd est régulée en continu par le système en boucle fermée.

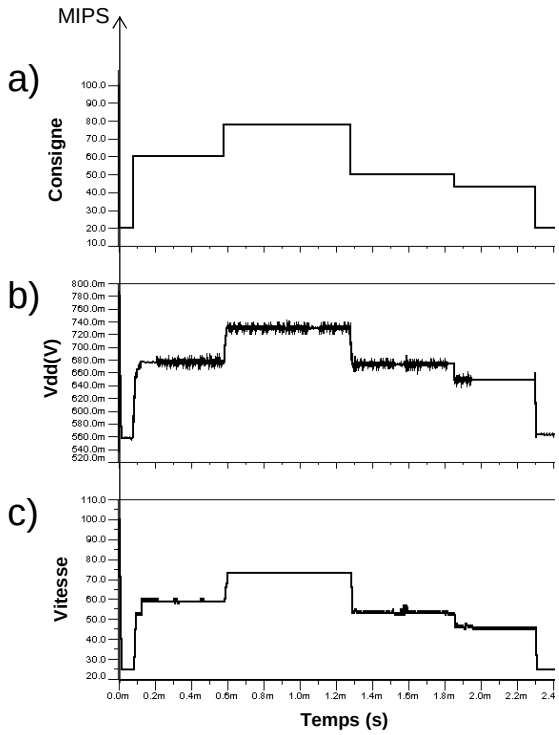


Figure 4-11 : Consigne, Tension en sortie du régulateur et vitesse du processeur

La Figure 4-12.a montre l'énergie consommée par le processeur sans contrôle. La courbe d'énergie pour cette simulation démontre bien que le processeur est inactif sur plusieurs périodes au cours desquelles le processeur ne consomme pas d'énergie, ceci peut être observé lorsque la courbe d'énergie présente des plateaux. L'énergie totale consommée pour le profil d'entrée est de 33µJ. La Figure 4-12.b présente l'énergie consommée par le système en appliquant le contrôle DVS, avec le régulateur à tension continue. Cette simulation délivre une énergie totale de 18µJ. Le contrôle DVS a permis d'obtenir un gain en énergie de 45%. Ceci montre bien que même si le processeur est inactif pendant certains instants durant l'exécution d'une tâche, augmenter le temps d'exécution des tâches avec une tension d'alimentation inférieure permet de réduire l'énergie consommée.

La consommation d'énergie de 18µJ correspond à la consommation du système qui inclut le processeur et le régulateur. La Figure 4-12.c montre l'énergie consommée par le processeur qui est de 10µJ. Le régulateur DC/DC a une consommation de 8µJ pour ce profil, ce qui correspond à un rendement de 55%. De manière générale, le régulateur choisi a un rendement qui varie en fonction de la tension de sortie. Plus la tension de sortie est faible, plus le rendement est mauvais. Inversement, plus la tension de sortie est élevée, plus le rendement est bon. Le régulateur a un rendement qui varie alors entre 45% et 70%. Ce rendement est très faible par rapport à d'autres régulateurs DC/DC plus performants.

Les trois simulations proposées dans la Figure 4-12, sont réalisées avec le même profil montré dans la Figure 4-11. Le temps d'exécution étant le même pour les trois simulations.

Bien que ce régulateur présente un grand avantage du fait qu'il puisse fournir une tension continue en sortie, avec un contrôle assez simple, son rendement et très inférieur au rendement attendu. Néanmoins, le gain obtenu avec ce régulateur n'est pas négligeable, et a permis de valider le système proposé. Avec un régulateur DC/DC à tension continue plus performant ayant un rendement de 85%, l'énergie consommée serait de l'ordre de 12µJ, ce qui représente un gain de 66% par rapport aux résultats obtenus avec le régulateur utilisé dans cette étude.

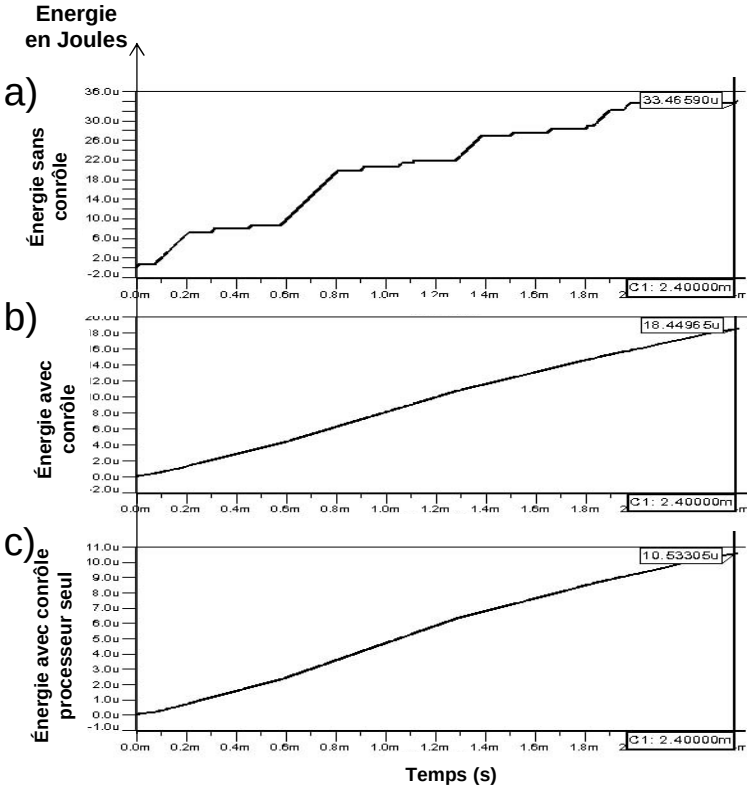


Figure 4-12 : Énergie consommée par :

a) Processeur a vitesse maximale sans DVS

b) Système avec DVS

c) Processeur avec DVS

6.2 Régulateur par paliers

Les régulateurs par paliers ont un meilleur rendement que les régulateurs à tension de sortie continue. Le système a été modifié pour contrôler deux régulateurs par palier. Le régulateur à tension continue a montré qu'avoir toute la plage de tension en sortie permet d'exécuter les tâches dans le

processeur à la vitesse minimale. Un régulateur par paliers ne pouvant pas fournir toute la plage de tension, il est nécessaire de choisir le nombre de paliers que le régulateur pourra générer. Il est évident que plus le nombre de paliers sera grand, plus le processeur pourra travailler à la vitesse minimale. Or, la complexité du régulateur va dépendre du nombre de paliers qu'il peut fournir. Ceci augmentera la complexité du contrôle nécessaire pour générer la commande en fonction du nombre de bits à générer. Deux régulateurs ont été choisis, le premier ayant une commande sur 4 bits pour être capable de générer 16 niveaux de tensions différentes. Le deuxième beaucoup moins complexe pourra fournir 4 niveaux de tension avec une commande sur 2 bits. Avec cette étude, le travail prétend déterminer le meilleur compromis entre le nombre de paliers que fournissent les régulateurs, la complexité et l'efficacité de la méthode.

6.2.1 Régulateur 4 bits

Le régulateur 4 bits choisi délivre une tension minimale de 0,4V lorsque le processeur est inactif et 15 tensions à un intervalle de 0,04V commençant par 0,54V. Le régulateur ne pouvant pas fournir la tension exacte pour avoir la vitesse minimale de fonctionnement, les tâches exécutées dans le processeur sont achevées avant la date d'échéance. Le processeur devra alors être inactif pendant quelques instants. La simulation de ce nouveau régulateur utilise le même profil que celui utilisé avec le régulateur continu. La Figure 4-13 montre la consigne utilisée et la vitesse du processeur obtenue après la simulation. Le régulateur ne fournissant pas la tension exacte pour répondre à la consigne demandée, le processeur doit être inactif pendant certains instants puisqu'il exécute les instructions plus vite que prévu.

Il est intéressant d'observer que le contrôle utilisé étant un contrôle proportionnel, il existe quelques surpassement de la vitesse voulue à certains instant lorsque la consigne change. Ce surpassement peut être corrigé en diminuant le paramètre P du contrôle proportionnel, ce qui rendra le système plus lent. Il est aussi possible d'utiliser un contrôle de type dérivatif-proportionnel, intragratif-proportionnel ou un mixte des trois pour diminuer les surpassements en maintenant la même vitesse de réponse. L'inconvénient de ces types de contrôle étant l'augmentation de la complexité du contrôle.

La simulation du système a donné une consommation de 10,85µJ. Cette consommation en revanche ne prend pas en compte la consommation du régulateur. Il est possible de trouver dans le marché des régulateurs commerciaux 4 bits ayant un rendement de 90%. Avec ce type de régulateur, la consommation du système serait de 12µJ.

