ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE ÉLECTRIQUE
Ph. D.

PAR
René GAGNÉ

JADIS SYNCHRONES, DÉSORMAIS GALS, LES ARCHITECTURES DE FPGA

MONTRÉAL, LE 14 DÉCEMBRE 2009

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE

M. Jean P. Belzile, directeur de thèse
Département de génie électrique à l'École de technologie supérieure

M. Claude Thibeault, codirecteur de thèse
Département de génie électrique à l'École de technologie supérieure

M. Stéphane Coulombe, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Ghyslain Gagnon, membre du jury
Département de génie électrique à l'École de technologie supérieure

M. Pierre Langlois, examinateur externe
Département de génie informatique et génie logiciel à l'École Polytechnique de Montréal

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

8 DÉCEMBRE 2009

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

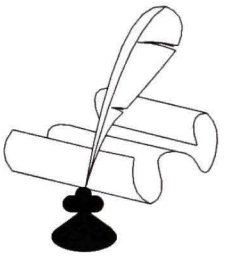
REMERCIEMENTS

Avant de commencer, je tiens à remercier mon directeur de recherche, Dr Jean P. Belzile, pour son audace à aborder le thème de l'asynchronisme, ses judicieux conseils et sa facilité à transmettre son savoir. C'est sa passion, transmise au fil des années, qui m'a amené à présenter mes résultats de recherche lors de différents événements internationaux. Des activités, pour lesquelles, j'ai également pu bénéficier des talents de rédaction et du support de mon codirecteur de recherche, Dr Claude Thibeault.

Un travail d'une telle ampleur n'aurait jamais vu le jour sans un support financier et matériel adéquat. À cet effet, je tiens à exprimer ma gratitude, en ordre d'importance de contribution, au Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), au Fonds québécois de la recherche sur la nature et les technologies (FQNRT), à l'École de technologie supérieure (ETS) et à Bell Canada pour leurs contributions financières. Du point de vue matériel, toute ma reconnaissance va à la Société Canadienne de Microsystèmes (CMC Microsystems) pour les différents outils de simulation et de conception mis à ma disposition et au LACIME pour m'avoir fourni un environnement de travail.

Malgré la présence du mentorat et des ressources, ce travail n'aurait jamais été effectué sans le support et l'amour de ma famille ainsi qu'à l'appui indéfectible, aux perpétuels encouragements, à la patience et à la confiance, de ma muse, Eugénie chérie, tout au long de cette aventure, pour le moins, tortueuse.

Merci!



JADIS SYNCHRONES, DÉSORMAIS GALS, LES ARCHITECTURES DE FPGA

René GAGNÉ

RÉSUMÉ

Il est de plus en plus difficile de répondre à la demande conflictuelle de circuits plus grands et plus rapides par les avancées seules des technologies des semi-conducteurs. À un certain point, on s'attend à ce que les concepteurs et les fabricants doivent abandonner la méthodologie de conception synchrone traditionnelle pour une méthodologie localement synchrone globalement asynchrone (GALS). De tels changements engendrent plus de contraintes de synchronisation, mais également plus de flexibilité.

En conséquence, une méthodologie pour l'implémentation de composants GALS sur FPGA synchrones traditionnels est d'abord présentée. Les objectifs sont de définir un ensemble minimal de composants asynchrones de base, de permettre leur implémentation et d'établir les contraintes et les limitations de tels circuits. Les résultats de simulation confirment que des conceptions GALS implémentées à l'aide de ressources du FPGA (tableau de correspondance et bascules) et des outils courants de placement et routage permettent l'implémentation de composants asynchrones tels que la ligne à retard, l'élément C de Muller et l'arbitre. Ces composants peuvent être implémentés dans des FPGA synchrones traditionnels tant que ces conceptions sont soumises à des contraintes appropriées et qu'elles sont utilisées en fonction des limitations du circuit.

Pour atteindre de meilleures performances, une nouvelle architecture de FPGA compatible avec les dispositifs synchrones existants et qui soutient intrinsèquement les conceptions GALS est présentée. L'objectif principal est simple : l'architecture proposée doit apparaître inchangée pour les conceptions synchrones, mais doit inclure un ensemble minimal de composants de base pour empêcher la métastabilité lors de communications asynchrones. Les résultats de simulation, d'un générateur d'horloge qui peut être arrêté, sont présentés. Tous ces résultats démontrent qu'avec très peu de circuits adaptés, une cellule standard de FPGA peut devenir appropriée pour les méthodologies GALS.

Un circuit de masquage des aléas temporels est finalement présenté pour masquer la métastabilité et les problèmes de synchronisation. Le but est de définir un circuit capable de mettre, physiquement, en application les contraintes qui masquent les sources de métastabilité de façon à ce que la synchronisation paraisse transparente. Les résultats de simulation confirment qu'un tel circuit peut masquer totalement toutes les sources de métastabilité sans dégradation des performances, mais avec une latence apparentée au temps nécessaire à la stabilisation d'une bascule de mémoire.

Mots clés : FPGA GALS, composants, architecture, masquage d'aléas temporels.

FROM SYNCHRONOUS TO GALS, THE ARCHITECTURE OF FPGA

René GAGNÉ

ABSTRACT

The conflictual demand of faster and larger designs is increasingly difficult to answer by the advances of solid state technology alone. At some point, it is expected that designers and manufacturers will have to give up the traditional synchronous design methodology for a Globally Asynchronous Locally Synchronous (GALS) one. Such changes imply more synchronization constraints, but also more flexibility.

Consequently, a methodology for implementing GALS design in conventional FPGAs using existing tools is first presented. The goals are to define the minimal set of basic asynchronous components, to permit the methodology of their implementation and to establish the design constraints and limitations of such circuits. Simulation results confirm that GALS designs implemented using the Look-Up Table or the Flip-Flop with Place & Route tools and asynchronous components such as the delay element, the Muller-C element or the arbiter are supported by conventional synchronous FPGAs as long as these designs are implemented within suitable constraints and operated within well defined circuit limitations.

To achieve better performances, a novel FPGA architecture that is compatible with existing devices and that can support GALS designs natively is presented. The main objective is simple: the proposed architecture must appear unchanged for synchronous design, but also includes a minimal amount of basic components to prevent metastability for efficient asynchronous communications. A pausible clock generator application and simulation results for the proposed architecture is presented. All results demonstrate that with a few additional customized circuits, a standard FPGA cell can become appropriate for GALS methodologies.

A glitch masking circuitry is finally presented to mask completely metastability and avoid synchronisation problems. The aim is to define a circuit able to implement physically the constraints that mask metastability sources and that appear transparent during synchronization. Simulation results confirm that such a circuit can totally mask metastability sources with no performances degradations, introducing only a latency equivalent to the setup time of a typical flip flop.

Keywords: GALS FPGA, components, architecture, glitch masking.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ASIC	Application Specific Integrated Circuits
CAD	Computer Aided Design
CASC	Coalition for Academic Scientific Computing
CLB	Configurable Logic Bloc
CMOS	Complementary Metal-Oxide Semi-conductor
CPU	Central Processing Unit
DI	Delay Insensitive
DSP	Digital Signal Processing
ECM	Élément C de Muller
EDA	Electronic Design Automation
EME	Élément Mutuellement Exclusif
EMI	ElectroMagnetic Interference
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
HDL	Hardware Description Language
I	Interconnection
IC	Integrated Circuit
IP	Intellectual Property
I/O	Input/Output
ITRS	International Technology Roadmap for Semiconductors
LB	Logic Bloc
LUT	Look-Up Table

MUX	MUltipleXer
MUTEX	MUTual Exclusion
NCL	Null Convention Logic
PCC	Pausing Clock Circuit
PLL	Phase-Locked Loop
P&R	Place & Route
PP&RS	Post Place & Route Simulation
QDI	Quasi-Delay Insensitive
RAM	Random Access Memory
SDF	Standard Delay Format
SI	Speed Independent
SoC	System on Chip
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale of Integration
VPCS	Variable Period Clock Synthesis

LISTE DES SYMBOLES ET UNITÉS DE MESURE

UNITÉS DE BASE

m	mètre (unité de longueur)
s	seconde (unité de temps)

UNITÉS DE TEMPS

s	seconde
ms	milliseconde
µs	microseconde
ns	nanoseconde
ps	picoseconde

Fréquence

GHz	gigahertz
MHz	mégahertz
kHz	kilohertz
Hz	hertz

UNITÉS GÉOMÉTRIQUES

Longueur

m	mètre
mm	millimètre
µm	micromètre
nm	nanomètre

Différence de potentiel (ou tension)
Force électromotrice

V	volt
mV	millivolt

Résistance et conductance électriques

Ω	ohm
mΩ	milliohm
µΩ	microhm

Capacité électrique

F	farad
µF	microfarad
nF	nanofarad
pF	picofarad
fF	femtofarad

INTRODUCTION

Contexte de réalisation

Entre 1971 et 2001, la densité des circuits intégrés a doublé tous les deux ans. Cette croissance, mue par la miniaturisation des circuits intégrés, a permis une intégration massive favorisant la production de circuits plus rapides à moindres coûts. Une évolution qui explique aussi l'escalade des performances des circuits intégrés.

Malgré les efforts concertés des concepteurs de circuits intégrés pour perpétuer la tendance, nous assistons, depuis quelques années, à un ralentissement. Bien qu'involontaire, ce dernier n'a pas pour objectif d'augmenter l'espérance de vie des circuits intégrés, mais plutôt de régler des difficultés, comme celui de la synchronisation liée à la miniaturisation. Pour comprendre le problème et construire les outils pour le contourner, une incursion dans l'univers des circuits intégrés s'impose.

Une horloge pour tous

Durant les belles années de l'intégration, la synchronisation globale régissait essentiellement la conception de circuits électroniques. Un principe selon lequel un signal périodique, communément appelé « horloge », est responsable d'indiquer le début et la fin de toutes les opérations. À la manière du maître de bord qui ordonne les mouvements de rames dans une embarcation, chaque opération se voit octroyer une quantité de temps pour son exécution. Dans la mesure où chacune d'elle est effectuée dans le temps prévu, le système fonctionne correctement. Il est donc aisé de comprendre la popularité de la synchronisation globale.

Cette simplicité apparente cache néanmoins des limites. À titre d'exemple, notons la fréquence d'opération limitée par la tâche requérant le plus grand temps de traitement dans le système. À l'instar d'une chaîne dont la force est limitée par son maillon le plus faible, la vitesse maximale d'un tel système dit synchrone est limitée à son élément le plus lent. De surcroit, l'intégration complexifie la synchronisation globale. En regroupant plusieurs sous-

circuits qui ne partagent pas forcément la même horloge, les circuits ainsi créés n'ont plus leur référence unique et les principes de synchronisation deviennent difficiles, voire impossibles à appliquer.

Pendant des années, les méthodes de conceptions synchrones offraient rapidité et simplicité. Cependant, l'ampleur des inconvénients actuels liés à cette méthodologie de conception suggère qu'il serait plus efficace d'avoir une alternative à la synchronisation globale.

La ponctualité malgré l'absence d'horloge

Les systèmes sans horloge, dits asynchrones, présentent nombre d'avantages. L'élimination complète du réseau de distribution du signal d'horloge permet de supprimer les problèmes de distribution du signal d'horloge et de récupérer les 30 à 50 % de la puissance nécessaire à son fonctionnement. Ces systèmes asynchrones compensent l'absence d'horloge par l'utilisation de protocoles de communications. Parmi les différents protocoles, certains maximisent la vitesse d'opération, d'autres minimisent les ressources. Dans tous les cas, ils améliorent les performances des systèmes asynchrones par rapport aux systèmes synchrones. Chaque module travaille à capacité maximale sans être limité par l'élément le plus lent du système. En conséquence, les communications sont plus rapides tout en réduisant le taux d'émission d'ondes électromagnétiques, EMI, (**E**lectro**M**agnetic **I**nterference). Une telle réduction est imputable à la nature aléatoire des communications qui ne sont pas dictées par le rythme régulier d'une horloge.

Malheureusement, l'absence d'horloge de synchronisation n'a pas que des avantages. Le manque de connaissances, d'outils et de techniques est le principal problème à l'émergence d'une méthodologie mature de conception asynchrone. Ces manques s'expliquent, en partie, par les besoins de l'industrie qui stimulent à leur tour la formation offerte. Les universités se concentrent sur les conceptions synchrones à cause de leur simplicité et de leur dominance. De fait, les concepteurs sont des experts de la conception synchrone et la disponibilité des outils reflète cette réalité. Dans ces circonstances, peu de concepteurs s'aventurent sur la voie de l'asynchronisme.

Malgré tout, quelques groupes ont travaillé sur la logique asynchrone au cours des dernières décennies. Ces travaux ont permis de mettre au point quelques outils relativement primitifs pour les conceptions asynchrones. Cependant, beaucoup de travail devra encore être fait pour créer des outils répondant aux besoins actuels en matière de conception numérique. Bien que plusieurs réalisations prouvent la viabilité de la logique asynchrone, elle demeure marginale.

Une solution à ce problème passe par une approche hybride. Une méthodologie exploitant les avantages des conceptions synchrones et asynchrones en minimisant leurs inconvénients.

Plusieurs horloges locales sans synchronisation globale

La logique Globalement Asynchrone, mais Localement Synchrone, GALS, (**G**lobally **A**synchronous **L**ocally **S**ynchronous) est spécialisée pour les circuits contenant différents domaines d'horloge. Proposée il y a plus de 20 ans par D. M. Chapiro, cette méthode refait surface. Ce retour est motivé par l'évolution des techniques asynchrones et l'exacerbation des problèmes associés à la synchronisation globale. Une forme populaire de réalisation GALS consiste à créer des circuits synchrones de taille restreinte et de les relier entre eux par des protocoles asynchrones. Dans cette optique, il est possible de construire des circuits de très grande taille sans avoir les difficultés de la synchronisation globale.

Alors, pourquoi est-ce que tous les systèmes synchrones n'ont pas encore été remplacés? En fait, il ne s'agit pas d'éliminer la synchronisation globale. Certains circuits se réalisent et continueront de se réaliser très bien de façon synchrone. Pour ces derniers, un changement de méthodologie exigerait des efforts et des coûts supplémentaires inutiles.

Seuls les circuits très rapides, de tailles démesurées ou qui ont naturellement plusieurs domaines d'horloge posent problème. Pour ces circuits, une alternative à la synchronisation globale est nécessaire. En ce sens, la logique GALS représente, actuellement, une alternative prometteuse. Malheureusement, cette migration vers des méthodes de conception GALS est ralentie par le nombre restreint d'outils soutenant l'asynchronisme dans les circuits intégrés. Bref, c'est encore une fois le problème de l'œuf et de la poule. La rareté d'outils freine les

concepteurs et le nombre restreint de concepteurs gêne le développement d'outils performants. Cependant, pour ces circuits, les jours de la synchronisation globale semblent bel et bien comptés.

Support de réalisation

Le support de réalisation choisi pour l'implémentation de systèmes GALS est un composant de logique programmable, FPGA, (**F**ield **P**rogrammable **G**ate **A**rray). Ce support est privilégié pour plusieurs raisons telles que les coûts et la disponibilité. Cependant, il présente également l'avantage intrinsèque d'être programmable et donc de servir d'outil de vérification pour des réalisations à base de semi-conducteurs, ASIC, (**A**pplication **S**pecific **I**ntegrated **C**ircuits). Étant donné que les coûts liés aux ASIC sont très élevés, une telle plateforme de vérification permet de diminuer les risques avant la fabrication, mais également de développer, et ce, rapidement, des preuves de concept.

Objectifs

Cette thèse regroupe trois objectifs complémentaires portant sur un même thème : les FPGA GALS. Pris séparément, chaque objectif pourrait déboucher sur une thèse complète. Cependant, une approche exploratoire a été privilégiée : aborder plusieurs thèmes connexes de façon à contribuer à l'avancement des connaissances, sur un large spectre, dans le domaine de la logique GALS. Malgré cette approche exploratoire, la recherche exhaustive sur ces trois objectifs nous a permis de cibler plusieurs aspects fondamentaux.

Le premier objectif est une méthodologie d'implémentation de composants GALS dans un FPGA synchrone. Il s'agit d'une méthode permettant d'utiliser les ressources disponibles des FPGA synchrones pour implémenter des composants asynchrones. Les composants ciblés sont ceux qui permettent d'établir des communications efficaces entre deux domaines d'horloge (de fréquence différente et de phase différente). Pour des systèmes de très grandes tailles ou de fréquences très élevées, les communications synchrones deviennent l'élément le plus contraignant pour la fréquence maximale du système. Dans ces circonstances, des

communications asynchrones, fonctionnant au maximum de performances des modules impliquées dans la communication, offrent un potentiel de performances supérieures à celles de la logique synchrone où le système est construit en fonction du pire cas.

De plus, cette méthodologie fournit un support viable de vérification pour les circuits asynchrones. Étant donné que la méthodologie proposée est basée sur un circuit programmable de type FPGA, il est possible d'exploiter un support connu peu coûteux et offert en masse pour faire des preuves de concept de circuits asynchrones avant de les réaliser sur des ASIC, beaucoup plus onéreux.

Malgré le potentiel de la méthodologie d'implémentation, le support synchrone limite les performances et l'étendue des applications réalisables. Ainsi, le second objectif consiste à proposer une nouvelle architecture GALS de FPGA. Cette architecture doit paraître fondamentalement synchrone tout en permettant les transferts asynchrones. Loin d'une architecture complètement asynchrone, cette architecture GALS cible les conceptions synchrones de très grandes tailles ou de fréquences très élevées pour lesquelles le remplacement des longs liens de communication synchrones par des liens asynchrones présente des avantages sur le plan des performances générales du système. Cet objectif n'invalide pas le précédent, en fait, il le complète en abordant une catégorie étendue de système GALS. Par contre, cet objectif ne couvre pas en détail la méthode utilisée pour déterminer la nature de ces liens et les remplacer au besoin. Cependant, certains éléments seront présentés, à cet égard, sous forme de travaux futurs.

Forts d'une méthodologie d'implémentation de composants GALS et d'une nouvelle architecture de FPGA GALS, nous constatons qu'un frein important à l'émergence des systèmes GALS et des systèmes asynchrones est la possibilité d'états métastables. C'est pourquoi le troisième objectif est de présenter un circuit qui masque complètement la métastabilité. L'approche exploitée ici est un circuit à une entrée et une sortie pour lequel seuls les signaux ne pouvant pas générer d'états métastables en sortie peuvent être portés de l'entrée à la sortie. L'aspect intéressant de ce circuit est l'absence d'hypothèse ou de restriction sur le signal d'entrée, le circuit se charge d'analyser la forme d'onde et de

masquer les aspects problématiques d'un signal. Le circuit produit un signal de sortie qui ne créera pas de métastabilité. Le problème est donc réglé avant qu'il ne se produise.

Malgré ce dernier objectif très ambitieux, le circuit doit être, en plus, suffisamment général pour être utilisable dans une pléiade d'applications. La plus importante est certainement celle de la synchronisation transparente de signaux haute fréquence et haut débit à travers différents domaines d'horloge.

Contributions

Cette thèse présente des contributions à différents niveaux :

- méthodologie de conception,
- architecture de circuit programmable,
- différents éléments de circuits.

Malgré son apparence simpliste, la contribution en méthodologie est très importante, puisqu'il n'est pas commun de synthétiser des composants asynchrones sur un FPGA synchrone à l'aide d'outils synchrones. Étant donné que la principale faiblesse des circuits asynchrones est le manque d'outils, une telle méthode s'avère un avantage de taille dans le but de réaliser un prototype programmable à faible coût avant de le réaliser sous forme d'ASIC. Dans ces circonstances, la contribution est un guide de synthèse pour implémenter les structures asynchrones souhaitées sans craindre les éventuelles optimisations synchrones de l'outil. De plus, la méthode permet d'identifier les éléments de synchronisation critiques étant donné l'utilisation d'outils synchrones. Ces résultats furent publiés à l'occasion de FETCH-2009 (Gagné, Belzile et Thibeault, 2009d), de NEWCAS-TAISA-2009 (Gagné, Belzile et Thibeault, 2009b) et du 77^{e} congrès de l'ACFAS (Gagné, Belzile et Thibeault, 2009c).