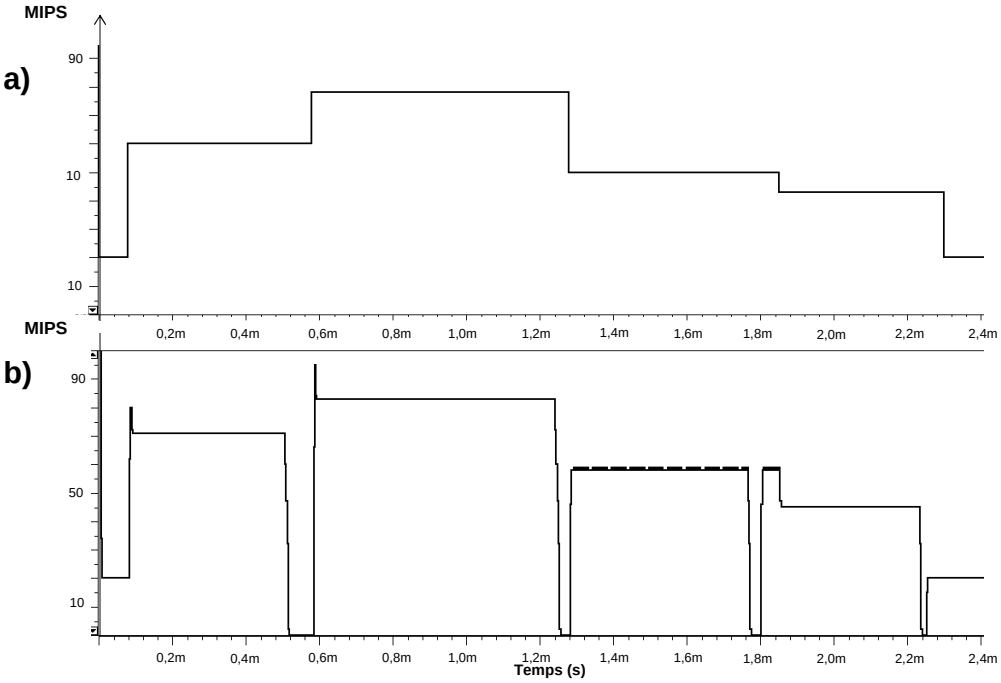


Figure 4-13 : consigne et vitesse du processeur pour le régulateur 4 bits

6.2.2 Régulateur 2 bits

En analysant l'énergie consommée par le régulateur 4 bits et le régulateur à tension continue, le surplus d'énergie consommée avec le régulateur 4 bits, n'est pas très grand. En prenant en compte seulement l'énergie consommée par le processeur sans celle des régulateurs, la différence est de 0.85µJ. Le prix à payer en ce qui concerne l'énergie consommée par le processeur est sûrement assez raisonnable puisque le régulateur 4 bits a un rendement supérieur au rendement du régulateur continu. Or, le contrôle nécessaire augmente en complexité puisque le co-processeur doit maintenant générer un signal sur 4 bits au lieu de 2 bits. Il est alors intéressant d'analyser le système avec un régulateur à paliers de 2 bits avec lequel le contrôle aurait environ la même complexité que le régulateur continu. Le régulateur choisi fournit quatre tensions de sortie de 0,4V, 0,66V, 0,93V et 1,2V.

La même consigne est appliquée pour le système avec le régulateur 2 bits. Cette consigne est supposée être calculée statiquement. La Figure 4-14 montre la consigne appliquée ainsi que la vitesse obtenue après la simulation. Le système donne la même réponse que le régulateur 4 bits. Avec ce régulateur, le temps d'inactivité est en revanche beaucoup plus élevé. En effet, la vitesse du processeur est plus élevée et en conséquence le processeur termine l'exécution des tâches plus rapidement.

Avec ce régulateur et la consigne donnée, le processeur consomme 15,49µJ. Cette énergie est mesurée sans prendre en compte la consommation du régulateur, qui devrait avoir un rendement d'environ 95%. Avec un tel régulateur, le système aurait une consommation de 16,3µJ. L'énergie

consommée par le processeur n'est plus négligeable par rapport au régulateur à tension continue. Elle reste néanmoins inférieure lorsque la consommation du système complet est prise en compte.

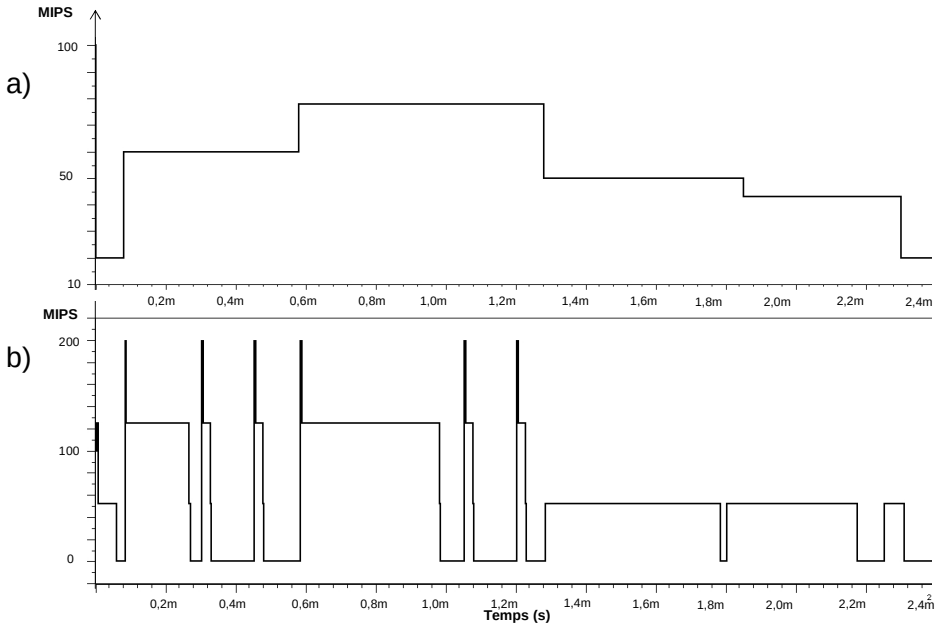


Figure 4-14 : Consigne et vitesse obtenue du processeur avec le régulateur 2 bits

6.2.3 Régulateur 2 bits avec calcul en temps réel de la consigne.

En analysant les résultats obtenus avec le régulateur 2 bits, on observe qu'il serait possible de générer une consigne en temps réel comme celle montrée dans la Figure 4-9 pour diminuer les temps d'inactivité du processeur. La consigne est alors calculée en fonction des instructions nécessaires pour finir l'application en temps réel. La vitesse obtenue est montrée dans la Figure 4-15. L'énergie consommée par le processeur après simulation est de 12,41µJ. Si le rendement du régulateur 4 bits est de 95%, l'énergie consommée par le système serait de 13µJ. Cette simulation montre que la gestion du calcul de la consigne est très importante pour diminuer la consommation du système. Cependant, il faudra faire attention à l'augmentation de la complexité de la génération de cette consigne, puisqu'elle est calculée en temps réel. La consommation générée par ce calcul n'est pas prise en compte dans ce travail, mais calculer en temps réel la consigne augmentera de 5 à 10% le nombre d'instructions à exécuter dans le processeur [BUR 08].

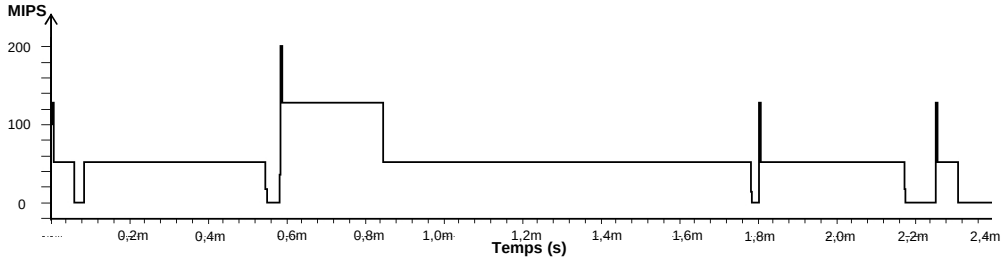


Figure 4-15 : Vitesse avec calcul en temps réel de la consigne.

7 Comparaison de la puissance consommée

La Figure 4-16 montre la consommation d'énergie pour les différents types de régulateur par palier ainsi que pour les différentes gestions de la génération de la consigne utilisée. Cette courbe montre clairement les périodes d'inactivité du processeur qui se traduisent par des plateaux dans les courbes lorsque le contrôle n'est pas appliqué. On observe aussi comment la durée des plateaux est réduite avec les régulateurs 2 bits et 4 bits. Pour le contrôle avec gestion en temps réel de la consigne, le processeur n'est pratiquement jamais inactif. La consommation obtenue lors de cette simulation est similaire à la consommation obtenue avec le régulateur 4 bits.

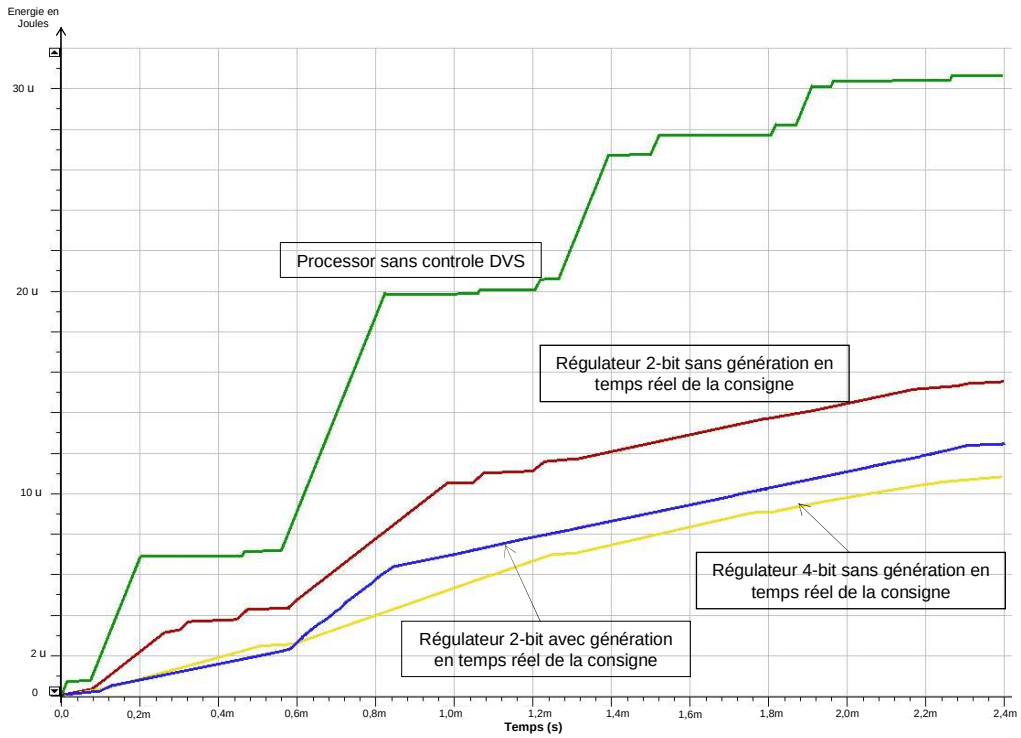


Figure 4-16 : Consommation du processeur avec les différentes architectures de régulateurs

8 Conclusions

Ce travail présente un co-processeur qui contrôle un régulateur pour faire varier la tension d'alimentation d'un processeur. Il présente aussi un environnement de simulation et des résultats obtenus après simulation avec un environnement mixte.

La Figure 4-11, la Figure 4-13, et la Figure 4-15 montrent qu'il est possible de contrôler de façon efficace la vitesse du processeur avec une consigne donnée.

La Figure 4-12 et la Figure 4-16 présentent l'énergie consommée par le système pour les différentes architectures et le profil d'application donné. Ces courbes montrent bien que la technique DVS réduit de façon considérable l'énergie consommée par le processeur.

En ce qui concerne la régulation de la vitesse, le régulateur à tension de sortie continue implémenté avec une pompe de charge donne la meilleure régulation de la vitesse du processeur. Ce régulateur permet d'obtenir la consommation du processeur seul qui est la plus optimale pour le profil d'application utilisée. Dans le cas des régulateurs par paliers, la vitesse optimale ne peut pas être atteinte. La différence entre la vitesse réelle du processeur et celle de la consigne varie en fonction du nombre de paliers mais aussi en fonction de la vitesse souhaitée. En effet, dans le cas idéal, pour un régulateur par paliers, la consigne correspond aux tensions d'alimentation que le régulateur est capable de fournir. Pour le profil analysé, la différence de consommation du processeur seul entre le régulateur à tension continue et les régulateurs à quatre bits, deux bits et deux bits avec gestion dynamique de la consigne sont respectivement de 8,4%, 35% et 24,1%.

Néanmoins, la consommation du régulateur doit aussi être prise en compte. Pour le régulateur à tension continue choisi dans cette étude, le rendement étant de 45% à 70% en fonction de la tension de sortie, il rend la consommation du système complet supérieure à la consommation d'énergie avec les régulateurs par paliers. Dans cette étude, le système qui présente la meilleure consommation étant le régulateur par paliers quatre bits avec un gain de 63,6% par rapport à l'énergie consommée par le système sans contrôle.

Le choix du nombre de paliers dans le régulateur est très important. Il détermine la précision du contrôle de la tension d'alimentation du système, ce qui entraîne une réduction de la consommation d'énergie. Or, plus le nombre de paliers est grand, plus le régulateur sera complexe. La complexité du contrôle du régulateur augmente aussi lorsque le nombre de paliers augmente. Il est alors important de définir le degré de précision souhaité pour le contrôle de la tension. Cette étude montre aussi que la différence entre le gain obtenu avec un régulateur quatre bits par rapport au gain obtenu avec le régulateur deux bits est de 13% pour le profil utilisé. Cependant, le contrôle pour le régulateur quatre

bits sera beaucoup plus complexe que celui du régulateur deux bits. Il faut aussi prendre en compte que le gain en consommation est en fonction du profil d'entré. Plus le processeur est inactif pour un profil donné, plus le gain en consommation sera grand. Inversement, si les tâches qui doivent être exécutées utilisent le processeur sans laisser des temps d'inactivité, la consommation sera pratiquement la même avec ou sans contrôle. L'étude présentée s'applique bien aux applications dans les réseaux de capteurs.

Un autre facteur est l'augmentation de la taille du circuit. Cette augmentation dépend du nombre de paliers que le régulateur peut fournir mais aussi du co-processeur nécessaire pour générer le contrôle.

Le travail présenté montre aussi l'importance de la génération de la consigne pour contrôler la vitesse du processeur. La première étude montre une méthode simple pour calculer la consigne afin de répondre aux besoins des tâches qui arrivent dans le processeur. Avec cette méthode, lorsqu'une tâche arrive, elle indique la vitesse nécessaire pour être exécutée avant sa date limite d'exécution. Le système d'exploitation calcule alors la nouvelle consigne qui est la somme des vitesses dans la queue plus la nouvelle vitesse. Lorsque la tâche est exécutée, la vitesse est retirée de la consigne précédente. Avec un régulateur à tension continue, il est possible de contrôler la vitesse du processeur pour que les tâches soient exécutées dans le temps voulu. L'utilisation d'un régulateur par paliers ne donnant pas la tension exacte nécessaire, le palier choisi lorsqu'une nouvelle tâche arrive à la queue d'exécution doit être supérieure à la vitesse souhaitée. Avec cette méthode de génération de la consigne, le processeur exécutera les instructions à une vitesse plus élevée. Les tâches finiront en avance et le processeur sera inactif de manière similaire au fonctionnement du processeur sans contrôle. Le temps d'inactivité est évidemment réduit, mais la consommation optimale d'énergie n'est pas obtenue.

Une deuxième méthode pour la génération de la consigne montre qu'il est envisageable de réduire la vitesse du processeur dès que possible. Ceci est réalisable puisque le processeur exécute les instructions à une vitesse supérieure et il sera possible de réduire la vitesse à un certain moment de l'exécution. Les avantages de cette méthode sont montrés dans la Figure 4-14 et la Figure 4-15, les deux figures montrent la vitesse du processeur avec le même régulateur par paliers à deux bits. Cette méthode permet une réduction de 20% de la consommation de l'énergie avec le même régulateur mais en appliquant une génération de la consigne différente. Néanmoins, l'énergie utilisée pour le calcul de la consigne en temps réel n'est pas prise en compte. Une étude plus approfondie sera nécessaire pour cela.

Conclusion

Malgré tous les avantages que présentent les circuits asynchrones, ils en sont encore au stade de développement. L'industrie ainsi que la recherche se sont focalisées jusqu'à présent sur le développement des circuits synchrones. Depuis la fin des années quatre vingt, plusieurs équipes de recherche ce sont tournées vers les circuits asynchrones. Actuellement il existe plusieurs microprocesseurs asynchrones qui ont été développés aussi bien par des groupes de recherche mais aussi par des entreprises privées comme Philips, Sharp et Epson. Les circuits asynchrones commencent peu à peu à être présent dans notre vie. Mais le manque de méthodologies ainsi que le manque d'outils pour la conception de ces circuits, freine le développement de ces derniers.