La contribution, en architecture, s'articule autour de la réutilisation de l'architecture des FPGA de Xilinx et de l'ajout de la quincaillerie nécessaire pour effectuer les mêmes

opérations que celles présentées pour la contribution méthodologique. Cependant, le tout est fait nativement afin d'augmenter les performances des systèmes implémentés. De plus, dans le but de rendre cet ajout transparent aux concepteurs (pour les conceptions synchrones) et de modifier au minimum l'architecture de référence, cette contribution a donné lieu à une plus petite contribution en circuit : un circuit d'arbitrage constitué à l'aide de composants asynchrones. Le circuit proposé exhibe une très faible probabilité de métastabilité par le masquage, a priori, mais également a posteriori, de sources de métastabilité. Ces résultats furent publiés à l'occasion de NEWCAS-2007 (Gagné, Belzile et Thibeault, 2007), de FETCH-2008 où j'ai reçu ***le prix du meilleur doctorant*** (Gagné, Belzile et Thibeault, 2008) et du 77^{e} congrès de l'ACFAS (Gagné, Belzile et Thibeault, 2009a). L'article de NEWCAS-2007 a, par la suite, été présélectionné pour être bonifié et publié dans la revue scientifique « microelectronics journal » (Gagné, Belzile et Thibeault, 2009e). Cet article est actuellement sous presse.

C'est ce circuit d'arbitrage qui est à la base du circuit de masquage complet des sources de métastabilité, la contribution majeure de cette thèse. Le circuit proposé implémente physiquement les contraintes de masquage de la métastabilité de façon à régler les problèmes de synchronisation en aval, et ce, de façon transparente pour le concepteur. Ce dernier n'a qu'à ajouter ce circuit sur les signaux de synchronisation et à oublier le concept même de la métastabilité. Le circuit masque les sources de métastabilité sans dégradation des performances, mais avec une latence de l'ordre du temps de stabilisation des éléments subséquents (peu importe leur nature). Ces résultats font l'objet d'une demande de brevet (numéro de dossier Valéo : VAL-077). Deux articles de revue sont en cours de rédaction.

Les retombées de cette thèse en matière de publications sont illustrées à la figure 1. Cette figure illustre, également, la progression de la thèse en fonction de chaque objectif. Les éléments du haut, allant jusqu'au remplacement des liens de communication, représentent les objectifs tels que vus initialement. Pendant ce temps, d'autres objectifs, plus intéressants, ont captivé notre attention. Il est toutefois à noter que l'outil de segmentation sera présenté, en détail, dans la section des futurs travaux de la conclusion de cette thèse. Naturellement, étant donné l'ampleur des travaux effectués, cette section sera tout de même bien étoffée.

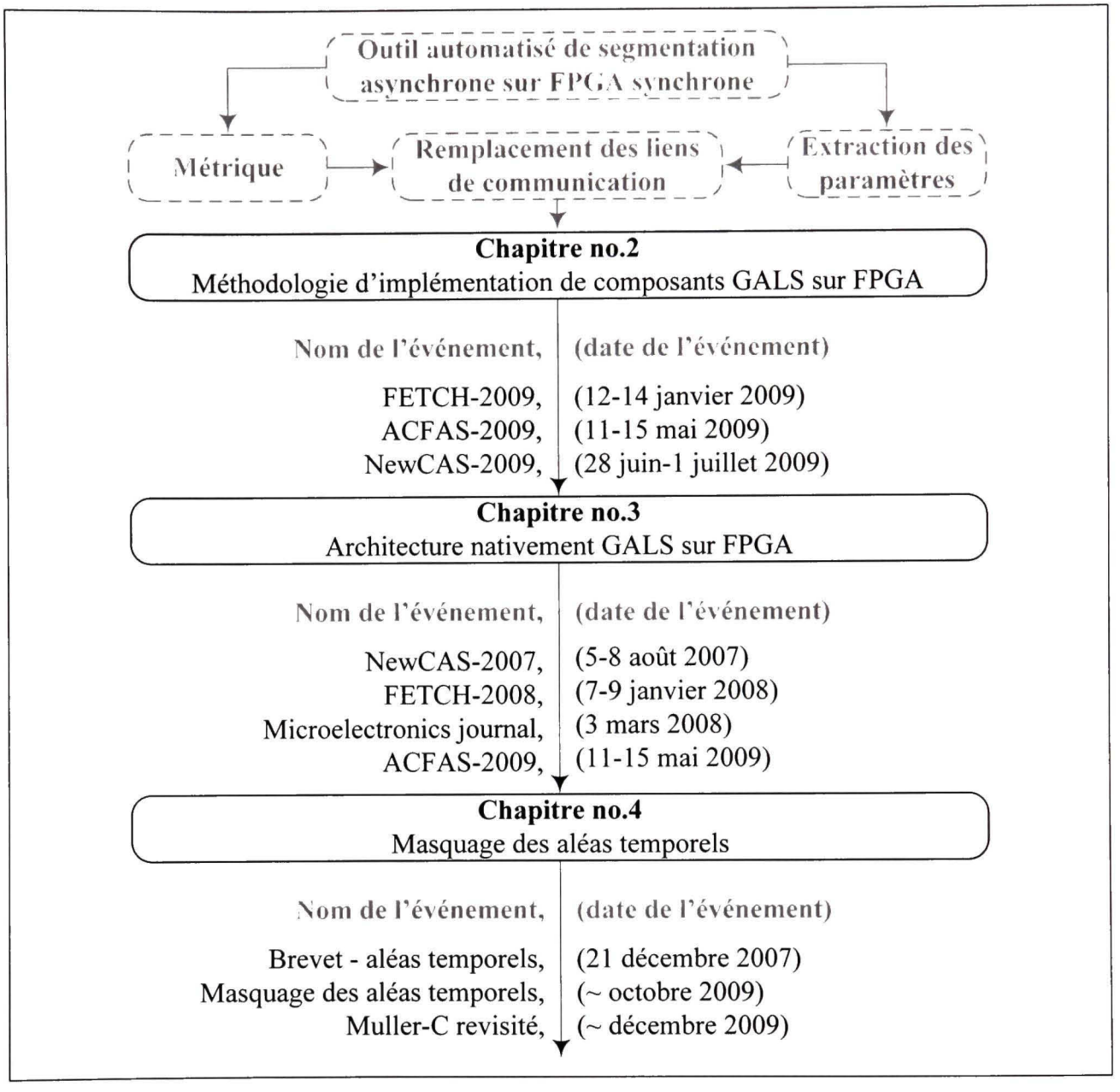


Figure 1 Vue d'ensemble des contributions.

Organisation

Cette thèse est divisée en quatre chapitres. Un premier chapitre de revue de la littérature chapeaute trois autres chapitres, à raison d'un chapitre par objectif.

Le chapitre 1 présente la revue de la littérature générale sur la logique synchrone et la logique asynchrone. Ce survol permet d'identifier les forces et faiblesses de façon à bien doser la

répartition synchrone/asynchrone pour les conceptions GALS. Il est à noter que chaque chapitre succédant possède également une brève revue de la littérature propre à chaque objectif de façon à cibler les aspects particuliers de chaque objectif.

Le chapitre 2 présente l'architecture générale des FPGA synchrones, la liste de contraintes pertinentes supportées par l'outil de synthèse pour FPGA ainsi que la méthode de conception proposée qui permet d'implémenter des circuits GALS sur un FPGA synchrone. De plus, les circuits implémentés sont comparés à d'autres implémentations équivalentes et sont caractérisés en fonction de leurs performances en thèmes de ressources utilisées.

Le chapitre 3 fait un bref retour sur les architectures de FPGA synchrones de façon à cibler l'endroit où implanter les ressources asynchrones et ainsi créer une architecture nativement GALS. Les éléments proposés sont présentés, analysés et simulés avant d'être incorporés à un exemple complexe de communication asynchrone.

Le chapitre 4 présente l'innovation majeure de cette thèse. Étant donné sa nature innovatrice, ce chapitre est construit de façon itérative, à la manière par laquelle le circuit a été conçu. Chaque étape de conception est basée sur une revue de la littérature exhaustive et les résultats de simulation aiguillent chaque prise de décision. Le chapitre converge rapidement sur la solution tout en restant suffisamment pédagogique pour être compris par un novice dans le domaine.

CHAPITRE 1

Revue de la littérature

Le chapitre de la revue de la littérature a pour objectif de couvrir de façon générale les éléments clefs permettant un passage de la logique synchrone à la logique asynchrone. Chacun des chapitres succédant présentera une brève revue de la littérature pour cibler les intérêts de recherche propres à chacun d'eux.

1.1 Logique synchrone

La logique synchrone est, depuis les années 1960, la méthodologie de conception la plus utilisée pour les circuits intégrés. Pourtant, il ne s'agit pas d'une approche naturelle dans la mesure où toutes les interactions physiques se font de façon asynchrone. Un bon exemple est celui d'une centrale téléphonique. Tous les appels entrent à des instants indépendants. Malgré tout, ces appels sont numérisés, synchronisés et traités de façon synchrone. La synchronisation, inutile à première vue, sert paradoxalement à simplifier le traitement.

1.1.1 Synchronisation

La synchronisation globale, faut-il le rappeler, exploite le principe selon lequel un signal périodique, communément appelé « horloge », est responsable d'indiquer le début des opérations. Cette synchronisation simplifie la vérification des circuits pour autant que le traitement soit terminé avant le début de la prochaine opération. On parle alors d'une synchronisation implicite. Cette contrainte, relative à la période de l'horloge utilisée pour cadencer les événements, force l'utilisation d'une horloge de période assez longue accommodant l'élément le plus lent du circuit considérant les pires conditions d'utilisation (valeur des entrées, température, tension d'alimentation, etc.). La méthode, couramment adoptée pour exploiter la synchronisation globale en réduisant les pertes de performances qu'elle peut engendrer, consiste à fractionner la logique combinatoire et à intercaler des

éléments de mémoire entre chaque sous-module créé. La figure 1.1 illustre une telle architecture communément désignée sous le nom « pipeline synchrone ».

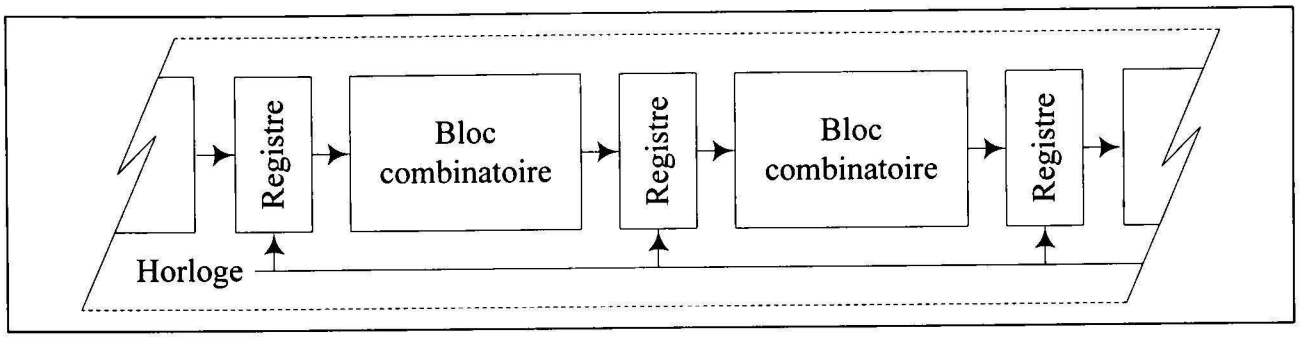


Figure 1.1 Pipeline synchrone.

Cette mémoire supplémentaire sert à synchroniser les données entre les éléments tout en minimisant la période d'horloge, ceci aux dépens de la latence et de la quantité de ressources qu'une telle organisation ajoute. Ces ajouts constituent des inconvénients mineurs étant donné que la mémoire sert également à masquer les aléas temporels (statique ou dynamique) engendrés par la logique combinatoire. La figure 1.2 illustre le masquage des aléas temporels pour la logique synchrone.

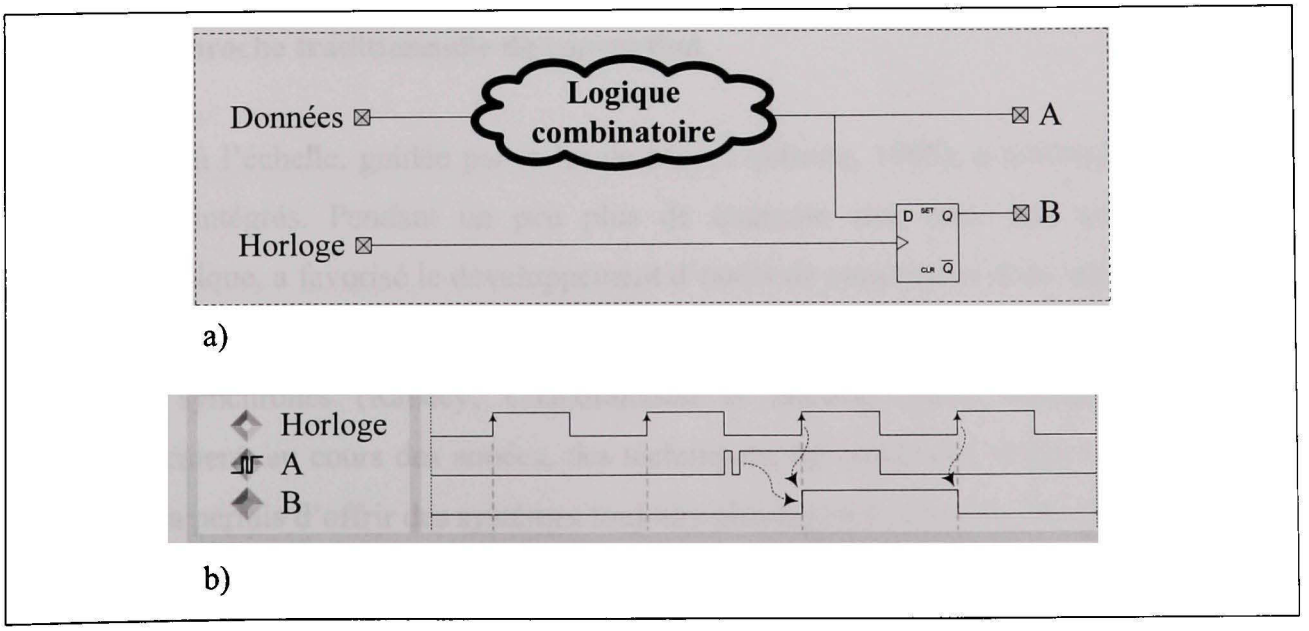


Figure 1.2 Circuit combinatoire.
a) Avec registre. b) Chronogramme associé.

Le circuit de la figure précédente reçoit une série particulière de données et produit un signal (signal A, figure 1.2) en fonction de l'agencement de la logique combinatoire. Cette sortie peut produire un aléa temporel qui se traduit par une donnée erronée. Un travail d'optimisation au niveau de l'équation booléenne de la logique combinatoire permet de prévenir la formation de cet aléa. Cependant, si on ajoute une bascule de synchronisation, ce travail supplémentaire s'avère inutile (signal B, figure 1.2). Cette bascule filtre les transitions indésirables lorsque les paramètres de la bascule sont respectés. Les spécificités de ces paramètres seront abordées dans les prochains chapitres. Malgré tout, il convient de noter qu'il ressort de ceci un désavantage clair pour la logique combinatoire, ou de façon plus générale, la logique asynchrone à générer des signaux sans aléas temporels (Cortadella, 2002). De plus, l'ajout de termes à l'équation booléenne crée des redondances qui rendent le test plus complexe et le circuit plus difficile à observer et à commander (Hazewindus, 1992).

Les éléments simplificateurs de la logique synchrone, dont le masquage des aléas temporels fait partie, font en sorte qu'elle est plus simple et plus facile à utiliser que la logique asynchrone. Ce sont ces éléments simplificateurs qui ont favorisé l'expansion et la dominance des méthodologies de conception synchrone.

1.1.2 Approche traditionnelle de conception

La réduction à l'échelle, guidée par la loi de Moore (Moore, 1965), a favorisé l'intégration des circuits intégrés. Pendant un peu plus de quarante ans, cette loi, au cœur de la microélectronique, a favorisé le développement d'outils de conception et de vérification pour les circuits à base de semi-conducteurs, CMOS, (**C**omplementary **M**etal-**O**xide **S**emi-conductor) synchrones (Rabaey, Chandrakasan et Nikolic, 2003; Savaria, 1988). Le perfectionnement, au cours des années, des techniques, des méthodes et des outils relatifs à ces circuits a permis d'offrir des systèmes toujours plus rapides, avec plus de fonctionnalités.

Cette évolution a cependant mis en évidence certaines limitations qui, selon l'ITRS, ITRS, (**I**nternational **T**echnology **R**oadmap for **S**emiconductors) ont tôt fait d'assombrir les perspectives de la logique synchrone (ITRS, 2007). Parmi les enjeux identifiés par l'ITRS,

notons la fréquence d'opération, la surface des circuits et la puissance consommée, des enjeux qui atteignent des envergures problématiques. Ainsi, malgré une évolution relativement constante des technologies CMOS, la disparité des différentes parties des systèmes exacerbe l'impact de ces enjeux sur les performances générales (Stallings, 1999). En ce sens, la vitesse d'accès mémoire, la capacité des mémoires, la vitesse des éléments de calcul et le taux de transfert entre ces éléments évoluent de façon disparate. À titre d'exemple, la capacité des éléments de calcul suit la loi de Moore et double tous les 18 mois alors que les performances des communications réseau doublent tous les 48 mois (Educause et CASC, 2009). Cette hétérogénéité complique considérablement la conception de systèmes de très grande taille et ceux de très grand débit (Johnson et Graham, 1993). Elle nuit également aux systèmes dits à contraintes temporelles dures comme les systèmes en temps réel (Laplante, 1997).

Traditionnellement, la logique synchrone bénéficiait d'un avantage de taille. La vitesse des portes logiques relativement lente par rapport à la vitesse de propagation des signaux dans les fils simplifiait l'analyse temporelle. Elle permettait même de la négliger dans certains cas. Il va sans dire que cet avantage a favorisé, par sa simplicité, l'utilisation des méthodologies de conception synchrones. Les percées technologiques des dernières années rendent cette réalité de moins en moins vraie. L'inertie favorise toujours les conceptions synchrones, mais il est de plus en plus évident qu'une place existe pour les logiques GALS et asynchrone dans le coffre à outils des concepteurs en microélectronique.

1.1.3 Approche moderne de conception

Malgré toutes les simplifications qu'apportent les méthodologies de conception synchrone lors de la conception de circuits intégrés, ces méthodologies s'avèrent de plus en plus difficiles à appliquer aux systèmes de très grandes envergures, VLSI, (**V**ery **L**arge **S**cale of **I**ntegration). En raison de leur maturité et leur lot impressionnant d'outils, il est plus attrayant de contourner le problème que de tenter de l'enrayer en changeant radicalement de méthodologie de conception. Le problème le plus complexe est celui de la distribution du signal d'horloge pour un budget de puissance donné (Rabaey, Chandrakasan et Nikolic,

2003). À titre d'exemple, il est courant d'utiliser des amplificateurs, qui consomment une puissance importante, pour minimiser les erreurs de phases sur le signal d'horloge. Un problème critique qui, malheureusement, prend de l'ampleur avec la taille des systèmes pour une fréquence donnée. Le même problème se pose pour des circuits de taille modeste, mais de fréquences très élevées.

L'augmentation des fonctionnalités, de la puissance de calcul et d'autres caractéristiques dont les usagers sont friands nécessite l'utilisation de composantes contenant de plus en plus de transistors. En ce sens, la vidéo sur appareil sans fil est un exemple pertinent dans le domaine des télécommunications. Malgré une réduction à l'échelle efficace, ces composantes atteignent des tailles inégalées et des fréquences d'opération toujours plus élevées. De plus, une telle tendance fait naturellement surgir l'idée de faible consommation de puissance pour assurer la durée de vie des piles.

Les longs liens de communication forcent la prise en compte des retards de communication provoqués. Les circuits sont de plus en plus grands et la réduction à l'échelle favorise des composants intégrés plus petits et plus rapides. Dans ces circonstances, le retard dans les portes logiques diminue, alors que celui des liens de communication demeure relativement constant. Ainsi, le retard dans les fils prend de l'importance par rapport à celui dans les portes logiques. Le résultat est que l'analyse temporelle devient plus complexe et l'énergie requise par l'horloge augmente relativement aux autres parties du circuit.

Bref, les circuits synchrones ont été privilégiés au cours des années à cause de leur simplicité de conception. Cependant, il devient de plus en plus évident que de tels systèmes ont des limites importantes.

1.2 Logique asynchrone

Parmi les possibilités de méthodologies de conception, l'ITRS mentionne le retour de la logique asynchrone : une logique qui n'exploite pas la synchronisation globale, mais une synchronisation locale (à l'aide d'un contrôle distribué) entre les modules pour lesquels cette

dernière a une signification fonctionnelle. C'est notamment le cas pour les modules impliqués dans une communication. Dans ces circonstances, les modules possèdent des signaux explicites de synchronisation permettant d'implémenter un protocole de communication de type « poignée de main » (« handshake ») entre modules associés aux transferts d'informations. La figure 1.3 illustre un exemple de communication avec transferts d'informations du type poignée de main.

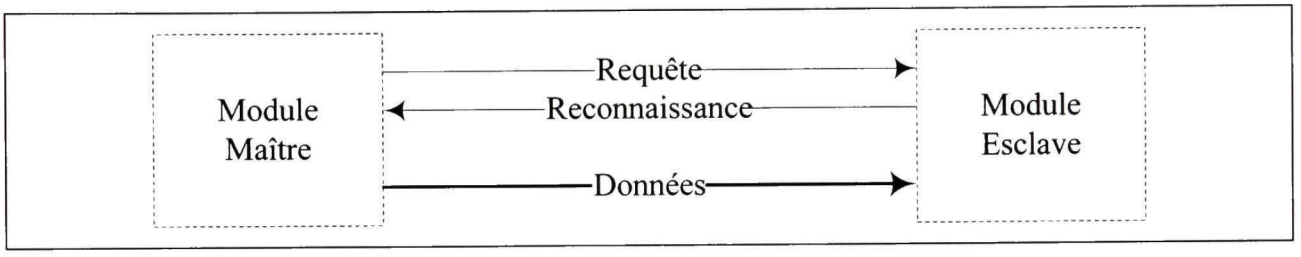


Figure 1.3 Exemple de communication du type poignée de main.

Le module maître signifie au module esclave, par l'entremise du signal de requête, qu'il veut mettre en œuvre un transfert et émettre des données. L'esclave répond à cette demande en activant le signal de reconnaissance lorsqu'il a récupéré les données, ce qui informe le maître de la possibilité d'enlever ou de changer les données. Ce dialogue simple illustre un protocole de communication de type poignée de main. Avant d'entrer dans les détails des protocoles de communication, voyons les forces et les faiblesses de la logique asynchrone.

1.2.1 Forces de la logique asynchrone

Les forces de la logique asynchrone sont bien connues (Furber, 1996; Hassoun, Kim et Lombardi, 2003). Elles correspondent exactement aux difficultés de la logique synchrone. Les enjeux qui favorisent l'utilisation de la logique asynchrone dans les conceptions numériques sont les suivants :

- élimination de l'obliquité d'horloge (« skew »);
- réduction du bruit;
- performance générale moyenne au lieu du pire cas;

- adaptation à l'environnement et aux variations du procédé de fabrication;
- modularité accrue;
- réduction de la consommation de puissance globale.

Le signal de synchronisation fourni sous forme d'horloge globale agit comme un chef d'orchestre. Cette horloge consomme une portion importante de puissance pouvant atteindre 50 % du budget de puissance du circuit, même si elle n'effectue aucun traitement. De plus, chaque transition synchronisée augmente le niveau de bruit et d'EMI dans le circuit à cause de la densité de transitions produites. Avec la réduction à l'échelle des circuits intégrés, ces problèmes de bruit, d'obliquité et de demande en puissance constituent des difficultés supplémentaires de conception. Dans cette optique, la suppression d'un signal global de synchronisation apaise les difficultés de conception des circuits intégrés.

Les performances des systèmes synchrones réfèrent au pire cas possible. Le signal d'horloge doit accommoder l'élément le plus lent du système. Les autres fonctionnent donc en deçà de leurs capacités. En ce sens, la logique asynchrone réfère à des performances moyennes, car seuls les modules impliqués dans un transfert peuvent influer sur les performances. Par exemple, si le module le plus lent d'un circuit est sollicité seulement 10 % du temps, le circuit fonctionnera 90 % du temps à une fréquence plus élevé que celle du module le plus lent. Ainsi, les circuits asynchrones ont un potentiel d'opération supérieur, en moyenne, par rapport aux circuits synchrones.

La logique asynchrone s'adapte mieux aux variations d'environnement et de fabrication, car elle peut être construite pour être insensible aux retards, DI, (**D**elay **I**nsensitive). Dans ces circonstances, toute variation de l'environnement peut nuire aux performances du circuit, mais sans influencer la fonctionnalité. Pour la même raison, la logique asynchrone bénéficie d'une modularité accrue. Plusieurs modules de propriété intellectuelle, IP, (**I**ntellectual **P**roperty) conçus séparément peuvent être assemblés sans vérification a posteriori, car la synchronisation locale des données s'opère automatiquement à fréquence maximale pour une technologie donnée. Finalement, seuls les modules sollicités dans un traitement précis ont besoin d'être actifs. Autrement, les modules sont inactifs et exploitent au maximum les

notions d'économie d'énergie des modes dormants. Ces derniers pourraient également être déconnectés des rails d'alimentation pour accroître le potentiel d'économie d'énergie.

Ces quelques avantages sont les plus importants et les plus cités. Pour l'ITRS, les problèmes reliés à la logique synchrone sont suffisamment importants pour changer de méthodologie de conception. Cependant, la logique totalement asynchrone ne semble pas, à ce jour, être une alternative viable à la logique synchrone. La section suivante met cet aspect en relief, en présentant les difficultés reliées à la logique asynchrone.

1.2.2 Faiblesses de la logique asynchrone

La logique asynchrone est souvent comparée à la logique dynamique, car les problèmes rencontrés sont de même nature. Les principales difficultés sont les suivantes (Furber, 1993; Hassoun, Kim et Lombardi, 2003) :

- les habitudes de conception;
- les outils de conception assistée par ordinateur, CAD, (**C**omputer **A**ided **D**esign);
- les outils de validation, vérification et de test;
- la sensibilité aux phénomènes de courses et aux aléas temporels;

Aujourd'hui, un concepteur n'a pas le choix des armes. Les méthodes les plus simples pour la réalisation de circuits intégrés de quelques millions de transistors sont celles concernant la logique synchrone. La raison est simple. La formation est concentrée sur les conceptions synchrones à cause de leur simplicité et de leur dominance dans l'industrie des semi-conducteurs. Les cours sont principalement fondés sur les conceptions synchrones en insistant sur le fait que les réalisations asynchrones apportent des tracas supplémentaires lors de la conception et de la réalisation. Ce comportement favorise le quasi-monopole de la logique synchrone, et ce, sans considérer les avantages potentiels d'autres formes de réalisation. Les habitudes de conception forment un cercle vicieux où les concepteurs synchrones favorisent les méthodologies de conception synchrones. De fait, les concepteurs sont des experts de la conception synchrone et la disponibilité des outils reflète cette réalité.

Les outils de CAD, de validation, de vérification et de test se sont, de ce fait, multipliés en peu de temps pour la logique synchrone alors qu'en parallèle, les outils de la logique asynchrone ont bénéficié d'un support minimum. En considérant que les outils développés ne peuvent pas facilement s'adapter d'une méthodologie à l'autre, les outils pour la logique asynchrone sont donc très rares. Quelques outils ont quand même été développés au fil des années, mais leur nombre demeure insignifiant.

Finalement, la sensibilité des technologies asynchrones aux phénomènes de course et d'aléas n'a pas aidé l'émergence de cette dernière. La logique synchrone simplifie ce problème dans la mesure où ces effets négatifs peuvent être masqués. Peu importe la valeur ou la variation du signal à l'entrée, les éléments de mémoire (bascules) gardent les sorties stables. De cette façon, lorsque la contrainte temporelle du signal d'entrée est satisfaite, le signal de sortie peut de nouveau produire une valeur correcte et stable en sortie (voir figure 1.2). Un stratagème similaire peut être utilisé pour la logique asynchrone, mais il exige un déploiement d'importance qui se quantifie par une utilisation plus importante de transistors. L'étude du cas de la logique à phase nulle, NCL, (**N**ull **C**onvention **L**ogic) est particulièrement intéressante et significative pour les circuits asynchrones (Martin, Nystrom et Wong, 2003). La technologie asynchrone NCL est dite correcte par construction sans aucun effet de course et d'aléas pour autant que la différence entre des retards concurrents, menant à une même porte logique, soit similaire. Cette caractéristique nommée « isochronic folk » propre au modèle temporel presque insensible aux retards, QDI, (**Q**uasi **D**elay **I**nsensitive) est utilisée pour la logique NCL. L'étude de cas sera présentée après avoir illustré les concepts sous-jacents aux circuits asynchrones. Commençons par les protocoles de communication.

1.2.3 Protocole de communication

Il existe deux méthodes particulièrement répandues pour implémenter un protocole de communication à poignée de main : un protocole à deux phases et un protocole à quatre phases. Étant donnée la nature binaire des signaux, les fronts montants (passage d'un niveau '0' logique à un niveau '1' logique) et les fronts descendants (passage d'un niveau '1' logique à un niveau '0' logique) sont les seuls événements qui peuvent avoir lieu sur les

signaux de requête et de reconnaissance. Naturellement, ces événements se succèdent dans le temps, c'est-à-dire qu'un front montant succédera toujours à un front descendant.

Protocole à deux phases

Le protocole à deux phases est implémenté en considérant que toutes les transitions sont porteuses d'informations (Sparsø et Furber, 2001). Dans ce cas, une transition sur le signal de requête correspond à une donnée valide sur le canal de communication, alors qu'une transition sur le signal de reconnaissance correspond à une opération complétée. Un tel protocole est illustré à la figure 1.4. Il est à noter que la transition de reconnaissance est de polarité opposée (une requête sur front montant est reconnue par un front descendant sur le signal reconnaissance) au signal de requête pour une question d'optimisation du circuit en vue d'une réalisation.

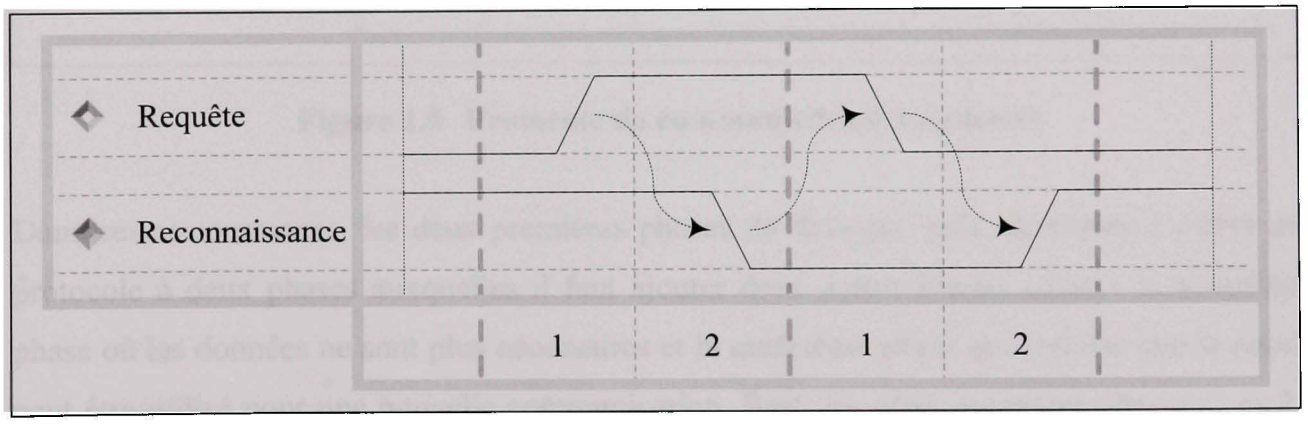


Figure 1.4 Protocole de communication à 2 phases.

Ce protocole possède l'avantage de réduire au minimum les transitions sur les signaux. Cependant, il est très peu utilisé en pratique étant donné la complexité du circuit nécessaire à un protocole qui réagit de la même manière, peu importe la nature du front (front montant et front descendant).

Protocole à quatre phases

Le protocole à quatre phases est une solution au problème du protocole à 2 phases : un seul type de transition est considéré pour chaque événement (fronts montants ou fronts descendants) (Sparsø et Furber, 2001). Cette approche diminue la bande passante du canal, mais assure la symétrie du circuit ce qui simplifie au maximum le circuit de détection des transitions. La figure 1.5 illustre un protocole à 4 phases.

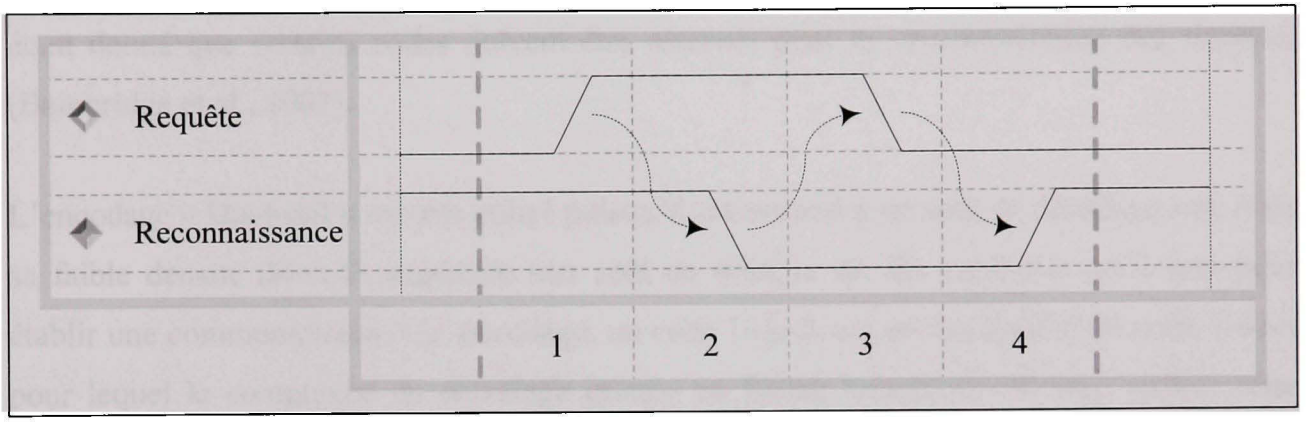


Figure 1.5 Protocole de communication à 4 phases.

Dans ces circonstances, les deux premières phases du dialogue sont identiques à celles du protocole à deux phases auxquelles il faut ajouter deux autres phases soient : la troisième phase où les données ne sont plus nécessaires et la quatrième phase qui signifie que le canal peut être utilisé pour une nouvelle communication. Bref, les deux premières phases (1 et 2, figure 1.5) servent à l'opération et les deux dernières phases (3 et 4, figure 1.5) servent à la remise à zéro des signaux.

Pour la suite de cette thèse, seul le protocole à quatre phases sera utilisé, puisqu'il est le plus exploité et que sa symétrie entre les itérations successives simplifie grandement la réalisation du circuit de détection. Une fois le protocole choisit, il reste plus qu'à déterminer l'encodage des données.

1.2.4 Encodage des données

L'encodage des données est essentiel pour les circuits asynchrones sur lesquels aucune hypothèse temporelle n'est faite. Dans ces circonstances, l'encodage fait référence, dans la totalité des cas, à un encodage incomplet avec un code m-de-n où m représente le nombre de codes utilisés sur une possibilité de n codes. Pour un encodage complet, où m = n, le taux d'efficacité est alors maximum (R=(m/n)=1). Cependant, cet encodage implique un décodage très complexe et il ne s'applique pas à la logique asynchrone sans hypothèses temporelles, étant donné que certains codes doivent être réservés pour la synchronisation des signaux. (Bainbridge et al., 2003).

L'encodage « Dual-rail » est très utilisé puisqu'il est associé à un coût de décodage nul, mais sa faible densité (R=0.5) augmente son coût en nombre de fils parallèles qu'il faut pour établir une communication. Cet encodage, un code 1-de-2, est un cas spécial du code 1-de-n pour lequel la complexité de décodage grimpe en flèche lorsque $n > 4$. Pour pallier cette difficulté, la solution est l'utilisation du code incomplet m-de-n par le jumelage de plusieurs codes simples (1-de-n) pour augmenter la densité du code. De cette façon, il est possible d'augmenter la densité du code sans trop influer sur la complexité de décodage. La solution optimale, performance par rapport au coût des ressources de communications, est atteinte par le jumelage de codes simples 1-de-n. Pour le reste de la thèse, seuls les codes simples 1-de-n seront considérés par souci d'efficacité.

Ce mécanisme de communication permet de rendre le système indépendant aux retards dans les circuits et dans les interconnexions. Cependant, cette synchronisation peut être simplifiée par certaines hypothèses temporelles qui définissent implicitement le mode de fonctionnement et qui permettent de classifier les circuits asynchrones.

Afin d'exposer les concepts sous-jacents aux circuits asynchrones, nous établirons, dans un premier temps, la classification de ces circuits asynchrones par leur mode de fonctionnement. Par la suite, nous résumerons le tout à l'aide d'une étude de cas.

1.2.5 Modèle de circuits asynchrones

Avant d'aborder la classification des circuits, il convient de caractériser le mode de fonction et le type d'interaction du circuit avec l'environnement.

Mode de fonctionnement

Trois modes principaux se distinguent pour les circuits asynchrones :

- le mode fondamental,
- le mode rafale,
- le mode entrée/sortie.

Dans le mode fondamental, le circuit doit être initialement stable (signaux internes et externes). Dans cet état, l'environnement peut modifier un maximum d'une seule entrée et doit attendre que le circuit se stabilise avant de pouvoir enlever cette donnée ou la changer à nouveau. La faiblesse des travaux de Huffman (Huffman, 1964) et Unger (Unger, 1969) sur le mode fondamental est l'impossibilité pour le circuit d'informer l'environnement de la stabilité du système. L'approche utilisée pour contourner le problème est une hypothèse temporelle basée sur le retard le plus grand du circuit. En utilisant une approche de retards bornés, les circuits ainsi réalisés ressemblent fortement à des circuits synchrones.

Le mode rafale constitue une extension au mode fondamental au sens où plusieurs entrées et donc plusieurs sorties peuvent changer d'état entre deux retours à l'état stable. Le modèle de retard demeure toutefois borné.

Le mode entrée/sortie fonctionne sans hypothèse sur les signaux internes. Les signaux de sorties doivent refléter les changements des entrées. Malgré ceci, si la causalité entre les transitions d'entrées et les transitions de sorties sont assurées, les entrées peuvent changer avant que les signaux de sortie n'aient le temps de se stabiliser. Dans un tel paradigme, les circuits DI ainsi créés doivent être composés uniquement d'inverseurs et d'éléments C de

Muller (mémoire asynchrone) (Muller et Bartky, 1959). Bien que limités en fonctionnalité, de tels circuits, dits « insensibles aux retards », existent où l'élément C de Muller supporte les retards non bornés et au cœur de cette recherche. Cet élément sera analysé en profondeur dans les prochains chapitres.

Classes de circuits asynchrones

La classification des circuits asynchrones, illustrée à la figure 1.6, qualifie le compromis robustesse/complexité propre à leur mode de fonctionnement (Vivet, 2001).