Les travaux de cette thèse ont pour objectif d'apporter une contribution à la conception des circuits asynchrones QDI. Spécifiquement, dans le domaine de la consommation d'énergie. Dans la première partie du manuscrit, une analyse des quatre sources de consommation d'énergie principales dans les circuits CMOS est présentée. Par la suite le manuscrit présente les différentes méthodologies existantes pour estimer la consommation des circuits. Deux grandes lignes peuvent être constatées. Les méthodes par simulation et les méthodes par non simulation. Le premier groupe de méthodes délivre généralement des résultats très fiables. Elles nécessitent en revanche d'avoir un modèle du circuit analysé très détaillé. Les temps de simulation sont en général élevés et dépendent des ressources matérielles utilisées lors de la simulation. Au contraire, les méthodes par non simulation offrent plusieurs avantages en terme de vitesse et de ressources matérielles nécessaire pour l'estimation de l'énergie consommée. La contre-partie de ces techniques est qu'elles ne peuvent pas donner une estimation aussi précise que les méthodes par simulation. Néanmoins, elles arrivent à donner des résultats avec une erreur qui peut être inférieure à 5%. Par la suite, le manuscrit présente quelques techniques pour réduire la consommation dynamique des circuits CMOS. Les techniques présentées utilisent l'Équation 4 de la consommation d'énergie présentée dans le Chapitre 1. Chacune

de ces techniques améliore un des paramètres qui sont présents dans l'équation. Finalement, le premier chapitre, présente l'évolution des microprocesseurs asynchrones qui ont été conçus jusqu'à nos jours.

Dans le deuxième chapitre, ce travail propose une méthodologie de conception des circuits asynchrones QDI. Cette méthodologie utilise trois logiciels qui ont été conçus dans le groupe de travail CIS du laboratoire TIMA. Cette méthodologie permet de synthétiser des circuits asynchrones QDI depuis le langage de haut niveau CHP avec l'outil TAST. Le flot de conception présenté dans cette méthodologie, a l'avantage d'être entièrement automatique. Il permet d'obtenir un circuit asynchrone QDI en portes standard à deux entrées dans un premier temps. Le deuxième outil permet d'obtenir un profil d'activité du circuit en comptant les transitions réalisées avec un vecteur d'entrée donné. Cette première estimation permet d'identifier les parties les plus actives du circuit. Les premières optimisations ou choix d'architectures peuvent être faites dans cette étape. Le troisième outil permet de réaliser la projection technologique. L'outil utilise la librairie TAL qui est une librairie de cellules asynchrones développée dans le groupe CIS. Il est possible par la suite de réaliser le placement routage du circuit avec les outils du marché comme SOC Encounter. Une fois réalisé cette étape, l'outil qui compte les transitions permet de pondérer ces dernières avec la capacité de charge de la sortie de chaque portes. Cette capacité est obtenue après le placement routage du circuit.

La première partie du troisième chapitre montre l'utilisation de la méthodologie proposée. Il présente deux sous blocs de l'ALU du proceseur MIPS 32 4ksc. La première est le bloc logique qui effectue les opérations AND, OR, XOR et NOR. Le deuxième est le bloc de décalage qui lui exécute les instructions de décalage et les instructions de rotation.

De part les résultats obtenus avec le bloc logique, ce travail montre l'importance de l'ordonnancement des entrées lors de la génération du format intermédiaire MDD. Il est possible dans cet exemple de réduire la consommation de 10,5%. La taille du circuit étant réduite de 17,8% lorsque les entrées sont ordonnancées de manière différente. L'utilisation de la librairie asynchrone TAL permet de réduire la taille du circuit en moyenne de 35%. L'équivalent synchrone de ce bloc est 4,7 fois plus petit que le circuit asynchrone. Ce résultat était prévisible puisque les circuits synchrones restent inférieurs en taille à cause du codage de données. Ce travail montre bien l'impact sur la consommation de l'arbre d'horloge dans un circuit synchrone. Pour le bloc logique montré, l'augmentation de la taille des opérandes, qui forcément font augmenter la taille du circuit et la complexité de l'arbre d'horloge, fait augmenter de façon quasi quadratique l'énergie consommée.

En ce qui concerne le bloc de décalage, ce travail montre que regrouper les blocs en blocs plus complexes permet de réduire la consommation d'énergie. Or la taille du bloc de décalage a augmentée de 13,4% par rapport à la version à 5 étages. L'équivalent synchrone du bloc de décalage, consomme

28% plus d'énergie. Il est, comme pour le bloc logique, plus petit en taille. Avec l'unité de décalage, ce travail montre aussi l'importance des vecteurs d'entrées. En effet, le circuit asynchrone présente un avantage en terme de consommation, puisque celle-ci est indépendante des entrées. Contrairement au circuit synchrone où les entrées ont un impact sur la consommation d'énergie. Pour quantifier ceci, une étude exhaustive a été réalisée en modifiant la distance de Hamming entre les vecteurs d'entrée du circuit. La consommation du circuit asynchrone reste constante lorsque la distance de Hamming varie de 0 jusqu'à 32. Pour le circuit synchrone, la consommation a un comportement linéaire lorsque cette distance augmente.

Le deuxième phénomène intéressant de cette étude est le pic de transition qui pour le circuit asynchrone est constant sur toutes les distances de Hamming. Dans les résultats montrés, le pic de transition est le pire cas pour le circuit asynchrone puisque les entrées arrivent en même temps. Dans le circuit synchrone, le pic de transition augmente jusqu'à une valeur maximum. Dans le pire des cas pour le circuit asynchrone, le pic de transitions est deux fois plus petit que pour le circuit synchrone dans le meilleur des cas.

Finalement dans le troisième chapitre, ce travail présente des choix d'architecture tel que la manière dont est généré le signal d'acquittement, la suppression des signaux de commande en utilisant des sorties passives et des entrées actives et finalement l'importance de connaître les probabilités d'utilisation des chemins de données.

Le quatrième chapitre présente une technique de réduction d'énergie en faisant varier la tension d'alimentation du circuit. Cette technique est appelée DVS (Dynamic Voltage Scaling). Dans le travail présenté, l'idée novatrice est de contrôler à l'aide d'un coprocesseur la tension d'alimentation. Ce contrôle se réalise en boucle fermée, et le contrôle utilisé est un contrôle proportionnel. L'étude présente les résultats obtenus avec différents régulateurs DC/DC. Le premier étant un régulateur DC/DC à tension continue. Ce régulateur permet d'avoir toute la plage de tensions souhaitées. Il a le désavantage d'avoir une efficacité très mauvaise. Le gain en énergie en utilisant ce régulateur est de 45%. Les deux autres régulateurs utilisés sont des régulateurs par palier. Ces régulateurs fournissent un nombre fini de tensions en sortie. Or l'étude montre que bien que la tension optimale d'alimentation ne soit pas utilisée dans le cas des régulateurs par palier, il est possible d'obtenir un gain en consommation de 63,6%. Ce gain est supérieur par rapport au régulateur à tension continue à cause de l'efficacité de ce dernier. Finalement l'étude présente l'importance du calcul de la consigne de la vitesse du processeur lors de l'exécution d'une application. Cette consigne peut être calculée de manière statique ou en temps réel. Dans l'exemple montré, le processeur avec le régulateur à quatre paliers, consomme quasiment la même énergie que le processeur avec le régulateur à seize paliers lorsque la consigne est calculée en temps réel.

Les travaux futurs de cette thèse seront d'estimer la consommation statique des circuits. Cette consommation commence à être comparable à la consommation dynamique dans les circuits CMOS actuel. L'outil pour estimer l'énergie dynamique doit aussi prendre en compte les courants de court circuit générés lors d'une transition à la sortie d'une porte. Il serait aussi intéressant de comparer des circuits beaucoup plus grand pour obtenir dans le cas synchrone l'arbre d'horloge conséquent aux circuits de grande taille. En ce qui concerne les circuits asynchrones, une étude plus approfondie sur le codage de données utilisé devra être mise en place. Le codage en quatre rails est moins consommant à priori en terme de transitions. Mais il a le désavantage de donner un circuit plus grand en taille, ce qui implique une consommation statique plus élevée.

Les circuits synchrones continuent à être toujours plus performants en taille, mais les circuits asynchrones ont une consommation inférieure. Bien que la librairie TAL de cellules asynchrones utilisée pour la synthèse des circuits étudiés dans ce travail, réduit considérablement la taille, il est nécessaire d'optimiser la librairie de cellules en ce qui concerne la taille. Il est nécessaire aussi d'optimiser les méthodes de synthèse pour obtenir des circuits plus performants. La recherche sur la synthèse de circuits asynchrones reste encore ouverte pour obtenir des circuits asynchrones qui soient comparables avec les circuits synchrones dans tous les paramètres de comparaison (taille, consommation, vitesse).

Finalement, dans la méthode DVS présentée il reste certains points qui n'ont pas été pris en compte, comme le coût en énergie du coprocesseur. Ce dernier ne devrait pas être très consommant puisque la fréquence d'opération n'est pas élevée. En effet le calcul de la nouvelle commande du régulateur se réalise à une fréquence de 32Khz. Dans le système étudié, le coprocesseur réalise en moyenne un calcul toutes les 1000 instructions réalisées par le processeur. Il reste néanmoins nécessaire de vérifier la consommation du coprocesseur.

Publications

Conférences :

- D. Rios-Arambula , Buhrig A., Renaudin M. “Asservissement de vitesse pour minimiser la puissance consommée par un processeur”, in FTFC conference, 18-20 may, Paris, France 2005.