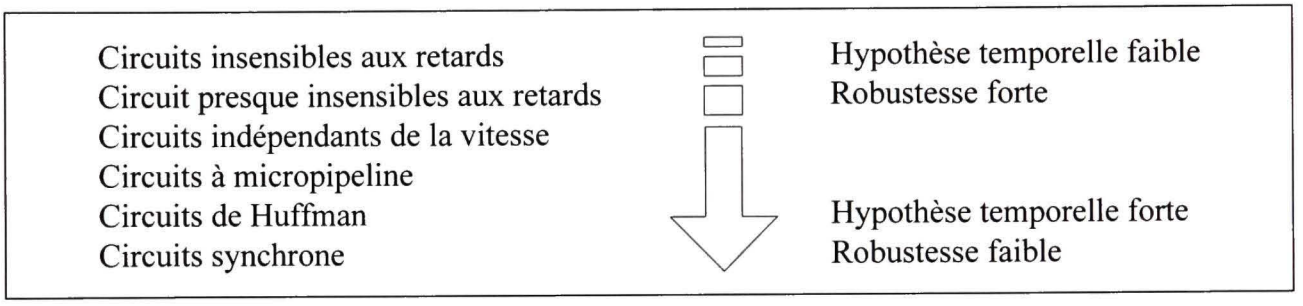


Figure 1.6 Classification des circuits.

Les circuits DI ne font aucune hypothèse temporelle. Ces circuits sont insensibles aux retards non bornés, et ce, aussi bien dans les portes logiques que ceux attribuables aux interconnexions (fils de communication). Ce comportement oblige d'informer explicitement le circuit récepteur que les données sont prêtes pour un traitement subséquent. Étant donné que la majorité des librairies modernes de porte logiques possèdent une seule sortie, ces portes s'avèrent inutilisables dans un tel paradigme où aucune hypothèse temporelle n'est faite. Naturellement, cette approche est très limitative en thème d'opérations réalisables puisque chaque élément devient très complexe.

Pour contourner ce problème, les circuits QDI acceptent une hypothèse temporelle où le retard d'un signal à plusieurs cibles (fourche) doit être égal. C'est-à-dire qu'une transition arrive aux différents points de destination en même temps. Cette hypothèse de fourches isochrones (« Isochronic fork ») permet l'utilisation de portes logiques à une seule sortie, puisqu'en supposant que les signaux se propagent de façon équivalente dans la fourche, il est

possible de reconnaître la fin du traitement à l'aide d'un seul signal. A.J. Martin a démontré que cette hypothèse est la plus faible qu'il est possible d'inclure pour permettre l'utilisation de porte logique à une seule sortie (Martin, 1993). Malgré tout, cette hypothèse demeure faible, car lorsque le temps de retard dans une fourche demeure inférieur au temps de traitement, cette contrainte est satisfaite (Martin, 1990).

Les circuits indépendants de la vitesse, SI, (**S**peed **I**ndependent) utilisent l'hypothèse que les retards dans les fils sont négligeables pour des retards non bornés dans les portes logiques. En considérant l'intégration et la réduction à l'échelle, ce concept est de plus en plus difficile à vérifier intuitivement. Cependant, en étudiant les caractéristiques propres aux circuits QDI et SI, un consensus existe consistant à considérer ces modèles comme équivalents. La figure 1.7 illustre l'équivalence des modèles temporels SI et QDI (Hauck, 1995) où α, β, γ et ε représentent respectivement les retards dans les portes logiques, dans les fils simples et dans les fourches.

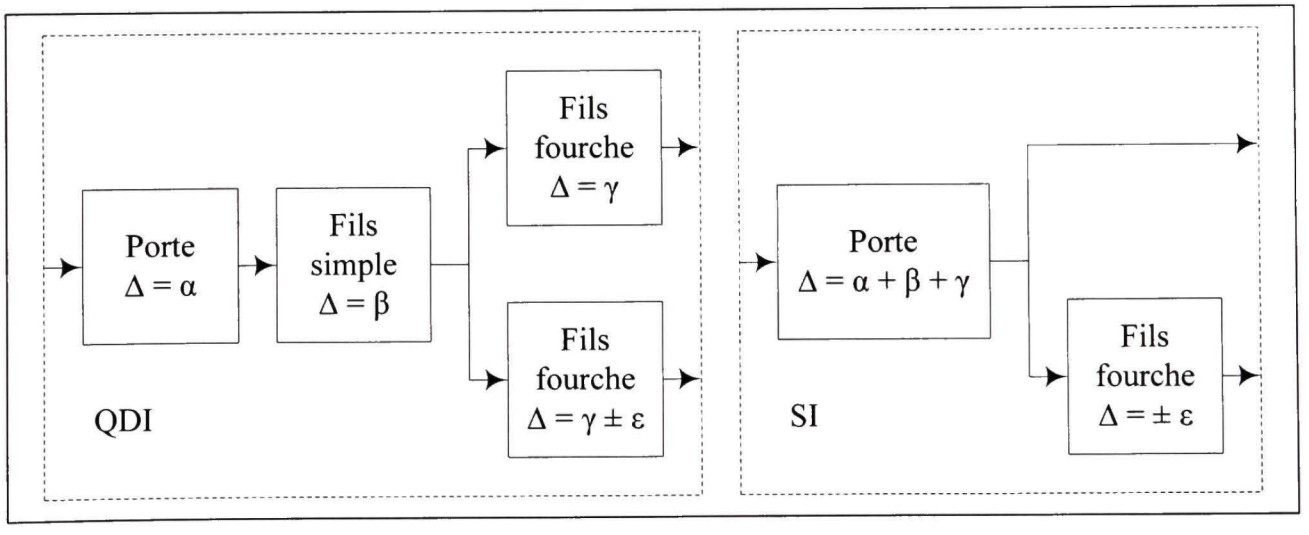


Figure 1.7 Équivalence des modèles temporels SI et QDI.

Naturellement, le modèle QDI est plus populaire que le modèle SI étant donné que le modèle QDI représente mieux le processus de fabrication actuel et qu'il y a beaucoup plus d'information disponible sur ce modèle temporel.

Les micro-pipelines introduits par Ivan Sutherland (Sutherland, 1989) sont des circuits où un chemin de données utilisant un modèle de retard borné (à l'image des circuits synchrones) est commandé par un protocole DI. La figure 1.8 illustre un micro-pipeline.

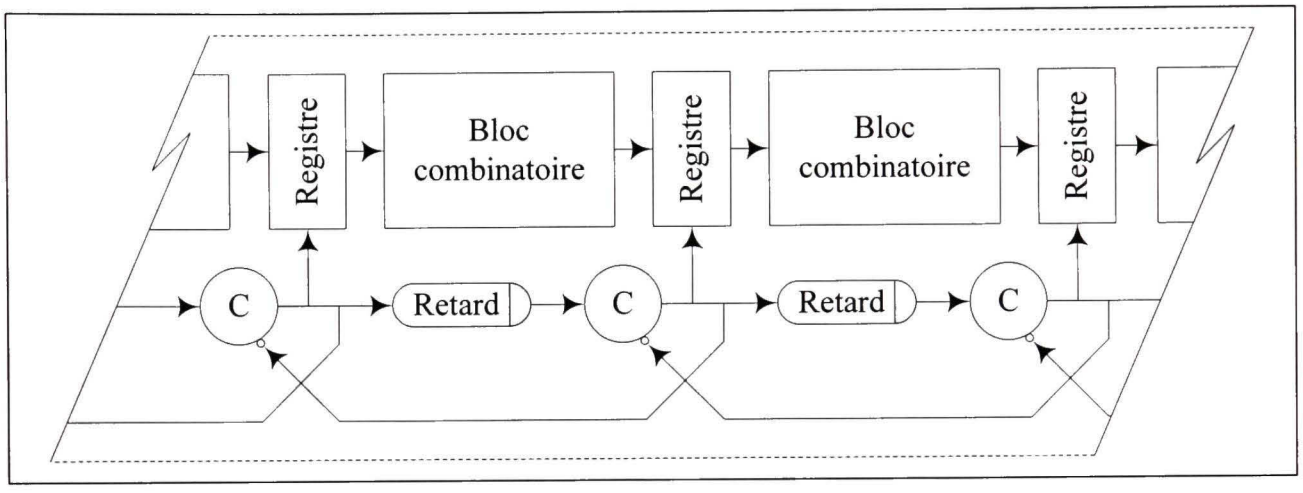


Figure 1.8 Micro-pipeline.

Le micro-pipeline ressemble au pipeline synchrone (figure 1.1) duquel le signal d'horloge a été remplacé par un protocole de communication à poignée de main où les éléments C de Muller sont représentés par des cercles avec l'étiquette « C ». Le micro-pipeline combine les avantages des circuits synchrones pour le chemin de données (outils disponibles) avec certains avantages de la logique asynchrone (disparition du signal d'horloge). Cependant, la conception au pire cas concernant la réalisation des retards nécessite une analyse temporelle, ce qui est superflu pour les modèles SI, QDI et DI.

Les circuits Huffman fonctionnent sous le principe de machine à états finis asynchrone (Huffman, 1964). Malgré l'élimination de l'horloge et l'utilisation de signaux de contrôle locaux s'apparentant à des horloges locales, ils sont pratiquement identiques aux circuits synchrones. Ces circuits sont sujets à des suppositions temporelles qui sont du même ordre que celles des circuits synchrones. Ainsi, ils ont tous les désavantages de la logique asynchrones sans les avantages. De plus, une erreur de conception sur un élément de retard (retard trop court) rend le circuit totalement non fonctionnel.

1.2.6 Étude de cas

La cellule de base NCL s'apparente à l'élément C de Muller, elle est asynchrone, mais elle possède la contrainte supplémentaire d'avoir des retards similaires pour chacune de ses entrées (Isochronic folk) (Bandapati, Smith et Choi, 2003). Cette limitation caractérise les cellules NCL. Une fois regroupées, de telles cellules forment des circuits QDI. Cette hypothèse permet de faire la reconnaissance d'une seule sortie au lieu d'avoir à considérer toutes les sorties.

Un circuit purement DI est, par définition, semi-modulaire. La semi-modularité se définit comme l'impossibilité de changer un état interne avant que cet état se stabilise. L'approximation QDI peut provoquer des violations de la semi-modularité (Taubin, Fant et McCardle, 2002). Cette violation se matérialise par l'apparition d'aléas sur une ou des sorties. Évidemment, ce comportement n'est pas souhaité en logique asynchrone. Cette violation de la semi-modularité survient lorsqu'une sortie possède des liens orphelins (non observables) et qu'elle passe de l'état « activé » à l'état « stable ». Ces liens orphelins n'ont pas de signaux de reconnaissance en sortie (aucune observation) pour un ensemble de signaux d'entrées. Heureusement, la détection d'aléas pour des circuits basés sur des cellules NCL se résume à la détection des chemins orphelins.

La méthodologie NCL permet de formuler des expressions symboliquement complètes contrairement à ce qu'offre la logique booléenne synchrone pour les conceptions asynchrones (Fant et Brandt, 1996). L'horloge globale est remplacée par le contrôle NULL ajouté aux données ('0' et '1') de la table de vérité. Les trois états sont normalement représentés sur 2 fils (dual rail). Ce contrôle permet de créer des vagues successives de données et de contrôle (NULL). Les vagues créées servent à transmettre puis effacer les données pour éviter les collisions et informer la cible de l'arrivée d'une nouvelle donnée.

Les implémentations statiques, semi-statiques et dynamiques de ces cellules en portes CMOS avec seuil et hystérésis simple (Sobelman et Fant, 1998) sont réalisées par des portes du type n-de-m qui doivent avoir « n » entrées avec des données valides « DATA » (autre que

NULL) pour que la sortie actualise sa sortie et qu'elle passe à « DATA ». La valeur de la donnée (« DATA ») dépend naturellement des entrées et de la fonction logique à réaliser. L'hystérésis est, quant à elle, réalisée en permettant à la sortie de retourner à « NULL » uniquement lorsque toutes les entrées redeviennent également « NULL».

Fort d'une bibliothèque de portes logiques, il reste à concevoir des circuits. Malheureusement, la logique asynchrone souffre d'une carence en outil de conception. Les modèles de description matérielle, HDL, (**H**ardware **D**escription **L**anguage) créés dans une optique de conception synchrone sont adaptables à la logique asynchrone, mais beaucoup d'efforts sont nécessaires pour effectuer la transition pour des résultats souvent décevants (Ligthart et al., 2000). Utiliser des outils synchrones existants favorise un apprentissage rapide pour des utilisateurs matures et performants.

L'utilisation du VHDL/Verilog permet d'utiliser le même code pour faire la vérification et la synthèse (Smith et Ligthart, 2001). L'ajout des bibliothèques, pour convertir le code VHDL standard en 3NCL (NULL (N), 1 (T), 0 (F)) jusqu'en 2NCL (dual rail), offre la possibilité d'effectuer la synthèse avec des outils standards.

Dans le but d'automatiser encore plus le processus de création asynchrone, un analyseur de cycle est nécessaire pour la logique NCL (Masteller et Sorenson, 2003). Le cycle est l'unité fondamentale pour la sauvegarde d'information. La méthode consiste à parcourir le fichier contenant la liste des connexions (« netlist ») du circuit pour en extraire les PSN (primary singular node). La méthode permet d'identifier :

- le nombre de portes NCL,
- le nombre de registres,
- le nombre de cycles,
- le nombre de registres de test et
- le temps pris pour les opérations.

Une décomposition adéquate permet l'automatisation du processus de création.

La solution proposée est intéressante en plus d'offrir un bon potentiel. Cependant, les outils et l'environnement de réalisation sont protégés par des droits de propriété intellectuelle, dont des brevets au Canada et aux États-Unis. L'achat de licence est possible. Cependant, rares sont les concepteurs qui vont payer les frais associés à la migration d'une méthodologie synchrone mature vers une méthodologie en émergence avec peu ou pas de façon de réutiliser tout ce qui a été fait. Une solution moins draconienne est celle de la logique GALS présentée à la section suivante.

1.3 GALS

Dans la mesure où la synchronisation globale devient un coût trop élevé à payer pour les conceptions numériques de hautes fréquences ou de grandes surfaces pour les nouvelles technologies et que l'approche asynchrone souffre de graves carences en outils de développement, l'ITRS propose le retour de la logique asynchrone insérée dans le concept GALS. Un paradigme où les communications sont assurées par des liens asynchrones reliant des modules synchrones de tailles restreintes (petites à moyennes) cadencés à l'aide d'une horloge locale.

Une architecture est GALS lorsque s'entrelacent des modules synchrones et asynchrones (Chapiro, 1985). L'agencement de ces modules dépend de l'application alors que l'objectif est de repousser les limites de la logique synchrone où, doit-on le rappeler, le système fonctionne à la vitesse de son élément le plus lent. L'approche GALS consiste à fragmenter une conception en diverses fonctions synchrones où les communications nativement synchrones dégradent les performances et elles sont remplacées par des communications asynchrones. Naturellement, le degré de fractionnement influe sur les performances. Dans ces circonstances, il ne fait aucun sens de segmenter la conception jusqu'aux éléments primaires, car le coût des communications asynchrones et la détection de la fin de cycle de la logique asynchrone s'avèrent rapidement prohibitifs. Le degré de segmentation est donc critique et dépend de l'application. De façon générale, la tendance consiste à minimiser le nombre de liens asynchrones pour exploiter au maximum les méthodologies synchrones.

De plus, comme la logique asynchrone doit être synchronisée, la question quasi inexistante d'état métastable en logique asynchrone refait surface pour la logique GALS. Les sections suivantes présentent les principaux concepts associés à la logique GALS.

1.3.1 File de mémoire FIFO

Matzke a évalué que pour un circuit d'un milliard de transistors réalisé à l'aide d'une technologie de 100nm, seulement 16 % de la surface est atteignable en un cycle d'horloge (Matzke, 1997). Ainsi, la distribution d'une horloge globale est à proscrire puisqu'elle cause également des problèmes de distorsions du signal d'horloge (Semeraro et al., 2002). Une méthode simple pour relier plusieurs domaines d'horloge de tensions et de fréquences différentes et obtenir un meilleur rendement énergétique est une simple file d'éléments de mémoire du type premier entré, premier sorti, FIFO, (**F**irst **I**n, **F**irst **O**ut) possédant deux ports (un port pour la lecture et un autre pour l'écriture). La solution impose des FIFO rapides qui peuvent donner leur statut à l'intérieur d'un cycle d'horloge. Cette contrainte permet des accès à chaque cycle sans conflits possibles pour les données. Le prix à payer pour cette technique est le temps de communication. Cette solution possède un bilan énergétique intéressant, mais souffre de lacunes au niveau des performances.

Les communications asynchrones sont utiles pour relier plusieurs domaines d'horloge par une interface synchrone utilisant les FIFO asynchrones (Lines, 2004). Par contre, l'élimination d'état mort (« dead lock ») provoque une perte de performance. Cette dernière peut, toutefois, être atténuée par un traitement en rafale.

Bref, la file de mémoire FIFO est une solution simple et rapide, mais elle est également une solution qui peut présenter des inconvénients au niveau de la taille et de la latence du circuit. Ces derniers peuvent être minimisés, mais ne peuvent pas être éliminés. Le mode de transfert par micro-pipeline semble être une méthode plus avantageuse que le simple FIFO.

1.3.2 Micro-pipeline

Le micro-pipeline est le mode de transfert le plus utilisé pour les architectures GALS, c'est particulièrement vrai pour les systèmes sur puce, SoC, (**S**ystem **o**n **C**hip) (Liljeberg, Plosila et Isoaho, 2003). À titre d'exemple, une implémentation GALS pour les micro-pipelines est comparée en matière de consommation de puissance à son homonyme synchrone (Rapaka et Marculescu, 2003). Deux solutions GALS (avec ou sans FIFO) sont comparées à une implémentation synchrone. Dans le but d'avoir une comparaison cohérente, la synchronisation des données pour masquer la métastabilité est réalisée à l'aide d'une simple redondance des bascules d'entrées (Bit valide). Les résultats montrent que les deux implémentations GALS supplantent l'implémentation synchrone et que l'implémentation GALS sans FIFO donne de meilleurs résultats au chapitre de la consommation de puissance à cause de la complexité supplémentaire de la mémoire de type FIFO.

Bref, les systèmes GALS sont définis par l'utilisation des logiques synchrones et asynchrones où les îlots synchrones sont liés entre eux par des liens asynchrones. La solution d'un micro-pipeline simple est généralement privilégiée à celle de la file de mémoire FIFO pour des questions de performance, mais l'application peut également influencer ce choix. De plus, dans l'optique d'une implémentation par micro-pipeline, les liens de communication asynchrone utilisant un protocole de communication à 4 phases sont les plus adaptés sur le plan de la complexité et de la consommation de puissance. Cependant, dans tous les cas, la notion d'état métastable rend la synchronisation risquée.

La section suivante illustre les cas typiques et l'évolution des générateurs du signal d'horloge de façon à proposer une solution viable et simple pour fournir une horloge dont la fréquence et la phase peuvent être ajustées pour masquer les risques d'états métastables.

1.3.3 Générateur d'horloge

Dans un contexte GALS, certains signaux asynchrones doivent être synchronisés afin d'interagir avec un domaine d'horloge. Les méthodes de synchronisation ont évolué jusqu'à

permettre le masquage complet des états métastables qui nuisent au fonctionnement d'un circuit synchrone. Cette progression a nécessité plusieurs itérations.

Avec états métastables

Les boucles à verrouillage de phase, PLL, (**P**hase-**L**ocked **L**oop) sont vulnérables à l'environnement en plus d'être coûteuses sur le plan de la puissance et de la surface. De plus, les PLL ont normalement besoin d'un temps de stabilisation assez long. Ainsi, elles ne sont pas propices à une application GALS. Une solution est d'utiliser une horloge de référence, un oscillateur en anneau, et une unité de contrôle (Nilsson et Torkelson, 1996). L'unité de contrôle permet de générer la fréquence désirée (le nombre de cycles voulus) par rapport à l'horloge de référence. L'avantage de cette solution est de pouvoir être réalisée avec une bibliothèque CMOS standard, mais la taille des transistors et le nombre d'inverseurs de l'oscillateur doivent être ajustés manuellement.

Pour résoudre ces problèmes, un simple FIFO asynchrone peut interconnecter un module synchrone à un module asynchrone dans un contexte de SoC (Yun et Dooply, 1999). Cette méthode est plus sûre et induit moins de latence que l'utilisation des deux bascules nécessaires à la réduction des probabilités de métastabilité lors de la synchronisation (force brute). L'approche est intéressante, mais une communication fiable et une faible latence sont atteintes au prix d'une limite de la bande passante d'environ 12 %.