- D. Rios-Arambula, A. Buhrig and M. Renaudin, “Power Consumption reduction using dynamic control of Micro Processor performance”, PATMOS 2005, 20-23 september, 2005, Leuven, Belgium

- D. Rios-Arambula, Aurélien Buhrig et Marc Renaudin, “Comparaison de régulateurs par palier 2 et 4 bits dans un système d’asservissement de la vitesse d’un microprocesseur pour réduire la consommation d’énergie”, JNRDM, Mai 2006

- D. Rios-Arambula, B. Folco, Y. Monnet, M. Renaudin, “Analyse du profil de consommation des circuits asynchrones QDI”, in FTFC conference, 21-23 Mai, Paris, France 2007.

Journaux :

-D. Rios-Arambula, A. Buhrig, G. Sicard et M. Renaudin, “On the use of Feedback Systems to Dynamically Control the Supply Voltage of Low-Power Circuits”, Journal of Low Power Electronics, vol. 2, p. 45-55. 2006

Bibliographie

[ABR 01] A. Abrial, P. Senn ,M. Renaudin and P. Vivet. “A new contactless smart card ic using on-chip antenna and asynchrounous microcontroller”. Journal of Solid-State Circuits, 36:1101–1107, 2001

[ATH 94] W.C.Athas, L. J. Svensson, J.G.Koller, N.Thartzanis and E. Chou. " Low-power digital systems based on adiabatic-switching principles. " IEEE Transactions on VLSI Systems, 2(4)398-407:, December 1994.

[BAB 00] Babu (H.), Md (H.) et Sasao (T.), Heuristics to minimize multiple-valued decision diagrams, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2000, p. 2498 2504.

[BAR 07] Bardsley (A.) et Edwards (D.), “Compiling the language Balsa to delay insensitive hardware”, Hardware Description Languages and their Applications (CHDL), 1997, p. 89-91.

[BEN 96] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, “Regression models for behavioral power estimation,” in Proc. Int. Wkshp. Power & Timing Modeling, Optimization, and Simulation (PATMOS), 1996

[BEN 97] L. Benini, G. De Micheli, Dynamic Power Management: Design Techniques and CADTools, Kluwer Academic Publishers, 1997

[BIS 96] L. Bisdounis, S. Nikolaidis, O. Koufopavlou, C. E. Goutis, Modeling the CMOS short-circuit power dissipation, in Proc. of ISCAS, May 1996.

[BOG 98a] A. Bogliolo and L. Benini, “Robust RTL power macromodels,” IEEE Trans. VLSI Systems, vol. 6, pp. 578–581, Dec. 1998.

[BOG 98b] A. Bogliolo, L. Benini, and G. De Micheli, “Characterization-free behavioral power modeling,” in Proc. Design Automation & Test Europe (DATE) Conf., pp. 767–773, Mar. 1998.

[BRE 07] V. Bregier, “Automatic synthesis of Asynchronous Proven Quasi Delay Insensitive Circuits”, PHD thesis, Institut National Polytechnique Grenoble, 2007

[BUR 93] R. Burch, F. N. Najm, P. Yang, and T. Trick. " A Monte Carlo approach for power estimation. " IEEE Transactions on VLSI Systems, 1(1):63Ð71, March 1993.

[BUR 95] T. Burd and R. Brodersen, “Energy Efficient CMOS Microprocessor Design”, Proceedings of the Hawaii International Conference on System Sciences, pp.288-297, 1995

[BUY 01] K.M. Buyuksahin and F. N. Najm, “High-level power estimation with interconnect effects,” in Proc. Int. Symp. Low Power Electronics & Design, pp. 197–202, Aug. 2001.

[CHA 92a] A. Chandrakasan, et al. “Low-power CMOS digital design.” IEEE Journal of Solid-state Circuits Vol 27 pg 473-484. April 1992.

[CHA 92b] A.P. Chandrakasan, M. Potkonjak, J. Rabaey, R. Brodersen, "An Approach for Power Minimization Using Transformations", 1992 IEEE Workshop on VLSI Signal Processing, Napa Valley, CA, pp. 500-503, October 1992.

[CHA 92b] A. Chandrakasan, M. Potkonjak, J. Rabaey and R. W. Brodersen. " HYPER-LP: A System for Power Minimization Using Architectural Transformation. " In Proceedings of the IEEE International Conference on Computer Aided Design, pages 300-303, November 1992.

[CHA 95a] J-M. Chang and M. Pedram. " Low power register allocation and binding. " In Proceedings of the 32nd Design Automation Conference, pages 29-35, June 1995.

[CHA 95b] J-M. Chang and M. Pedram. " Power efficient module allocation and binding. " CENG Technical Report 95-16, University of Southern California, June 1995.

[CHAN 95] A. R. Chandrakasan, R. W. Brodersen, Low Power Digital CMOS Design, KluwerAcademic Publishers, 1995

[CHE 98] C.-H. Chen and C.-Y. Tsui, “Towards the capability of providing power-area-delay tradeoff at the register transfer level,” in Proc. Int. Symp. Low Power Electronics & Design, pp. 24–29, Aug. 1998.

[CHO 95] T-L. Chou and K. Roy. " Statistical estimation of sequential circuit activity. " In Proceedings of the IEEE International Conference on Computer Aided Design, pages 34-37, November 1995.

[CIR 87] M. A. Cirit, “Estimating dynamic power consumption of CMOS cucuits,” IEEE Int. Con5 on Computer-Aided Design. pp. 534-537, Nov. 9-12, 1987.

[CME 91] R.F. Cmelik, S.I. Kong, D.R. Ditzel and E.J. Kelly, "An Analysis of MIPS and SPARC instruction set utilization on the SPEC benchmarks", In ASPLOS-IV Proceedings, SIGARCH Computer Architecture News 19, pp 290-302, 2 April 1991.

[DAV 95] B. Davari, R. H. Dennard and G. G. Shahidi. " CMOS scaling for high performance and low power. " Proceedings of IEEE, 83(4):408-425, April 1995.

[DIN 02a] A.V. Dinh Duc, J.B. Rigaud, et al. “TAST CAD Tools”, in Asynchronous Circuit Design, Munich, Germany. 2002.

[DIN 02b] A.-V. Dinh-Duc, L. Fesquet, and M. Renaudin. “Synthesis of QDI Asynchronous Circuits from DTL-style Petri Nets”, in 11th IEEE/ACM International Workshop on Logic and Synthesis. 2002.

[DIN 03] A.-V. Dinh-Duc, “Synthèse automatique de circuits asynchrones QDI”, research repport, INP of Grenoble, 2003.

[EBE 91] J. Ebergen, "A formal approach to designing delay-insensitive circuits", Distributed Computing, Vol. 5, N°. 3, pp. 107-119, July, 1991.

[EDW 02] Edwards (D.) et Bardsley (A.), “Balsa : An Asynchronous Hardware Synthesis Language ”, The Computer Journal, vol. 45, no 1, 2002, p. 12.

[EMN 00] F. Emnett and M. Biegel, “Power Reduction Through RTL Clock Gating”, Proceedings of the Synopsys User Group (2000), San Jose.

[FLA 01a] K. Flautner, “Automatic Monitoring for Interactive Performance and Power Reduction”, Dissertation, Michigan University, 2001.

[FLA 01b] K. Flautner, S. K. Reinhardt, and T. N. Mudge, “Automatic performance setting for dynamic voltage scaling”, In Mobile Computing and Networking, pages 260–271, 2001.

[FOL 07] B. Folco, “Contribution to Synthesis of Asynchronous Quasi Delay Insensitive Circuits, Application to Secured Systems”, PHD thesis, Institut National Polytechnique Grenoble, 2007

[FUR 89] S.B. Furber, VLSI RISC Architecture and Organisation. New York: Marcel Dekker Inc., 1989.

[FUR 94] S.B. Furber, P. Day, J.D. Garside, N.C. Paver, and J. V. Woods, “AMULET1: A micropipelined ARM”, Proceedings of IEEE Computer Conference (COMPCOM), Mar. 1994. pp.476-485

[FUR 98] S.B. Furber, J.D. Garside and S. Temple, “Power-saving features in Amulet2e”, Power Driven Microarchitectures Workshop, Jun. 1998.

[FUR 00] S.B. Furber, D.A. Edwards, and J.D. Garside, “AMULET3: a 100 MIPS Asynchronous Embedded Processor”, Proceedings of International Conference Computer Design (ICCD), Sep. 2000.

[GAG 98] H. Gageldonk, D. Baumann, K Berkel, D. Gloor, A. Peeters and G. Stegmann, “An asynchronous low-power 80c51 microcontroller”, Proceeding of International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1998. Pp. 96-107

[GEO 94] B. J. George, D. Gossain, S. C. Tyler, M. G. Wloka, and G. K. H. Yeap. " Power analysis and characterization for semi-custom design. " In Proceedings of the 1994 International Workshop on Low Power Design, pages 215Ð218, April 1994.

[GOV 95] K. Govil, E. Chan, and H.Wasserman. Comparing algorithm for dynamic speed-setting of a low-power CPU. In Mobile Computing and Networking, pages 13–25, 1995.