Il est alors facile de constater que les systèmes GALS n'ont pas une méthode de conception unique. Une autre solution propre aux systèmes GALS, utilisant les résultats précédents, est un multiplicateur d'horloge qui peut changer la fréquence du signal d'horloge (Olsson et al., 2000). Malheureusement, la méthode est basée sur un signal d'horloge basse fréquence où la phase du signal d'horloge généré est imprévisible. Ainsi, la métastabilité demeure possible. Malgré tout, cette solution demeure intéressante pour les circuits de faible puissance et de petite taille (faible volume).

L'évolution suivante consista à réaliser une horloge pouvant être ajustée ou arrêtée de façon à éviter les états métastables ou pour économiser de l'énergie (Moore et al., 2000). Un état métastable est quand même possible à cause de l'utilisation d'un arbitre. Le mécanisme d'ajustement d'horloge est basé sur une ligne à retards variables à auto calibrage. Le retard est constitué de deux éléments : un court et un long (environ 10 fois plus grand). Ceci permet d'ajuster les retards pour générer une fréquence d'horloge précise. Les résultats montrent que la stabilité de l'horloge générée est semblable à celle d'un PLL, en plus d'être faite d'une façon numérique et totalement ajustable.

Même si chaque itération apporte une nouvelle innovation dans le domaine, les états métastables demeurent un problème non résolu. La section suivante explique comment les travaux ont évolué pour masquer les états métastables d'un générateur d'horloge.

Sans états métastables

Il faut attendre quelques années pour avoir une solution qui masque les états métastables (Shengxian et al., 2002). Cette proposition est une amélioration des travaux de (Yun et Dooply, 1999), car elle permet l'élimination du module exclusion mutuelle, MUTEX, (**MUT**ual **EX**clusion) pour la conception du contrôle d'un système GALS. La puissance est prise en compte puisque la solution utilise une nouvelle topologie de porte logique qui réduit les pertes de puissances. Par contre, l'application de cette méthode est sujette à 3 contraintes pour masquer la métastabilité :

- les requêtes d'information sont toujours demandées par un module maître (W-port) et les données sont toujours acceptées par un module esclave (R-port);
- lorsque le module maître (W-port) active ses sorties, son horloge interne est arrêtée, elle sera réactivée uniquement par la reconnaissance du module esclave (R-port);
- les modules maîtres (W-port) et esclaves (R-port) sont indépendants.

Ces contraintes permettent d'éliminer un nombre d'états du protocole de communication et donc de masquer la métastabilité. Cette solution donne des circuits simples dont la forme de

l'horloge peut être déformée. Cette déformation est assurée par un oscillateur en anneau fait à l'aide d'un élément C de Muller, soit une solution couramment utilisée.

Dans la même optique, une architecture capable de multiplier ou de diviser la période d'une horloge en un coup d'horloge est proposée (Boyer et al., 2004). L'approche considère X horloges à la même fréquence avec un déphasage de 1/X de cycle pour la génération du signal d'horloge. Les avantages sont :

- la fréquence peut être ajustée à n'importe quelle résolution en un coup d'horloge;
- l'horloge n'est pas arrêtée, il n'y a donc pas de retard de réactivation;
- la synthèse d'une période d'horloge variable, VPCS, (**V**ariable **P**eriod **C**lock **S**ynthesis) permet la division et la multiplication de son horloge de référence.

Le fonctionnement est simple. Le mot d'entrée représente le multiplicateur en point fixe. Les fronts déphasés des horloges sont comptés et un masque laisse passer uniquement les fronts correspondants à la fréquence souhaitée. Il est à noter qu'en fonction de la fréquence de référence, les plus petites valeurs du mot d'entrée peuvent ne pas être possibles (parallélisme de fronts pour masquer les conditions de métastabilité). Cependant, toutes les autres valeurs sont possibles et permises. Une telle solution nécessite beaucoup plus de transistors.

Bref, un bon générateur d'horloge doit fournir un signal de synchronisation stable et rapide tout en requérant peu de volume et de puissance. Il est clair qu'un tel module est un outil utile et même nécessaire pour obtenir une bonne performance pour les systèmes GALS. Ce qu'il faut retenir c'est que les solutions qui masquent les états métastables nécessitent l'usage de contraintes. Dans certains cas, les contraintes sont implicites aux circuits et rendent leurs respects transparents lors de l'utilisation. Lorsque cette approche n'est pas possible, les conditions d'utilisation permettent le respect des contraintes. Ces concepts seront exploités aux chapitres suivants soit ceux de l'architecture de FPGA GALS et du masquage des aléas temporels.

1.4 Discussion

La logique synchrone possède effectivement beaucoup de qualités. La simplicité de vérification et le masquage des effets de courses et d'aléas temporels sont les principales. Cependant, elle possède aussi des limitations. Certaines de ces limitations obligent le développement de nouvelles méthodes de conception. L'énergie dissipée, les problèmes de distorsions du signal de synchronisation et les performances sont des éléments suffisants pour motiver l'élaboration d'une nouvelle méthodologie.

D'un autre côté, la logique asynchrone offre beaucoup de solutions aux problèmes énoncés, mais elle en crée d'autres, et ce, sans compter le manque flagrant d'outils de conception. Ainsi, la solution consiste à créer une nouvelle méthode de conception qui sait tirer avantage de la logique synchrone et de la logique asynchrone, sans être trop influencés par leurs effets négatifs.

La présente revue de littérature est générale. Dans les faits, chacun des chapitres (les composants, l'architecture et le masquage des aléas temporels) présentera une revue de littérature plus exhaustive traitant des aspects importants et relatifs à chacun d'eux.

CHAPITRE 2

Méthodologie d'implémentation de composants GALS sur FPGA

2.1 Introduction

La distribution d'un signal d'horloge possédant une faible erreur de phase est de plus en plus difficile à réaliser à mesure que les circuits intégrés deviennent plus rapides et plus particulièrement lorsqu'ils augmentent en surface (Matzke, 1997). Les FPGA commerciaux sont également confrontés à cette problématique. À titre d'exemple, le FPGA XC5VLX50-FF676AGU0617, un « petit » FPGA de la famille Virtex-5 de Xilinx, atteint une surface de gaufre de 132.2 mm^2 (Chipworks, 2008). Les perspectives d'avenir n'ont rien d'encourageant puisque l'ITRS prévoit des gaufres de FPGA de plus de 800 mm^2 d'ici 2011. Au même moment, les circuits à multiples gaufres « Waffer scale » devraient être assez matures pour limiter la taille de ces composantes en exploitant les réalisations en trois dimensions. Cependant, même si le « waffer scale » résolvait le problème de surface, la contrainte de puissance dissipée continuera à prendre de l'importance. Ces perspectives font de la logique asynchrone ou, du moins, des communications asynchrones, une alternative intéressante. Cette option facilite la distribution de signaux locaux d'horloge pour des circuits de grande taille. Elle favorise également la diminution d'émission d'ondes électromagnétiques, la diminution de la consommation de puissance et l'augmentation de la réutilisation des composants utilisés dans des systèmes sans horloge. Cependant, le manque de connaissances, d'outils et de technique, à cet égard, demeure les principaux obstacles, selon l'ITRS, à la mise au point d'une méthodologie mature de conception asynchrone.

Parmi les solutions, l'ITRS propose la logique GALS introduite par Chapiro (Chapiro, 1985) comme solution intermédiaire. Il existe deux approches pour faciliter la transition des conceptions synchrones pures aux conceptions GALS pour FPGA. Intuitivement, la première consiste à développer une nouvelle architecture qui supporte nativement les transferts asynchrones (Payne, 1996; Teifel et Manohar, 2004). Bien que des solutions puissantes et efficaces de ce type aient été proposées, celles-ci demeurent complexes à utiliser en raison de

leur dépendance aux nouvelles connaissances et techniques, mais également aux nouveaux outils. Cette approche sera, tout de même, abordée au prochain chapitre.

La seconde approche consiste à appliquer des méthodes asynchrones directement aux FPGA synchrones existants. La logique à retard de phase (« Phased logic ») dans un contexte synchrone (Traver, Reese et Thornton, 2001) est un exemple commun de cette approche. Cependant, des réglages précis sur la phase des signaux sont nécessaires pour assurer la qualité des transferts mésochrones. Pour des conceptions de grande surface, ces réglages de phase sont à ce point nombreux, qu'ils s'avèrent prohibitifs. Une application plus prometteuse de cette deuxième approche est basée sur l'implémentation de composants asynchrones comme l'élément C de Muller (Quoc Thai et al., 2002; Xin, Tapani et Jari, 2006) ou l'arbitre (Najibi et al., 2005) dans des circuits synchrones utilisant les ressources existantes. Cette application est avantageuse, car elle enlève la nécessité d'avoir recours à des changements architecturaux. Ce chapitre présente une méthodologie de ce type. Il est à noter qu'une telle méthodologie est essentielle, car le flot de conception habituel pour les FPGA synchrones peut mener à des erreurs fonctionnelles (Quoc Thai et al., 2002). Ainsi, nous présenterons les performances et limitations fonctionnelles de trois composants typiques employés pour implémenter une conception GALS avec les ressources synchrones d'un FPGA. De plus, nous comparerons différentes versions possibles de ces composants autorisant les opérations GALS sur FPGA.

Ce chapitre est divisé en cinq sections. La seconde section examine les particularités des conceptions GALS, l'architecture des FPGA commerciaux synchrones visés et notre méthodologie basée sur une approche par consignes. La troisième section présente l'implémentation d'une ligne à retard, d'un élément C de Muller et d'un arbitre. La quatrième et la cinquième section présentent respectivement l'implémentation d'une horloge qui peut être arrêtée, PCC, (**P**ausing **C**lock **C**ircuit) comme exemple d'application, ainsi que les conclusions et l'orientation des prochains chapitres.

2.2 Logique GALS pour FPGA synchrone

Dans cette section, nous présentons les caractéristiques générales et communes des FPGA relatives à nos objectifs de communication GALS.

2.2.1 Lacunes de synchronisation

Les systèmes synchrones doivent, pour fonctionner, travailler avec des signaux stables avant et après la transition du signal de synchronisation, le signal d'horloge (Friedman, 2001). L'information à enregistrer doit être présente durant un laps de temps déterminé avant la transition de l'horloge (temps de préparation) et elle doit demeurer inchangée pour un certain temps après ladite transition (temps de maintien). La figure 2.1 illustre ces temps qui peuvent être regroupés pour former une zone défendue où le signal doit demeurer constant pour assurer, par construction, la viabilité des transferts d'information.

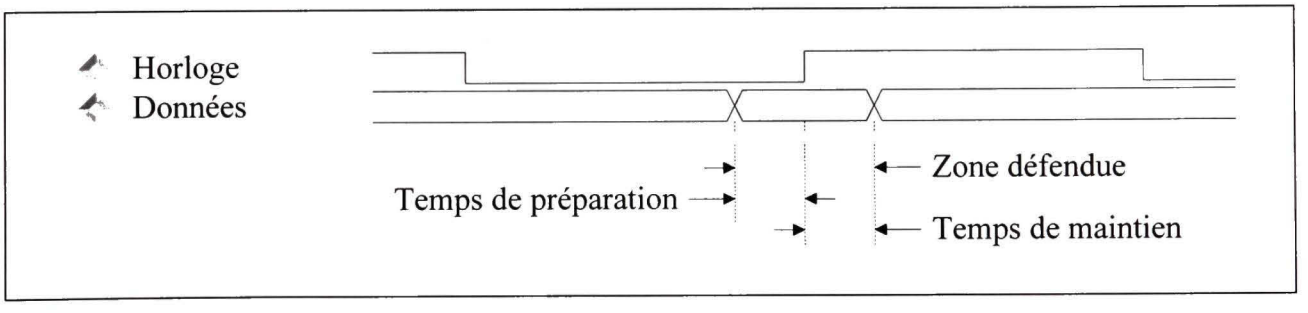


Figure 2.1 Zone critique pour la synchronisation.

Les FPGA commerciaux sont des circuits construits autour de cette approche synchrone. Dans un contexte asynchrone, ces FPGA présentent des défis de conception (Najibi et al., 2005; Quoc Thai et al., 2002; Xin, Tapani et Jari, 2006). L'absence de référence globale met de l'avant la nature événementielle des opérations et se traduit par l'inapplicabilité de l'analyse simple des circuits synchrones. Il s'agit d'aspects importants sur lesquels nous reviendrons lors de l'implémentation des composants GALS.

2.2.2 Particularité des conceptions GALS

Les systèmes GALS utilisent les communications asynchrones entre les îlots synchrones. Chaque liaison assure la causalité événementielle par une synchronisation locale basée sur un protocole de poignée de main. La difficulté majeure des conceptions GALS réside à l'interface synchrone/asynchrone. La métastabilité peut apparaître durant la synchronisation des données lorsque la cadence du récepteur n'est pas mise en relation par rapport à celle de l'émetteur pour la synchronisation du transfert (Sparsø et Furber, 2001). Pour toutes informations complémentaires concernant les particularités de la logique asynchrone (les protocoles, le cryptage de données et autres), le lecteur intéressé trouvera une revue de la littérature particulièrement étoffée dans (Rigaud, 2002).

Une interface GALS, possédant des ports de communication et un générateur d'horloge qui peut être arrêté, nécessite très peu de nouvelles connaissances et de composants GALS pour être implémentée à l'aide d'un FPGA synchrone. Du point de vue de la complexité, cette approche rivalise avantageusement contre celle basée sur des files du type FIFO ou d'autres formes d'implémentation. Les ports de communication peuvent être facilement mis en application avec les ressources standards des FPGA (Xin, Tapani et Jari, 2006). Cependant, le générateur d'horloge a besoin d'au moins une ligne à retard et d'un élément C de Muller. Les arbitres ne sont pas essentiels, mais puisqu'ils simplifient les ports de communication et circonscrivent la métastabilité, ce composant sera également présenté.

2.2.3 Architecture du FPGA Virtex-II

L'architecture du FPGA doit être connue pour exploiter au maximum ses ressources. Malgré les particularités de chaque FPGA, certains principes de base demeurent les mêmes. Sans perte de généralité, l'architecture XC2V de Xilinx a été employée, car la plupart des résultats disponibles dans la littérature sont basés sur cette famille (Alfke, 2002). Il convient de noter que la méthodologie présentée dans ce chapitre pourrait être facilement modifiée pour être utilisée avec d'autres familles de FPGA (Xilinx, Altera, etc.). La figure 2.2 illustre une vue simplifiée d'une tranche, CLB, (**C**onfigurable **L**ogic **B**loc) du Virtex-II XC2V4.

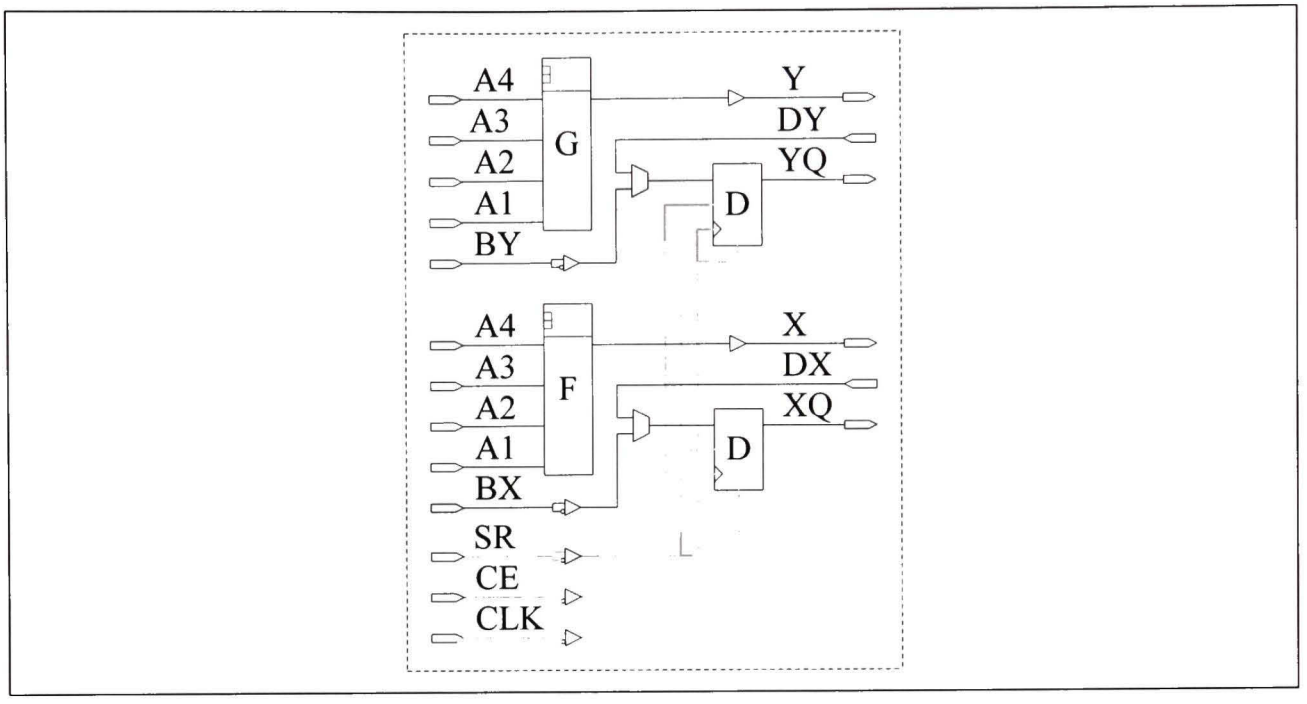


Figure 2.2 Vue simplifiée d'une tranche du Virtex-II XC2V4.

Il existe plusieurs types de ressources dans un FPGA. Les ressources qui n'apparaissent pas sur la figure 2.2 ne présentent aucun, ou peu, d'attrait pour la logique asynchrone. C'est le cas pour la chaîne de retenue et les multiplicateurs dédiés (MULT). C'est également le cas pour les tampons à trois états, la matrice d'interconnexion et les ports d'entrées/sorties, I/O, (**I**nput/**O**utput). Ces ressources n'ont pas été développées dans un contexte GALS. Il convient toutefois de noter que ces ressources sont utiles et même nécessaires pour les modules synchrones d'un système GALS, mais pour l'implémentation de composants asynchrones, elles sont inutilisables ou inefficaces. Les modules de gestion d'horloge numérique (DCM) et les modules de mémoire à deux ports (SelectRAM) pourraient être intéressants pour gérer la phase des horloges ou construire des files du type FIFO. De tels outils supporteraient des transferts GALS. Cependant, leur quantité relativement limitée rend ces solutions inutilisables dans une multitude de situations. Les ressources qui apparaissent sur la figure 2.2, sont présentes en nombre suffisant pour implémenter des composants asynchrones : les modules de logique programmable, LUT, (**L**ook-**U**p **T**able) F et G et sous certaines conditions les registres (bascule D).

La conception de circuits asynchrones basés sur des ressources de FPGA soulève plusieurs difficultés (Hauck et al., 1994; Payne, 1996) :

- les ressources LUT ne peuvent garantir une opération sans transition inopinée à cause des retards de routage et des changements de signaux imprévisibles;
- les suppositions temporelles sont difficiles à garantir dans des FPGA (dues au routage);
- les ressources LUT n'ont pas été créées pour gérer l'arbitration des requêtes.

Malgré ces difficultés, la LUT est construite de façon à avoir le même temps de génération indépendamment des transitions des signaux d'entrées. Dans ces circonstances, la LUT permet une implémentation efficace de composants asynchrones lorsque ceux-ci sont implémentés dans le mode asynchrone fondamental (un seul changement est permis aux entrées lorsque les états internes et les sorties sont stables) et dans quelques autres modes (Quoc Thai et al., 2002).

Les registres pourraient, une fois connectés en chaîne, être utilisés pour effectuer la tâche de synchroniseurs. Cependant, même si une telle approche diminue la probabilité d'erreur de métastabilité, elle ne l'annule pas. En dépit de ceci, les registres demeurent intéressants, car leur circuit interne, ayant un gain élevé, permet une résolution rapide de métastabilité, à un point tel que le retard dû à la métastabilité peut être ignoré sans risque pour des signaux en dessous de 200 MHz sur la famille de XC2VP FPGA (Alfke, 2002).

Les LUT et les registres sont groupés dans des tranches pour les architectures XC2V (deux registres, deux LUT et quelques éléments logiques liés à la chaîne de retenue). Les ressources d'une tranche laissées inutilisées par une application donnée ne sont pas verrouillées et demeurent disponibles pour d'autres applications.

2.2.4 Indexation des ressources

La méthodologie d'implémentation de composantes asynchrones est basée sur un placement et un routage, P&R, (**P**lace & **R**oute) stricts par consignes. Pour assurer la fonctionnalité

desdits composantes sur une architecture donnée, il est nécessaire que ces consignes de P&R respectent l'indexation des ressources.

Malgré le nombre limité des ressources disponibles dans un FPGA, ces ressources ne sont que très rarement uniques. Il est donc indispensable de pouvoir les différencier et de les indexer. À cet effet, la méthode courante d'indexation des ressources est un simple index linéaire par rapport à un point d'origine (Xilinx, 2005). La figure 2.3 illustre l'organisation générale des tranches dans un FPGA.

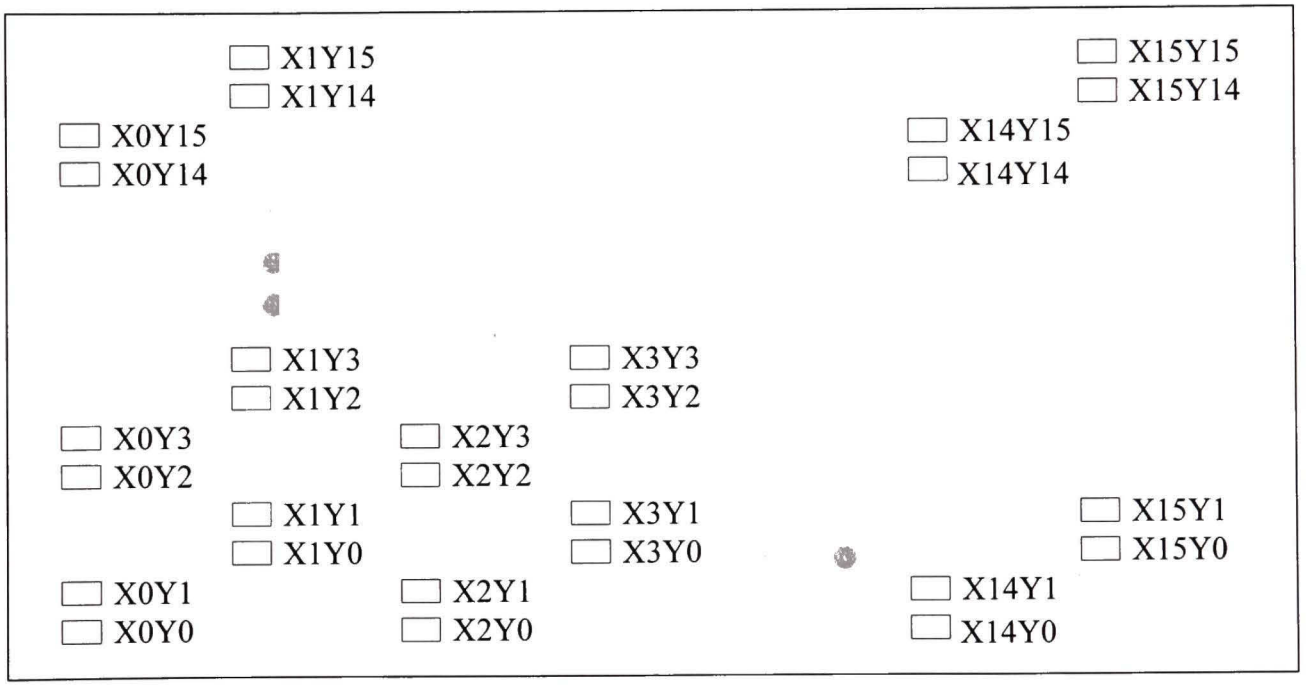


Figure 2.3 Disposition générale des tranches du Virtex-II XC2V4.

Concernant l'architecture XC2V, les ressources sont référencées par un index XY en fonction des tranches. L'origine se situe dans le coin inférieur gauche du FPGA où l'index X incrémente selon l'axe horizontal droit et l'index Y incrémente vers le haut selon l'axe vertical. Cette indexation ne correspond pas à des ressources précises (LUT, registre, etc.), mais à l'index des tranches de FPGA, il est donc généralement préférable de laisser l'outil de P&R décider quelles ressources employer, à l'intérieur d'une tranche, pour minimiser les retards imputables au routage. Les ressources de routage internes sont tellement spécialisées et limitées qu'il est rarement possible de faire un placement manuel complet.

2.2.5 Routage des composantes

Les FPGA sont pourvus de deux mécanismes de routage généraux : les interconnexions locales et les interconnexions globales. Dans un contexte général, les interconnexions globales réservées aux signaux d'horloge à faibles retards ne peuvent pas être utilisées efficacement à d'autres fins dues à leur nombre restreint. Ainsi, ces ressources particulières ne seront pas considérées, seules les interconnexions pour le routage des signaux le seront. La figure 2.4 illustre le mécanisme général de routage d'un Virtex-II.

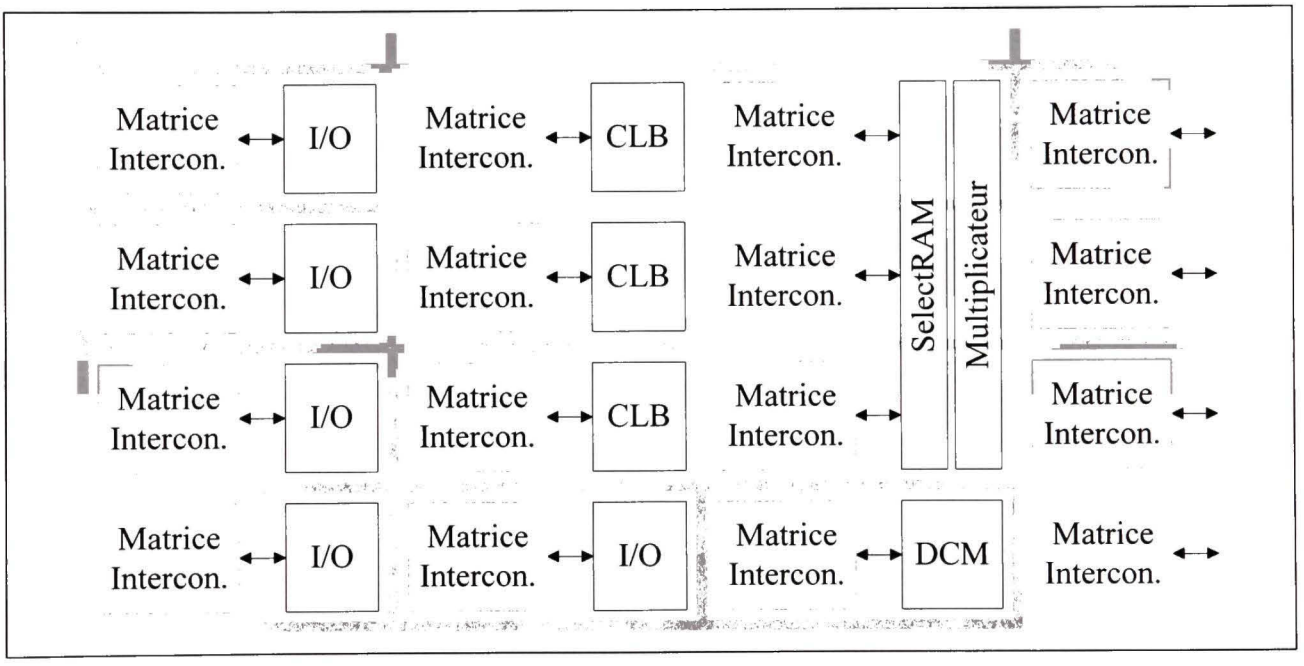


Figure 2.4 Coin inférieur gauche de la matrice de routage du Virtex-II XC2V4.

Une matrice de routage symétrique offre des avantages. La rapidité et l'efficacité de routage n'en sont que quelques-uns. En fait, la régularité sous-jacente à la matrice d'interconnexion favorise, par sa prédictibilité, de bonnes performances générales dans la mesure où la majorité des signaux passent par des ressources globales d'interconnexions, soit les liens (trait large) horizontaux et verticaux. Ces liens agissent comme des autoroutes entourant et reliant les matrices d'interconnexions locales (figure 2.4).

En regard aux applications synchrones, cette structure est appropriée. Le routage a relativement peu d'effet dans la mesure où il est possible de limiter les retards de synchronisation. Dans le cadre d'une implémentation de composants GALS à l'aide de ressources synchrones de FPGA, l'objectif est différent : les retards de communication doivent être minimisés (Van Berkel, 1992). Ces derniers ont un impact sur le composant à créer. Sachant que les retards engendrés par les liens de communication sont directement proportionnels à la distance à parcourir, il est essentiel d'éviter, autant que possible, les liens globaux, car ils présentent de plus grands retards. Ainsi, pour optimiser les composants GALS, les liens de communication dédiés à l'intérieur des tranches et des matrices d'interconnexion seront favorisés.

2.2.6 Consignes pour les FPGA

Les consignes pour les FPGA sont nécessaires à la création de composants de base asynchrones puisque les outils de synthèse pour FPGA enlèvent, par le processus d'optimisation, les composants qui semblent ne pas effectuer de travail utile. L'exemple classique est la chaîne d'inverseurs (nombre impair) remplacée par un unique inverseur. Cependant, pour certains composants asynchrones, cette logique « superflue » peut être utile, voire indispensable, c'est pourquoi l'utilisation des contraintes spécifiant explicitement les consignes aux différents outils permet de préserver certains modules du processus d'optimisation. Ainsi, dans cette section, nous passerons en revue les contraintes des FPGA pour l'implémentation de circuits asynchrones tout en ciblant un sous-ensemble spécialisé à cette tâche. Il est à noter que les contraintes revues dans ce chapitre sont celles des FPGA de la famille Virtex-II utilisant l'outil de synthèse XST. Chaque constructeur propose un ensemble dédié de contraintes. De plus, la méthode présentée dans cette section peut être généralisée pour n'importe quelle famille de contraintes de FPGA synchrones.

Contraintes des FPGA Virtex-II

Les contraintes du Virtex-II peuvent être regroupées en six catégories comme l'illustre le tableau 2.1 (Xilinx, 2005). Il est à noter que ce tableau n'est pas exhaustif puisqu'il est basé

sur le guide des contraintes de la version 8.1i du gestionnaire de projet de Xilinx (ISE). Les autres contraintes sont généralement présentées sous forme de paramètres, c'est pourquoi nous ne nous intéresserons pas à ces paramètres à l'exception de LUT_MAP. C'est un paramètre qui fut considéré jadis comme une contrainte, sur lequel nous reviendrons.

Tableau 2.1 Contraintes du Virtex-II

Transcription de la description matérielle (« mapping ») / groupe		
AREA_GROUP	COMPGRP	SAVE NET FLAG
BLKNM	HBLKNM	XBLKNM
H_SET	HU_SET	U_SET
KEEP		
Placement		
BEL	Directed Routing	LOC
LOCATE	RLOC	RLOC_ORIGIN
RLOC_RANGE	USE_RLOC	PROHIBIT
Synchronisation		
ASYNC_REG	DROP_SPEC	FROM-THRU-TO
FROM-TO	MAXDELAY	MAXSKEW
OFFSET	PERIOD	PRIORITY
TIG	TIMEGRP	TIMESPEC
TNM	TNM_NET	TPSYNC
TPTHRU	TSidentifier	USELOWSKEWLINES
Assembleur (« fitter »)		
WIREAND	COLLAPSE	MAXPT (CPLD)
NOREDUCE	PWR_MODE	REG
Entrées / sorties		
BUFG (CPLD)	CONFIG	CONFIG_MODE
COOL_CLK	DATA_GATE	DCI_VALUE
DRIVE	FAST	FEEDBACK
FLOAT	KEEPER	INREG
IOB	IOBDELAY	IOSTANDARD
LOCK_PINS	NODELAY	OPEN_DRAIN
PIN	PULLDOWN	PULLUP
SLOW	SCHMITT_TRIGGER	
Fonctions globales		
DISABLE	ENABLE	FILE
MAP	OPT_EFFORT	OPTIMIZE
SLEW	SYSTEM_JITTER	TEMPERATURE
VOLTAGE	VREF	

Les contraintes ombragées sont celles qui présentent un attrait particulier pour créer des composants GALS (la section 2.3 présentera ces composants), mais avant de cibler ces contraintes particulières, révisons d'abord l'ensemble des contraintes.

Les contraintes liées aux fonctions globales, à l'assembleur et aux entrées/sorties sont sans intérêt pour la création de composants asynchrones, puisqu'elles servent à spécifier la nature ou le mode d'opération d'éléments synchrones. Dans la partie synchronisation, la contrainte ASYNC_REG semble naturellement intéressante. Malheureusement, elle n'est d'aucune utilité pour nos besoins, attendu qu'elle provoque l'implémentation d'un verrou transparent qui possède, à l'instar du registre, des temps d'enregistrement et de maintien incompatibles avec la logique asynchrone. Les autres contraintes de cette catégorie sont liées à l'analyse temporelle. Ainsi, dans le contexte d'une conception GALS, elles peuvent s'avérer très utiles, mais ce genre d'utilisation dépasse le cadre de la présente méthodologie.

Parmi les deux premières catégories de contraintes du tableau 2.1, la catégorie de placement est la seule qui semble pertinente. Elle permet d'imposer l'utilisation d'un composant précis et d'un chemin de routage en particulier. Cependant, le placement de ressources dans un FPGA exige que les éléments à contraindre fassent partie d'un groupe défini. Ainsi, les contraintes de regroupement deviennent tout aussi pertinentes pour notre objectif d'implémentation de composants GALS.

Le choix des contraintes de regroupement est relatif au nombre d'éléments à contraindre. Une conception de plus forte ampleur qu'une implémentation de composants GALS exigerait des contraintes pour gérer plus efficacement une quantité élevée d'éléments. Les contraintes BLKNM, HBLKNM et XBLKNM facilitent la gestion de grands groupes. Cependant, compte tenu de la portée réduite des composantes à créer, les contraintes de regroupement du type « set » (U_SET, H_SET et HU_SET) sont préférables. La contrainte de transcription de la description matérielle KEEP permet de préserver des éléments du processus d'optimisation en les sélectionnant un à un. Dans ce cas, il n'est pas nécessaire de former un groupe, mais il n'existe pas, non plus, de méthode directe pour contraindre le placement de ces éléments.

Les contraintes de placement définissent l'emplacement spécifique des diverses ressources d'un groupe. Pour les mêmes raisons que celles invoquées pour les contraintes de regroupement, les contraintes BEL, LOC et LOCATE offrent un niveau de granularité très fin, trop fin, en fait, pour nos besoins. L'inconvénient de ces contraintes est l'ampleur de la tâche à accomplir, tous les éléments de la conception doivent être contraints. Une alternative intéressante est la famille de contraintes RLOC qui permet un placement relatif. En spécifiant les éléments à contraindre par rapport aux éléments du groupe, il est possible de laisser l'outil de placement et routage faire le reste du travail relié à la logique synchrone, ce qu'il fait très bien et de façon automatique.

Sous-ensemble de contraintes

Le sous-ensemble de contraintes sélectionné pour guider les outils de positionnement de la logique à l'intérieur du FPGA et préserver les composants du processus d'optimisation est constitué des contraintes : KEEP; RLOC; H_SET; HU_SET; U_SET et LUT_MAP (Xilinx, 2005). Évidemment, en d'autres circonstances, certaines autres contraintes pourraient s'avérer utiles, mais pour l'instant, nous nous limiterons à ce sous-ensemble.

KEEP est une contrainte de synthèse et de translation de la description matérielle. Cette contrainte s'applique seulement aux signaux et elle s'apparente à une propriété NOMERGE. Elle préserve les signaux, considérés comme doublons logiques, dans la base de données de la conception.

RLOC est une contrainte de synthèse, de translation de la description matérielle et de placement. Cette contrainte permet de spécifier l'endroit relatif d'un élément et elle s'applique aux descriptions d'entités, aux composants et aux descriptions d'instances. Dans l'optique où le P&R de chaque élément est long et inefficace, la contrainte RLOC est utile pour forcer le placement relatif de quelques composants sans avoir à placer chacun d'eux comme pour la contrainte de LOC. Dans ce cas spécifique, c'est à l'outil de P&R qu'incombe la tâche de fournir le placement relatif entre les modules contraints. Il est à noter qu'une contrainte RLOC doit s'appliquer sur un élément membre d'un groupe préalablement défini.

SET est une contrainte de translation de la description matérielle qui permet de former des groupes. Elle s'applique sur un minimum de deux composants ou descriptions d'instances. Il existe trois contraintes de type SET pour grouper des éléments ensemble. Il y a H_SET, HU_SET et U_SET où « H » signifie « hiérarchique » et « U » signifie « utilisateur ». Chaque élément d'un groupe doit être affecté d'une contrainte RLOC relative aux autres éléments du groupe. Il est possible de créer des ensembles multiples, mais un élément doit être étiqueté dans un seul groupe. Pour une contrainte U_SET, le concepteur choisit les éléments et les regroupe sans se soucier de la hiérarchie sous-jacente. En revanche, H_SET définit implicitement un groupe en corrélation avec sa hiérarchie. De cette façon, les éléments et les sous-éléments affectés d'une contrainte RLOC seront liés. Dans le cas opposé, un élément dépourvu d'une contrainte RLOC ne sera pas associé au groupe malgré sa hiérarchie. Finalement, HU_SET est un groupe hiérarchique, similaire au groupe formé par la contrainte H_SET, où il est possible de personnaliser les liens entre les éléments à la manière de la contrainte U_SET selon les besoins du concepteur. Compte tenu de ces particularités, nous emploierons seulement des contraintes H_SET pour la construction des composants de base, puisqu'elles représentent le mécanisme fondamental hiérarchique des méthodologies récentes de conception.

LUT_MAP est un paramètre pour instancier directement une LUT à l'aide d'un langage HDL. Il s'applique aux descriptions d'entités seulement. Ce paramètre permet à un concepteur de définir sa propre fonction et de l'instancier dans une unique LUT, si les ressources de la LUT le permettent. Naturellement, il est aussi possible d'imposer un placement relatif de cette LUT en attachant les contraintes H_SET et RLOC à l'instance de cette entité.

Les contraintes en elles-mêmes ne servent à rien si elles ne sont pas transmises de façon adéquate aux différents outils. Une contrainte saisie à un endroit inopportun ne généra pas nécessairement d'erreurs, mais ne sera pas prise en compte et l'implémentation sera ratée, c'est pourquoi la saisie des contraintes est très importante.

Saisie des contraintes

Il y a plusieurs manières de saisir les contraintes (Xilinx, 2005). La plus connue est le fichier de contrainte d'utilisateur (UCF). Cependant, il est également possible de saisir des contraintes lors de la capture schématique ou directement dans le code source. Il est donc nécessaire de choisir un niveau de capture permettant de spécifier toutes les contraintes de notre sous-ensemble de contraintes. La liste suivante illustre le niveau de capture ainsi que les contraintes qu'il supporte :

- capture schématique, → KEEP, H_SET et RLOC;
- code VHDL, → KEEP, LUT_MAP, H_SET et RLOC;
- code Verilog, → KEEP, LUT_MAP, H_SET et RLOC;
- code ABEL, → KEEP;
- fichiers UCF/NCF/XCF, → KEEP, LUT_MAP, H_SET et RLOC;
- éditeur de planification de surface, → RLOC.

Pour notre sous-ensemble de contraintes, seuls les fichiers de contraintes UCF/NCF/XCF et les codes HDL sont viables pour la saisie des contraintes. Une approche intuitive est de spécifier les contraintes sous forme d'attributs directement dans le code VHDL et d'utiliser le fichier UCF pour les caractéristiques relatives aux ports d'entrées et de sorties. Pour la saisie des contraintes dans le code VHDL, les contraintes liées à l'entité doivent être saisies dans la description d'entité alors qu'elle se fait entre les mots-clés « ARCHITECTURE » et « BEGIN » dans le corps d'architecture pour toutes les autres contraintes.