[GRU 00] D. Grunwald, P. Levis, K. I. Farkas, C. B. Morrey III, and M. Neufeld. Policies for dynamic clock scheduling. In Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI), pages 73–86, 2000.

[GUP 00] S. Gupta and F. N. Najm, “Analytical models for RTL power estimation of combinational and sequential circuits,” IEEE Trans. Computer- Aided Design, vol. 19, pp. 808–814, July 2000.

[HAU 95] S. Hauck, "Asynchronous Design Methodologies : An Overview", Proceeding of the IEEE, Vol. 83, N° 1, pp. 69-93, January, 1995.

[HAM 50] Hamming, R. W., "Error detecting and error correcting codes", Bell System Tech. J., vol. 29, pp. 147-160, 1950

[HAU 95] S. Hauck, "Asynchronous Design Methodologies : An Overview", Proceeding of the IEEE, Vol. 83, N° 1, pp. 69-93, January, 1995.

[HED 87] N. Hedenstierna and K. Jeppson. " CMOS circuit speed and buffer optimization. " IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 6(3):270-281, March 1987.

[HED 93] N. Hedenstierna and K. Jeppson. “Comments on A module generator for optimized CMOS buffers”, IEEE Trans. Computer-Aided Design, vol.12, pp. 180-181, Jan. 1993.

[HOA 78] C. A. R. Hoare, “Communicating sequential processes”, Communications of the ACM, v.21 n.8, p.666-677, Aug. 1978

[HUA 95] C. X. Huang, B. Zhang, A-C. Deng and B. Swirski. " The design and implementation of PowerMill. " In Proceedings of the 1995 International Symposium on Low Power Design, pages 105-110, April 1995.

[IBM 07] IBM PowerPC. http://www.chips.ibm.com/products/powerpc/

[INT 08] Intel XScale. http://www.intel.com/design/intelxscale/

[KAM 98] Kam (T.), Villa (T.), Brayton (R.) et Sangiovanni-Vincentelli (A.), Multi-valued decision diagrams : theory and applications, Multiple-Valued Logic, vol. 4, no 1-2, 1998, p. 962.

[KAN 86] S. M. Kang. " Accurate simulation of power dissipation in VLSI circuits. " IEEE Journal of Solid State Circuits, 21(5):889Ð891, October 1986.

[KAR 05] Karaki, N.; Nanmoto, T.; Ebihara, H.; Utsunomiya, S.; Inoue, S.; Shimoda, T., “A flexible 8b asynchronous microprocessor based on low-temperature poly-silicon TFT technology”, Solid-State Circuits Conference, 2005. ISSCC. 2005 IEEE International, Feb. 2005 Page(s): 272 - 598 Vol. 1

[KOB 94] T. Kobayashi and T. Sakurai. " Self-adjusting threshold-voltage scheme for low voltage high speed operation. " Proceedings of CICC, pages 271-274, May 1994.

[KOM 89] S. Komori, K. Shima, S. Miyata, T. Okamoto, and H. Terada, “The data-driven microprocessor,” IEEE Micro. Mag., vol. 9, pp. 45–59, June 1989.

[KUD 94] P. Kudva and V. Akella, “A technique for estimating power in asynchronous circuits,” in Proc. Znt. Symp. Adv. Res. in Asynchronous Circuits Syst., 1994, pp. 166-175.

[LAN 93] P.E. Landman and J. Rabaey. " Power estimation for high level synthesis. " In Proceedings of the European Conference on Design Automation, pages 361-366, February 1993.

[LAN 95] P. Landman and J. M. Rabaey, “Architectural power analysis: The dual bit type method,” IEEE Trans. VLSI Systems, vol. 3, pp. 173– 187, June 1995.

[LEN 95] C. Lennard and A. R. Newton. " An estimation technique to guide low power resynthesis algorithms. " In Proceedings of the 1995 International Symposium on Low Power Design, pages 227-232, April 1995.

[LLO 98] R. P. Llopis and F. Goossens, “The Petrol approach to high-level power estimation,” in Proc. Int. Symp. Low Power Electronics & Design, pp. 130–132, Aug. 1998.

[LOR 01] J. R. Lorch and A. J. Smith. Improving dynamic voltage scaling algorithms with PACE. In SIGMETRICS/Performance, pages 50–61, 2001.

[MAN 96] Manohar (R.) et Martin (A. J.). “Quasi-delay-insensitive circuits are turing-complete” , janvier 1996.

[MAR 89a] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus. “The design of an asynchronous microprocessor”. In C. L. Seitz, editor, Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI, pages 351–373. MIT Press, 1989.

[MAR 89b] A.J. Martin, S.M. Burns, T.K. Lee, D. Borkovic, and P.J. Hazewindus “The First Asynchronous Microprocessor: The Test Results”, Computer Architecture News, 17(4), 95 110, June 1989.

[MAR 89c] Alain J. Martin, “Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits”, California Institute of Technology, Pasadena, CA, 1989

[MAR 90] Martin (A. J.). “The limitations to delay-insensitivity in asynchronous circuits. sixth mit conference on advanced research in vlsi, ed. wj dally”, 1990.

[MAR 93] Alain J. Martin. Synthesis of Asynchronous VLSI Circuits. Internal Report. Caltech-CS-TR-93-28. California Institute of Technology, Pasadena, CA. 1993.

[MAR 95] D. Marculescu, R. Marculescu, and M. Pedram, “Information theoretic measures for energy consumption at the register-transfer level,” in Proc. Int. Symp. Low Power Design, pp. 81–86, Apr. 1995.

[MAR 97] A.J. Martin, A. Lines, R. Manohar, M. Nystroem, P. Penzes, R. Southworth, and U. Cummings. “The design of an asynchronous MIPS R3000 microprocessor”. In Advanced Research in VLSI, September 1997

[MAR 00] A. Martin, “An asynchronous approach to energy-efficient computing and communication”, Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (2000).

[MAR 01a] Martin, Alain J. and Nystroem, Mika and Penzes, Paul and Wong, Catherine, “An Asynchronous Microprocessor in Gallium Arsenide” 01 January 2001, Monograph.

[MAR 01b] A. J. Martin, M. Nyström, P. Pénzes, and C. Wong, “Speed and energy performance of an asynchronous MIPS R3000 microprocessor”, Tech. Rep. CSTR:2001.012, California Institute of Technology, 2001.

[MART 01] T. L. Martin. “Balancing Batteries, Power, and Performance: System Issues in CPU Speed-Setting for Mobile Computing”, PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2001.

[MAR 03] A. J. Martin, M. Nystrom, et al. “The Lutonium: sub-nanojoule asynchronous 8051 microcontroller”. In Proc. of the Ninth Int’l. Symposium on Asynchronous Circuits and Systems, pages 14–23, May 2003.

[MAT 07] http://www.mathworks.com/

[MEI 00] J. D. Meindl and J. A. Davis, “The fundamental limit on binary switching energy for terascale integration (TSI),” IEEE J. Solid-State Circuits, vol. 35, pp. 1515-1516, Oct. 2000.

[MIP 07] http://www.mips.com

[MIY 02] M. Miyazaki, J. Kao, A. Chandrakasan, “A 175mV Multiply-Accumulate Unit using an Adaptive Supply Voltage and Body Bias (ASB) Architecture”, ISSCC 2002, pp. 58-59

[MOL 99] F. Møller, “Algorithm and architecture of a 1-V low-power hearing instrument DSP”, in Proc. ISLPED, pp. 711, Aug. 1999.

[MON 95] J. Monteiro, J. Rinderknecht, S. Devadas and A. Ghosh. " Optimization of combinational and sequential logic circuits for low power using precomputation. " In Proceedings of the 1995 Chapel Hill Conference on Advanced Research on VLSI, pages 430-444, March 1995.

[MOO 65] G. Moore, Cramming more components onto integrated circuits, Electronics Vol 38 n° 8, 19 avril 1965, pp. 114-117

[MUL 65] R.E. Muller, “Sequential circuits”, Chapter 10, Switching theory, Vol 2, N.Y. Wiley, 1965

[MUL 91] K. D. Muller-Glaser, K. Kirsch, and K. Neusinger, “Estimating essential design characteristics to support project planning for ASIC design management,” in Proc. Int. Conf. Computer-Aided Design, pp. 148–151, Nov. 1991.

[PEN 02] P. Pénzes and A. J. Martin. “Energy-delay efficiency of vlsi computations”. In Proc. of the Great Lakes Symposium on VLSI, 2002.

[PID 07] http://www.pidlab.com/

[PIG 04] C. Piguet, “Low-Power Electronics Design”, CRC Press, USA, 2004

[NAJ 95] F. N. Najm, S. Goel and I. Hajj. " Power estimation in sequential circuits. " In Proceedings of the 32nd Design Automation Conference, pages 635-640, June 1995.