Fort de ces informations, il est maintenant plus facile d'implémenter des composants GALS qui, à leur tour, permettront les communications GALS.

2.3 Composants de base

Le composant asynchrone par excellence est l'élément C de Muller (Sutherland, 1989). Ce dernier a prouvé son utilité pour la mise en œuvre de communications asynchrones où il a été

utilisé dans une pléiade d'applications : du détecteur d'achèvement de tâches à l'arbitrage. Dans tous les cas, sa fonction de rendez-vous le rend indispensable. Les autres éléments tels l'oscillateur local, la ligne à retard, le détecteur de fronts, l'arbitre, et autres sont facultatifs à l'implémentation de systèmes GALS. Cependant, ils peuvent être utiles pour certaines applications. Certes, l'absence de ces composants restreint le champ d'application, mais il faut les voir comme des éléments utiles aux systèmes GALS et non comme des éléments indispensables.

Malgré ceci et afin d'illustrer clairement la méthode d'implémentation des composants de base, nous ferons l'implémentation d'une ligne à retard, un premier exemple mettant en lumière la méthodologie utilisée pour l'implémentation de l'élément C de Muller (un classique en logique GALS). L'arbitre sera également présenté puisqu'il circonscrit la métastabilité et autorise l'implémentation de générateurs d'horloge, de synchroniseurs de données et d'autres modules encore plus complexes (Moore et al., 2002; Moore et al., 2000).

2.3.1 Implémentation d'une ligne à retard

La ligne à retard est particulièrement utile pour ajuster la période de l'horloge dans un générateur d'horloge (Moore et al., 2000). Il existe deux façons de créer une ligne à retard dans une conception sur FPGA. Elle peut être jumelée à d'autres éléments à contraindre ou être indépendante. Dans le premier cas, la ligne à retard est implémentée à l'aide d'une seule LUT à une entrée (LUT1). Dans le second cas, l'utilisation de la contrainte KEEP permet un résultat similaire, mais où le placement manuel n'est pas permis. Lorsque le placement spécifique est requis, il faut créer un groupe contenant au moins deux éléments, deux LUT dans cet exemple. Cette solution se compare désavantageusement à la ligne à retard à une seule LUT à cause du dédoublement de logique et d'une granularité deux fois plus grande pour le retard résultant. Bref, la meilleure approche est d'ajouter une seule LUT1 à un groupe existant à contraindre.

La méthode la plus simple d'implémentation du composant LUT1 se fait par l'entremise de la bibliothèque UNISIM. Toutefois, l'utilisation du composant LUT1 génère un

avertissement associé aux composants implémentés explicitement à l'aide d'une LUT. Cet avertissement indique que le composant LUT1 contient le paramètre LUT_MAP. Évidemment, cet avertissement peut être négligé, mais il suffit de déclarer ce composant comme une boîte noire (« black_box ») pour éliminer les avertissements à cet égard.

Le placement précis des ressources du FPGA pour l'implémentation d'une ligne à retard basée sur des ressources LUT1 se fait à l'aide des contraintes H_SET et RLOC. Il est à noter que si le placement n'est pas important, la contrainte KEEP peut être utilisée à la place. La figure 2.5 illustre la hiérarchisation des contraintes à l'égard de l'ajout d'une ligne à retard (« LigneRetard ») dans un projet (« Projet »). Les contraintes inscrites en caractères gras illustrent des contraintes héritées, et ce, de deux façons. L'entité « Projet » hérite, dans un premier temps, de sa contrainte par l'outil de P&R qui fixe la position exacte de l'instanciation. Le sous-module « LigneRetard » hérite, dans un second temps, du placement relatif de ces composants. Les autres contraintes proviennent du fichier source.

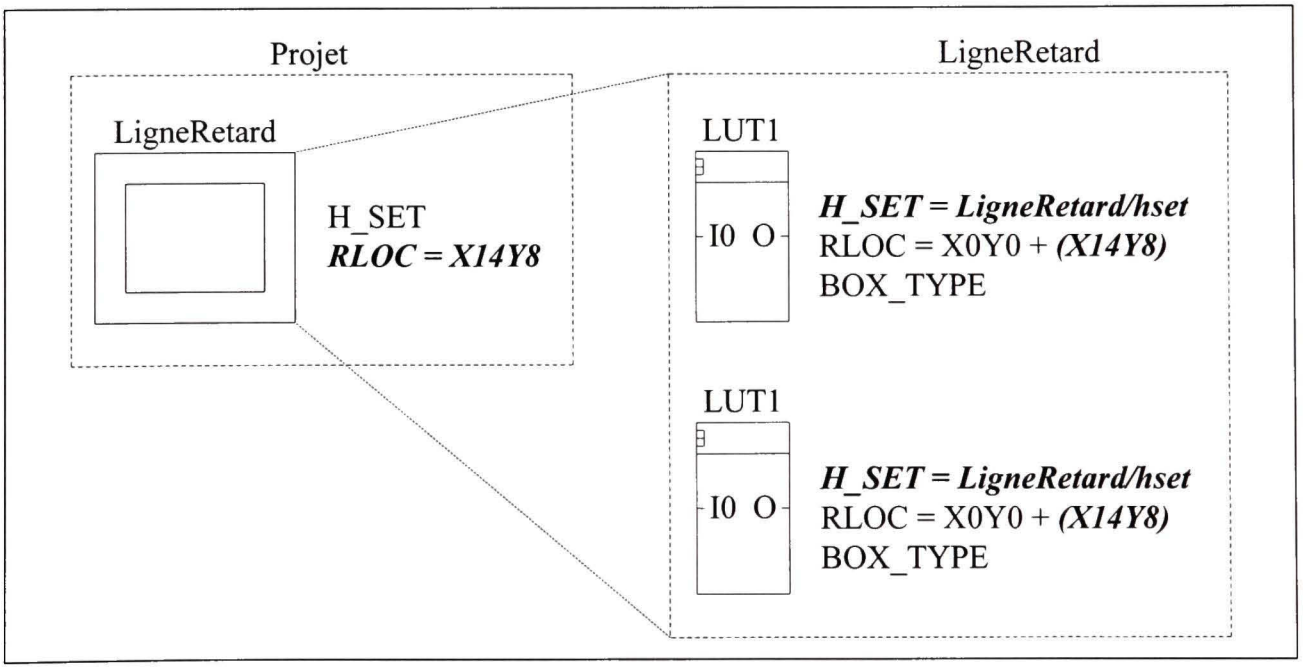


Figure 2.5 Contraintes d'une ligne à retard.

La figure 2.6 illustre le code VHDL d'une ligne à retard formée de deux LUT1 nommées « retard », conformément aux contraintes de la figure 2.5. Il s'agit du pire cas

d'implémentation, mais également du cas le plus complet, c'est pourquoi il est utilisé à titre d'exemple. Il est à noter que le vecteur d'initialisation « 10 » pour le composant LUT à deux entrées (LUT1) implémente un tampon. Lorsque le signal d'entrée est à '0', la composante génère le signal de sortie '0'. De la même façon, elle génère le signal de sortie '1' pour un signal d'entrée à '1'.

```
LIBRARY IEEE, UNISIM;
USE IEEE.STD_LOGIC_1164.ALL;
USE UNISIM.VComponents.ALL;
ENTITY LigneRetard IS
      ( AVANCE : IN STD_LOGIC; RETARD : OUT STD_LOGIC );
      ATTRIBUTE lut_map : STRING;
      ATTRIBUTE lut_map OF LigneRetard : ENTITY IS "yes";
END LigneRetard;
ARCHITECTURE Implementation OF LigneRetard IS
SIGNAL intermediaire_s : STD_LOGIC;
COMPONENT LUT1
      GENERIC( INIT : STD_LOGIC_VECTOR(1 DOWNTO 0) := "10" );
      PORT( I0 : IN STD_LOGIC; O : OUT STD_LOGIC );
END COMPONENT;
ATTRIBUTE h_set : STRING;
ATTRIBUTE rloc : STRING;
ATTRIBUTE box_type : STRING;
ATTRIBUTE h_set OF U0 : LABEL IS "retard";
ATTRIBUTE h_set OF U1 : LABEL IS "retard";
ATTRIBUTE rloc OF U0 : LABEL IS "X0Y0";
ATTRIBUTE rloc OF U1 : LABEL IS "X0Y0";
ATTRIBUTE box_type OF LUT1 : COMPONENT IS "black_box";
BEGIN
      U0 : LUT1 GENERIC MAP( INIT => "10" )
         PORT MAP( I0 => AVANCE, O => intermediaire_s );
      U1 : LUT1 GENERIC MAP( INIT => "10" )
         PORT MAP( I0 => intermediaire_s, O => RETARD );
END Implementation;
```

Figure 2.6 Code VHDL d'une ligne à retard incluant les contraintes (attributs).

La translation générique de ce code VHDL en portes logiques est illustrée à la figure 2.7a. De plus, la figure 2.7b illustre la simulation logique après P&R de la ligne à retard émanant de la

description matérielle VHDL de la figure 2.6. Il est à noter que cette simulation logique n'a pas d'échelle de temps pour illustrer le concept de façon général, mais le retard est illustré comme étant $2\alpha + \beta$ où α et β représentent respectivement le retard dans les portes logiques (LUT1) et dans le fils (intermediaire_s).

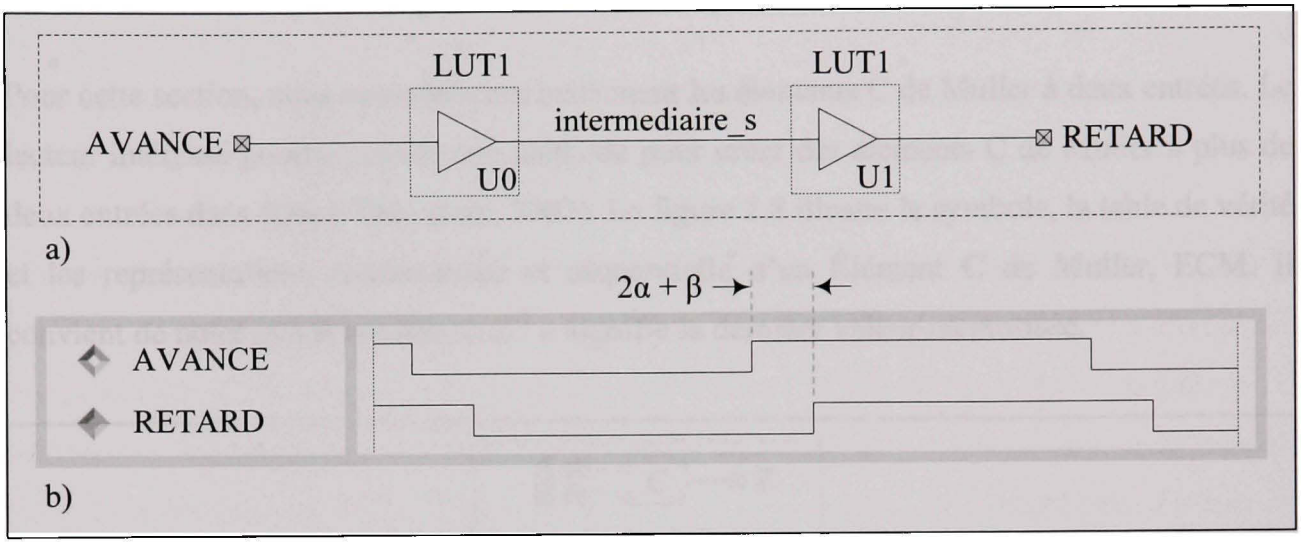


Figure 2.7 Ligne à retard.
a) Implémentation de la description matérielle VHDL.
b) Simulation logique après P&R.

La simulation logique incluant les retards de P&R, PP&RS, (**P**ost **P**lace & **R**oute **S**imulation) de la figure 2.7b confirme que la ligne à retard n'a pas de limitations fonctionnelles lorsqu'elle est implémentée sur le FPGA XC2V40-6-fg256. Cependant, la disponibilité des ressources du FPGA et la variabilité du procédé de fabrication influent sur la granularité du retard implémenté. Ces limites définissent les performances du circuit.

2.3.2 Implémentation d'un élément C de Muller

L'élément C de Muller est intéressant dans un contexte de protocole de communication pour établir la priorité des événements asynchrones (Sparsø et Furber, 2001). La sortie de l'élément C de Muller prend la valeur de ses entrées quand toutes les entrées ont le même niveau logique. Par conséquent, la sortie est placée à un niveau haut (niveau bas) quand toutes les entrées sont forcées à un niveau haut (niveau bas). Pour toutes les autres

conditions, la sortie demeure inchangée en mode de mémorisation. Dans ce mode, l'élément C de Muller se comporte comme un verrou transparent. Par conséquent, les protocoles peuvent tirer profit de la propriété de « rendez-vous » de l'élément C de Muller pour mettre en application des échanges de données « point à point » à l'aide de signaux de poignée de main (Sutherland, 1989).

Pour cette section, nous considérerons seulement les éléments C de Muller à deux entrées. Le lecteur intéressé pourra trouver une méthode pour créer des éléments C de Muller à plus de deux entrées dans (Quoc Thai et al., 2002). La figure 2.8 illustre le symbole, la table de vérité et les représentations combinatoire et séquentielle d'un **É**lément **C** de **M**uller, ECM. Il convient de noter que la notation « Z^{-1} » signifie la dernière valeur mémorisée.

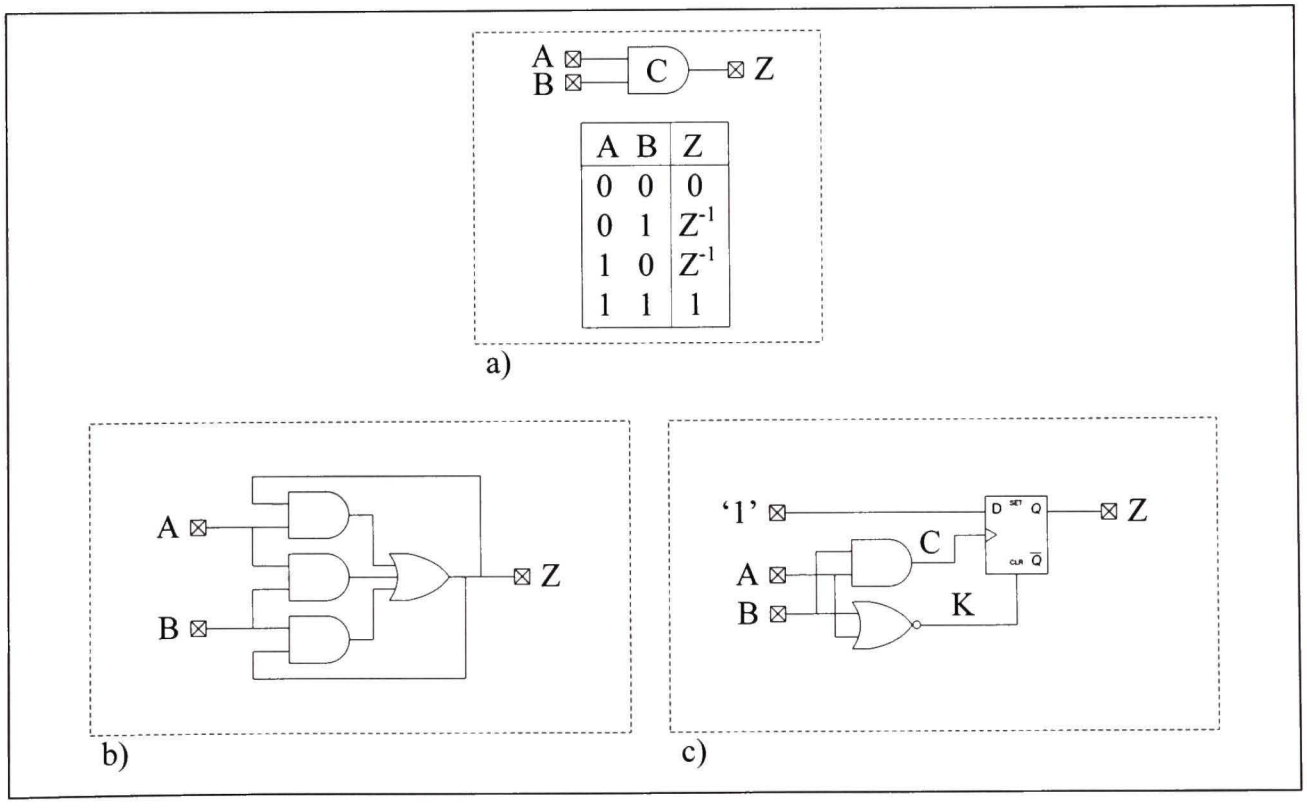


Figure 2.8 Élément C de Muller.
a) Symbole et table de vérité.
b) Représentation combinatoire. c) Représentation séquentielle.

Malheureusement, les FPGA synchrones n'incorporent pas d'ECM à titre de ressources internes. Pour résoudre ce problème, une tendance consiste à implémenter ces éléments

(Quoc Thai et al., 2002; Xin, Tapani et Jari, 2006) en accord avec la méthode introduite à la section précédente. La figure 2.9 illustre deux exemples d'implémentation de l'ECM pour un Virtex-II où les lignes grises représentent des connexions inutilisées. La figure 2.9a met en application la représentation combinatoire de l'ECM (figure 2.8b), un circuit complet incluant les retards de routage. La seconde implémentation, illustrée partiellement, nécessite un registre et deux LUT. Pour cette implémentation, une conception basée sur un registre de type bascule « D » est préférable à celle basée sur une bascule « RS » parce qu'elle diminue les risques de métastabilité par son entrée de données connectée au rail d'alimentation.

Compte tenu du fait que ces solutions possèdent des signaux de rétroaction (externe pour la LUT et interne pour le registre), l'ECM doit être activé ou désactivé pour un intervalle de temps minimum ($Tecm_{MIN}$) de manière à ce que la sortie se stabilise par l'intermédiaire de la rétroaction. En dessous de cette limite d'opération, le circuit ne fonctionne pas correctement. Pour l'implémentation LUT, $Tecm_{MIN}$ correspond à la somme des retards du chemin de rétroaction (198ps), du retard de génération du signal à la sortie de la LUT (236ps) et du retard de l'amplificateur de sortie de la LUT (111ps). $Tecm_{MIN} = 198 + 236 + 111 = 545ps$. Les autres valeurs de retards illustrés à la figure 2.9 ne sont pas utiles pour caractériser l'ECM, elles illustrent un P&R. Aucun effort n'a été fait pour optimiser ce P&R puisqu'il s'agit ici de démontrer les limites de l'ECM en fonction de son paramètre $Tecm_{MIN}$. Toutefois, pour une utilisation courante, il est pertinent de minimiser $Tecm_{MIN}$. Pour l'implémentation basée sur un registre $Tecm_{MIN} = 610ps$ ce qui correspond à la largeur minimum d'impulsion pour un registre. De plus, au chapitre des performances, le P&R et les variations de fabrication influent, de façon similaire à la granularité de la ligne à retard, sur la précision du paramètre $Tecm_{MIN}$.