[NAJ 91] F. Najm, “Transition density, a stochastic measure of activity in digital circuits,” 28th ACWIEEE Design Automation Con$. San Francisco, CA, pp. -9, June 17-21, 1991

[NEM 99] M. Nemani and F. Najm, “High-level area and power estimation for VLSI circuits,” IEEE Trans. Computer-Aided Design, vol. 18, pp. 697–713, June 1999.

[PED 89] M. Pedram, B. T. Preas. " Interconnection length estimation for optimized standard cell layouts. " In Proceedings of the IEEE International Conference on Computer Aided Design, pages 390-393, November 1989.

[PED 91] M. Pedram and N. Bhat. " Layout driven technology mapping. " In Proceedings of the 28th Design Automation Conference, pages 99-105, June 1991.

[PEE 04] Peeters (A.) et Solutions (H.), “Bringing handshake technology to the open market”, Asynchronous Circuits and Systems, 2004. Proceedings. 10th International Symposium on, 2004.

[PER 98] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In Proceedings of the International Symposium on Low Power electronics and Design, pages 76–81, 1998.

[POW 90] S. R. Powell and P. M. Chau, “Estimating power dissipation of VLSI signal processing chips: The PFA technique,” in Proc. VLSI Signal Processing IV, pp. 250–259, Sept. 1990.

[POW 95] S. R. Powell and P. M. Chau. " A model for estimating power dissipation in a class of DSP VLSI chips. " IEEE Transactions on Circuits and Systems, 36(6):646-650, June 1995.

[QUA 89] T. Quarles. " The SPICE3 Implementation Guide. " Technical Report No. M89-44, Electronics Research Laboratory, University of California, Berkeley, California, April 1989.

[RAB 96] J. Rabaey, M. Pedram, Low Power Design Methodologies, Kluwer AcademicPublishers, 1996

[RAJ 94] S. Rajgopal and G. Mehta. " Experiences with simulation-based schematic level current estimation. " In Proceedings of the 1994 International Workshop on Low Power Design, pages 9Ð14, April 1994.

[RAJ 95] S. Raje and M. Sarrafzadeh. " Variable voltage scheduling. " In Proceedings of the 1995 International Symposium on Low Power Design, pages 9-13, April 1995.

[REI 84] Reinhold P. Weicker, “Dhrystone: a synthetic systems programming benchmark”, Communications of the ACM, v.27 n.10, p.1013-1030, Oct 1984

[REN 98] M. Renaudin, P. Vivet et F. Robin “ASPRO-216: a Standard-Cell Q.D.I. 16-bit RISC Asynchronous Microprocessor”, Proc. ASYNC'98, San Diego, March 29-31, 1998, pp. 22-31.

[RIG 02] J.B. Rigaud, “Spécification de bibliothèques pour la synthèse de circuits asynchrones”, research repport, INP of Grenoble, 2002

[SAP 95] S. S. Sapatnekar and W. Chuang, Power vs. delay in gate sizing: conflicting objectives? Proc. Int. Con on Computer-Azded Design, 13: 463-466, 1995

[SEN 97] O. Sentieys, “Réduction de consommation d’énergie en électronique embarquée”, Journée scientifique électronique embarquée du 24 avril 1997.

[SPE 07] http://www.spec.org/benchmarks.html

[SRI 96] M. Srivastava, A. Chandrakasan and R. Brodersen, “Predictive system shutdown and other architectural techniques for energy efficient programmable computation”, IEEE Transactions on Very Large Scale Integration Systems (1996), Vol. 4, N° 1, pp.42-55.

[SUT 89] Ivan E Sutherland “Micropipelines”, Communication of the ACM Volume 32 N°6 June 1989

[TAK 97] A. Takamura, M. Kuwako, M. Ima, T. Fujii, M. Ozawa, I. Fukasaku, Y. Ueno, and T. Nanya, “TITAC-2: An asynchronous 32-bit microprocessor based on scalable-delayinsensitive model” In Proc. International Conf. Computer Design (ICCD), pages 288–294, October 1997.

[TAJ 07] Tajalli, A. Vittoz, E. Leblebici, Y. Brauer, E.J., “Ultra-low power subthreshold current-mode logic utilising PMOS load device”, Electronics Letters Volume 43, Issue 17 pp. 911–913, August 16 2007.

[TER 95] H. Terada, M. Iwata, S. Komori, and S. Miyata, “Superpipelined dynamic data driven VLSI processors,” in Advanced Topics in Dataflow Computing and Multithreading, G. R. Gao, L. Bic, and J. L. Gaudiot, Eds. New York: IEEE Press, 1995, pp. 75–85.

[TIW 94] V. Tiwari, S. Malik and W. Wolfe. " Power analysis of embedded software: a first step towards software minimization. " IEEE Transactions on VLSI Systems, 2(4):437-445, December 1994.

[TRA 08] Transmeta Crusoe. http://www.transmeta.com

[TIW 95] V. Tiwari, S. Malik and P. Ashar. " Guarded evaluation: Pushing power management to logic synthesis/design. " In Proceedings of the 1995 International Symposium on Low Power Design, pages 221-226, April 1995.

[TUR 95] S. Turgis, N. Azemard and D. Auvergne. " Explicit evaluation of short circuit power dissipation for CMOS logic structures. " In Proceedings of the 1995 International Symposium on Low Power Design, pages 129-134, April 1995.

[TYA 87] A. Tyagi. " Hercules: A power analyzer of MOS VLSI circuits. " In Proceedings of the IEEE International Conference on Computer Aided Design, pages 530Ð533, November 1987.

[UDD 86] Udding (J.), “A formal model for defining and classifying delay-insensitive circuits and systems”, Distributed Computing, vol. 1, no 4, 1986, p. 197-204.

[VAN 93] P. Van Oostende, P. Six and J. Vandewalle and H. De Man. " Estimation of typical power of synchronous {CMOS} circuits using a hierarchy of simulators. " IEEE Journal of Solid State Circuits, 28(1):26-39, January 1993.

[VEE 84] H. J. M. Veendrick. " Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. " IEEE Journal of Solid State Circuits, 19:468Ð473, August 1984.

[WEI 94] M.Weiser, B.Welch, A. J. Demers, and S. Shenker. Scheduling for reduced CPU energy. In Operating Systems Design and Implementation, pages 13–23, 1994.

[WOO 97] V. Woods, P. Day, S. B. Furber, J. D. Garside, N. C. Paver, and S. Temple, “AMULET1: An asynchronous ARM microprocessor,” IEEE Trans. Comput., vol. 46, pp. 385–398, Apr. 1997.

[WUY 94] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergaele and H. De Man. " Global communication and memory optimizing transformations for low power systems. " In Proceedings of the 1994 International Workshop on Low Power Design, pages 203-208, April 1994.

[WYB 97] Edsger Wybe Dijkstra, “A Discipline of Programming”, Prentice Hall PTR, Upper Saddle River, NJ, 1997

[ZAD 65] L.A. Zadeh, “Fuzzy sets”, Information and Control, vol. 8, pp. 338-353, 1965.

[ZHU 03] Y. Zhu and F. Mueller, "Feedback Dynamic Voltage Scaling DVS-EDF Scheduling: Correctness and PID-Feedback", North Carolina State University, 2003.

Annexe A

Code CHP pour réaliser les opérations AND, OR, XOR et NOR

```
component ALU_LOGICAL_SLICE_ELEMENT
   port(
   cmd_alu    : in DI MR[4][1];
  op1_slice   : in DI MR[4][1];
  op2_slice   : in DI MR[4][1];
   res                    : out DI MR[4][1])
begin
process  ALU_LOGICAL_SLICE_ELEMENT
   port(
   cmd_alu    : in DI MR[4][1];
  op1_slice   : in DI MR[4][1];
  op2_slice   : in DI MR[4][1];
   res                    : out DI MR[4][1])
variable cmd_alu_var    : MR[4][1];
variable op1_slice_var: MR[4][1];
variable op2_slice_var: MR[4][1];
begin
[cmd_alu ? cmd_alu_var,
op1_slice ? op1_slice_var,
op2_slice ? op2_slice_var;
   @[op1_slice_var = "0"[4] =>
        @[op2_slice_var = "0"[4] =>
             @[cmd_alu_var = "0"[4] =>                    -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "0"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "0"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "3"[4];
                  break];
        break
        op2_slice_var = "1"[4] =>
             @[
             cmd_alu_var = "0"[4] =>                      -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "1"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "1"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "2"[4];
                  break
             ];
        break
        op2_slice_var = "2"[4] =>
             @[
             cmd_alu_var = "0"[4] =>                      -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "2"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "2"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "1"[4];
                  break
             ];
        break
        op2_slice_var = "3"[4] =>
             @[
             cmd_alu_var = "0"[4] =>                      -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "3"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "3"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "0"[4];
                  break];
        break];
   break
   op1_slice_var = "1"[4] =>
        @[
        op2_slice_var = "0"[4] =>
             @[cmd_alu_var = "0"[4] =>                    -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "1"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "1"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "2"[4];
                  break];
        break
        op2_slice_var = "1"[4] =>
             @[cmd_alu_var = "0"[4] =>                    -- AND
                  res ! "1"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "1"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "0"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "2"[4];
                  break];
        break
        op2_slice_var = "2"[4] =>
             @[cmd_alu_var = "0"[4] =>                    -- AND
                  res ! "0"[4];
                  break
             cmd_alu_var = "1"[4] =>                      -- OR
                  res ! "3"[4];
                  break
             cmd_alu_var = "2"[4] =>                      -- XOR
                  res ! "3"[4];
                  break
             cmd_alu_var = "3"[4] =>                      -- NOR
                  res ! "0"[4];
                  break];
```

```
                break
                op2_slice_var = "3"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "1"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "2"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break];
        break
        op1_slice_var = "2"[4] =>
                @[op2_slice_var = "0"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "0"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "2"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "2"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "1"[4];
                                break];
                break
                op2_slice_var = "1"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "0"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break
                op2_slice_var = "2"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "2"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "2"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "0"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "1"[4];
                                break];
                break
                op2_slice_var = "3"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "2"[4];
                                break

                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "1"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
```

```
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "1"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break];
        break
        op1_slice_var = "3"[4] =>
                @[op2_slice_var = "0"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "0"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break
                op2_slice_var = "1"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "1"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "2"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break
                op2_slice_var = "2"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "2"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "1"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break];
                break
                op2_slice_var = "3"[4] =>
                        @[cmd_alu_var = "0"[4] =>                       -- AND
                                res ! "3"[4];
                                break
                        cmd_alu_var = "1"[4] =>                         -- OR
                                res ! "3"[4];
                                break
                        cmd_alu_var = "2"[4] =>                         -- XOR
                                res ! "0"[4];
                                break
                        cmd_alu_var = "3"[4] =>                         -- NOR
                                res ! "0"[4];
                                break
                        ];
                break
                ];
        break
        ];
```

```
                    break
                cmd_alu_var = "3"[4] =>                    -- NOR
                    res ! "0"[4];
                    break];
            break];
        break];
    loop];
    end; -- PROCESS  ALU_LOGICAL_SLICE_ELEMENT
    end; -- COMPONENT ALU_LOGICAL_SLICE
```

Annexe B

Code VHDL d'un digit pour réaliser les opérations AND, OR, XOR et NOR

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
use STD.textio.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_textio.all ;
use IEEE.numeric_std.all;
use IEEE.std_logic_arith.all;
use ieee.numeric_std.all;
use IEEE.std_logic_arith.ALL;
entity logical_sync_1_digit is
port (
reset: in std_logic;
clk : in std_logic;
D_in_a : in std_logic_vector(1 downto 0);
D_in_b : in std_logic_vector(1 downto 0);
cmd_alu_logical : std_logic_vector(1 downto 0);
D_out : out std_logic_vector(1 downto 0));
end logical_sync_1_digit;
architecture a of logical_sync_1_digit is
begin  -- a
logical_element : process(clk, reset)
begin
   if (reset = '1') then
         D_out <= "00";
   elsif (clk'event and clk = '1') then
         if (cmd_alu_logical = "00") then
               D_out <= D_in_a and D_in_b;
         elsif (cmd_alu_logical = "01") then
               D_out <= D_in_a or D_in_b;
         elsif (cmd_alu_logical = "10") then
               D_out <= D_in_a xor D_in_b;
         elsif (cmd_alu_logical = "11") then
               D_out <= D_in_a nor D_in_b;
         end if;
   end if;
end process;
out_register : process(clk, reset)
begin
   if (reset = '1') then
         D_out <= "00";
   elsif (clk'event and clk = '1') then
         D_out <= D_out_comb;
   end if;
end process;
end a;
```

Annexe C

Chemin critique pour l'unité logique synchrone

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
        -sort_by group
Design : logical_sync
Version: Y-2006.06
Date   : Fri Jun 20 02:06:15 2008
****************************************

Operating Conditions: Worst   Library: CORE9GPLL
Wire Load Model Mode: enclosed

  Startpoint: cmd_alu_logical[1]
              (input port)
  Endpoint: D_out_reg_0
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  logical_sync       area_3Kto4K           CORE9GPLL

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock (input port clock) (rise edge)     0.00       0.00
  input external delay                     0.00       0.00 f
  cmd_alu_logical[1](in)                   0.00       0.00 f
  U251/Z (ND2LL)                           0.45       0.45 r
  U250/Z (BFLLX05)                         0.78       1.24 r
  U453/Z (AO2LL)                           0.42       1.65 f
  U450/Z (MUX21LL)                         0.29       1.94 f
  U449/Z (AO7LL)                           0.10       2.04 r
  D_out_reg_0/D (FD2QLLP)                  0.00       2.04 r
  data arrival time                                   2.04

  clock clk (rise edge)                    3.00       3.00
  clock network delay (ideal)              0.00       3.00
  D_out_reg_0/CP (FD2QLLP)                 0.00       3.00 r
  library setup time-                      0.34       2.66
  data required time                                  2.66
  -----------------------------------------------------------
  data required time                                  2.66
  data arrival time-                                  2.04
  -----------------------------------------------------------
  slack (MET)                                          0.62
```

Chemin critique pour l'unité de décalage synchrone

```
****************************************
Report : timing
       -path full
       -delay max
       -max_paths 1
       -sort_by group
Design : alu_shift_sync_5_stages
Version: Y-2006.06
Date   : Tue Jun  3 21:08:06 2008
****************************************
Operating Conditions: Worst   Library: CORE9GPLL
Wire Load Model Mode: enclosed

  Startpoint: op1[0] (input port)
  Endpoint: D_out_reg[24]
            (rising edge-triggered flip-flop clocked by CLK_0)
  Path Group: CLK_0
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  alu_shift_sync_5_stages
                     area_3Kto4K           CORE9GPLL

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock (input port clock) (rise edge)     0.00       0.00
  input external delay                     0.00       0.00 f
  op1[0] (in)                              0.00       0.00 f
  U244/Z (IVLLX05)                         0.54       0.54 r
  U238/Z (NR2LL)                           0.94       1.47 f
  U237/Z (BFLLX05)                         0.66       2.13 f
  U452/Z (AO11NLL)                         0.58       2.71 f
  U445/Z (AO9NLL)                          0.31       3.02 f
  U444/Z (AO10LL)                          0.36       3.37 r
  U410/Z (AO35LL)                          0.47       3.84 f
  U404/Z (AO2ALL)                          0.24       4.08 r
  U403/Z (MUX21LL)                         0.24       4.33 r
  U402/Z (NR3LL)                           0.34       4.66 f
  U267/Z (IVLLX05)                         0.53       5.19 r
  U265/Z (AO52LL)                          0.29       5.49 f
  D_out_reg[24]/D (FD2QLL)                 0.00       5.49 f
  data arrival time                                   5.49

  clock CLK_0 (rise edge)          6.00    6.00
  clock network delay (ideal)              0.00       6.00
  D_out_reg[24]/CP (FD2QLL)                0.00       6.00 r
  library setup time                       -0.36      5.64
  data required time                                  5.64
  -----------------------------------------------------------
  data required time                                  5.64
  data arrival time                                   -5.49
  -----------------------------------------------------------
  slack (MET)                                         0.16
```

Systemes a microprocesseurs asynchrones basse consommation.

Cette thèse présente une contribution à la conception de circuits asynchrones Quasi Insensibles aux Délais (QDI) faible consommation. Une brève étude des méthodes d'estimation de l'énergie dans les circuits CMOS est présentée. Dans le deuxième chapitre, la méthodologie proposée sera présentée. Cette méthodologie utilise trois outils qui permettent la synthèse, l'optimisation et l'estimation d'énergie des circuits asynchrones QDI. La conception de ces circuits se fait à partir d'un langage de haut niveau (CHP). Le troisième chapitre expose une étude sur les choix d'architectures lors de la conception des circuits asynchrones QDI en utilisant la méthodologie proposée. Une comparaison avec les équivalents synchrones des architectures étudiées sera aussi montrée. Finalement, le quatrième chapitre présente une technique pour réduire la consommation d'un circuit en régulant la tension d'alimentation avec un asservissement à boucle fermée pour contrôler la tension d'alimentation.

Low power asynchronous microprocessor systems

This Work presents a contribution to the design of asynchronous QDI (Quasi Delay insensitive) circuits for low power consumption. A quick study of the power estimation techniques will be shown. The methodology proposed will be presented in the chapter 2. This methodology uses 3 tools that perform the synthesis, optimization and the estimation of the asynchronous QDI circuits. The design of those circuits is done with a high level language for asynchronous circuits (CHP). The third chapter shows a study of different architectures to select the best one in terms of power consumption, speed and size. That chapter also shows a comparison between the equivalent synchronous circuits. In the final chapter, a technique for the reduction of the power consumption is presented. This technique changes the voltage of the circuit with a feedback control.

ISBN: 978-2-84813-122-1