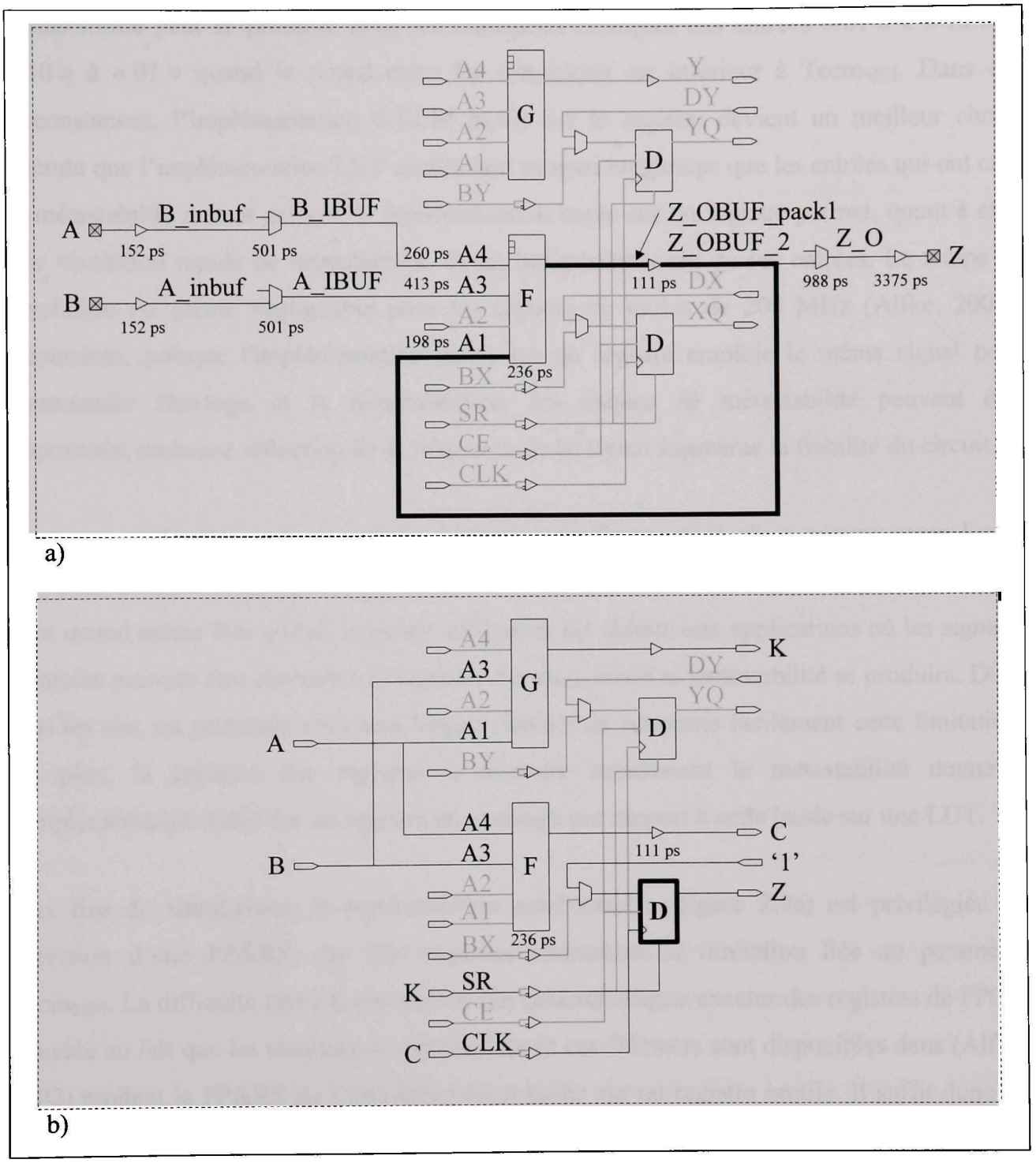


Figure 2.9 Élément C de Muller implémenté dans un Virtex-II.
a) Implémentation dans une LUT. b) Implémentation basée sur un registre.

En présence d'un environnement asynchrone fondamental, il n'y a pas de risque de métastabilité. L'implémentation LUT s'avère, dans ce cas, le meilleur choix parce qu'elle exige moins de ressources et est plus rapide. Cependant, dans un environnement GALS, la

métastabilité peut se produire pour les transitions multiples des entrées « 01 » à « 10 » et « 10 » à « 01 » quand le retard entre les transitions est inférieur à $Tecm_{MIN}$. Dans ces circonstances, l'implémentation d'ECM basée sur le registre devient un meilleur choix, attendu que l'implémentation LUT oscille tant et aussi longtemps que les entrées qui ont créé la métastabilité restent actives. L'implémentation basée sur un registre permet, quant à elle, une résolution rapide de métastabilité, et ce, indépendamment de ses entrées. Le temps de résolution est même négligeable pour les signaux de moins de 200 MHz (Alfke, 2002). Cependant, puisque l'implémentation basée sur un registre emploie le même signal pour commander l'horloge et la réinitialisation, les risques de métastabilité peuvent être importants, mais une réduction de la fréquence de ce signal augmente la fiabilité du circuit.

Pour une application qui respecte la limitation de $Tecm_{MIN}$, il n'y a aucune autre limite d'opération. Cependant, dans le cas où le respect de $Tecm_{MIN}$ ne peut pas être garanti, l'ECM peut quand même être utilisé, mais son utilisation est réduite aux applications où les signaux d'entrées peuvent être contraints à respecter $Tecm_{MIN}$ sinon la métastabilité se produira. Dans tous les cas, un protocole avec une longue rétroaction surmonte facilement cette limitation. De plus, la capacité des registres à résoudre rapidement la métastabilité donne à l'implémentation basée sur un registre un avantage par rapport à celle basée sur une LUT.

Aux fins de simulations, la représentation combinatoire (figure 2.9a) est privilégiée en prévision d'une PP&RS, car elle explicite clairement la limitation liée au paramètre $Tecm_{MIN}$. La difficulté liée à la simulation des caractéristiques exactes des registres de FPGA jumelée au fait que les résultats de simulations de ces éléments sont disponibles dans (Alfke, 2002) rendent la PP&RS de l'implémentation basée sur un registre inutile. Il suffit donc de simuler l'implémentation basée sur une LUT. Dans ces circonstances, supposons, à titre de stimuli de simulation, que les deux entrées de l'ECM sont stables et de polarité opposée (mode de mémorisation). La sortie est également stable et à un niveau bas (Z = '0'). Une impulsion sur l'entrée qui était à un niveau bas, active la sortie (cas C, D, F, H et J, la figure 2.10). Cependant, si la longueur de l'impulsion d'entrée est inférieure à $Tecm_{MIN}$, qui est de 545ps dans cet exemple, cela signifie que le signal de sortie n'a pas le temps de se stabiliser. En conséquence de quoi, la sortie oscille, et ce, aussi longtemps que l'ECM reste en mode de

mémorisation (cas E, G, I et K, figure 2.10). Finalement, les deux premiers cas de simulation représentent des zones de mémorisation sans changement à la sortie pour l'ECM (cas A et B, figure 2.10). La figure 2.10 illustre ces cas de PP&RS.

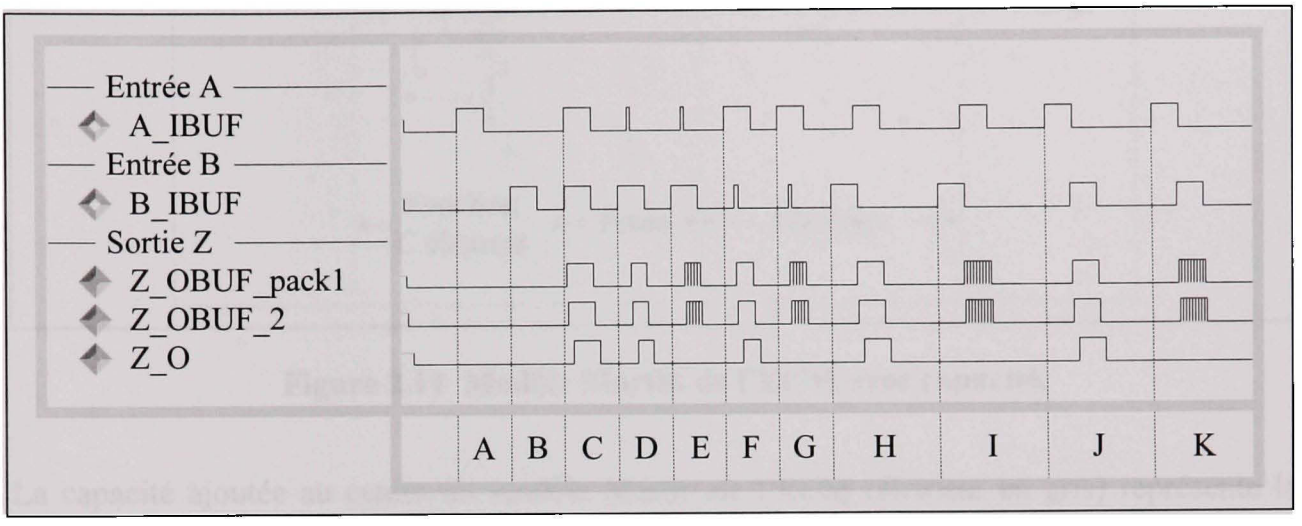


Figure 2.10 PP&RS de l'ECM dans un Virtex-II.

La figure 2.10 illustre également la limitation de la bande passante causée par la capacité parasite sur le plot de sortie (des changements rapides de sortie sont filtrés sur Z_O et Z (cas E, G, I et K, figure 2.10) par les éléments de sortie (figure 2.9)). Les résultats de la PP&RS laissent croire que la métastabilité est masquée, mais, en fait, les conditions de métastabilité sont seulement déplacées. En d'autres mots, les conditions engendrant les oscillations ont été altérées, ce qui se traduit par un changement des conditions créant la métastabilité, sans l'éliminer. Conséquemment, la sortie Z oscillera lors d'une implémentation.

Dans le but de confirmer l'hypothèse résultante de la PP&RS, une simulation SPICE s'impose. En vue de cette simulation, le modèle Martin de l'ECM est utilisé (Shams, Ebergen et Elmasry, 1998). Ce modèle particulier se distingue par sa simplicité et sa lenteur d'opération due à une cellule de mémoire réalisée à l'aide de deux inverseurs connectés en boucle. Cette technique est simple, mais elle rend le fonctionnement dépendant de la taille des transistors utilisés. La figure 2.11 illustre le modèle Martin de l'ECM.

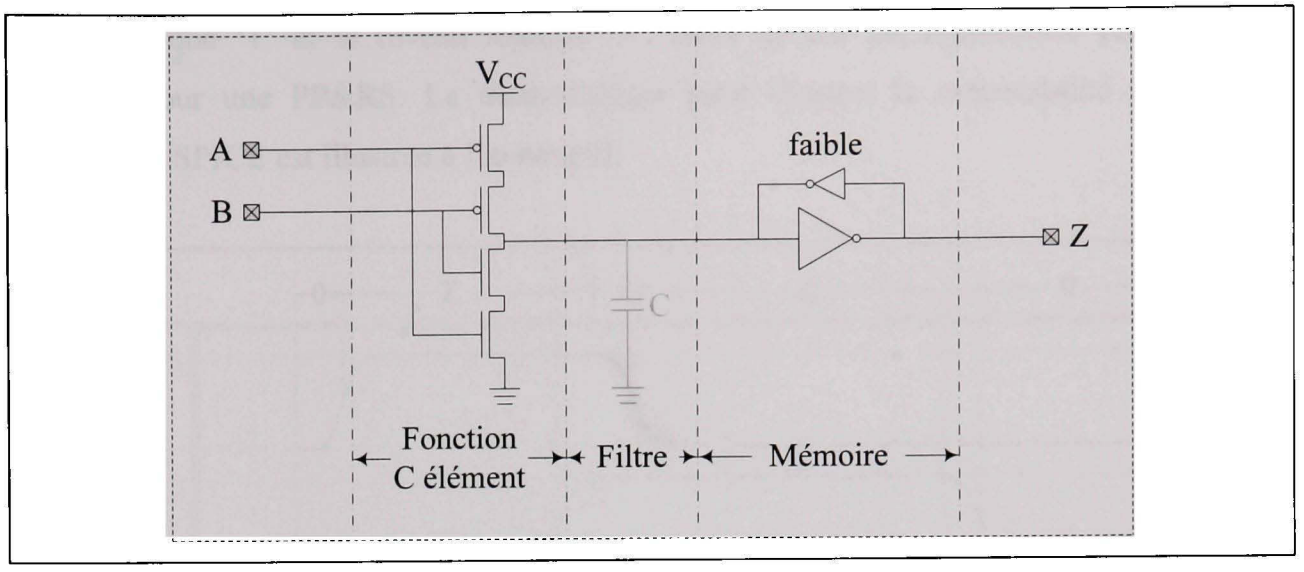


Figure 2.11 Modèle Martin de l'ECM avec capacité.

La capacité ajoutée au centre du modèle Martin de l'ECM (élément en gris) représente la capacité parasite du circuit. Cette dernière permettra de vérifier l'hypothèse selon laquelle le réglage d'une capacité de filtrage déplace le problème de métastabilité sans jamais le régler. La figure 2.12 illustre la simulation SPICE du modèle Martin de l'ECM avec une capacité de filtrage nulle. Pour toutes les simulations SPICE, des tampons d'isolation composés de deux inverseurs en série ont été utilisés pour transformer les entrées parfaites du simulateur SPICE en des signaux représentatifs de la réalité. Malgré ce tampon, les transitions d'entrées (idéales) ont également été limitées à 1ps pour être représentatives de la technologie utilisée pour les simulations soit une technologie 0.18µm avec une bibliothèque SPICE BSIM3.2. Le choix de ce taux de changement pour les transitions des signaux est fait de façon à s'assurer de dépasser les spécifications de la technologie sous-jacente. De plus, toutes les simulations SPICE incluent l'extraction des capacités parasites pour obtenir un comportement plus représentatif de la réalité. Toutes ces conditions de simulation se retrouvent sous forme d'aide-mémoire à l'annexe I.

La métastabilité est créée lorsqu'une transition est amorcée (zone identifiée à '1') , mais qu'elle n'a pas eu le temps d'être complétée avant son retour en phase de mémorisation (zone identifiée à 'Z'). Il convient de noter que la métastabilité de la simulation SPICE est représentée par une transition impliquant un état intermédiaire, un niveau se situant entre le

niveau logique '1' et le niveau logique '0', alors qu'elle est représentée par un signal oscillant pour une PP&RS. La méthodologie pour illustrer la métastabilité à l'aide de simulations SPICE est illustrée à l'annexe II.

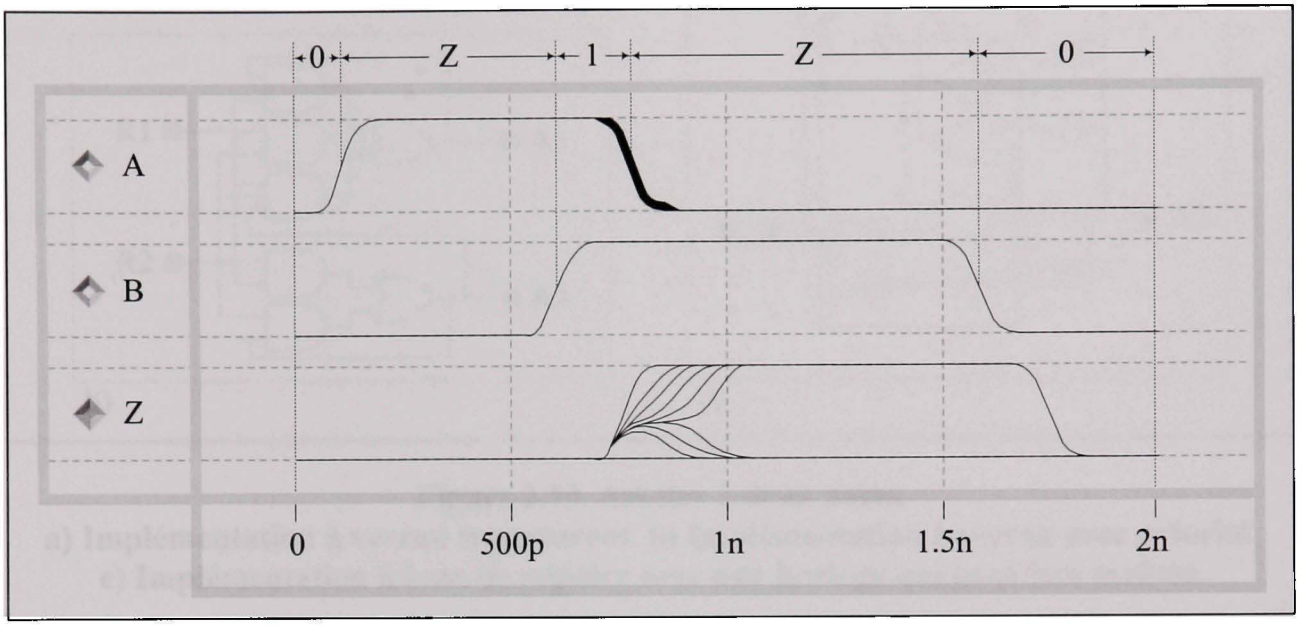


Figure 2.12 Simulation SPICE du modèle Martin de l'ECM.

La simulation SPICE à la figure 2.12 confirme qu'un filtre capacitif n'est pas viable pour masquer la métastabilité. D'autres simulations SPICE, avec des capacités de 20fF et 50fF ont été faites (non montrées) et donnent des formes d'ondes similaires, mais avec une latence et un $Tecm_{MIN}$ plus grand. En fin de compte, peu importe la valeur de la capacité de filtrage, la métastabilité ne peut pas être éliminée de cette façon.

2.3.3 Implémentation d'un arbitre

La fonction de l'arbitre est d'accorder un droit exclusif à une seule des demandes externes sur la base du premier arrivé, premier servi. Dans un contexte de logique GALS ou asynchrone, loin du mode asynchrone fondamental, l'arbitre est sujet à la métastabilité.

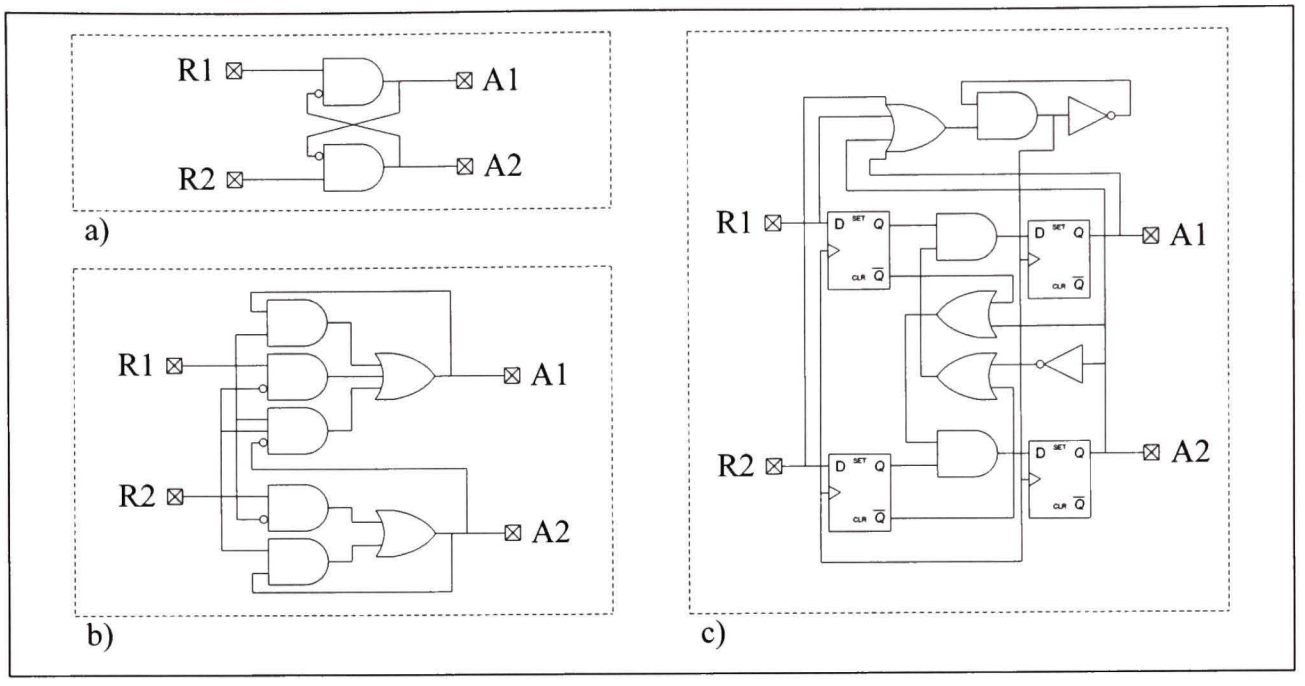


Figure 2.13 Arbitre à deux ports.
a) Implémentation à verrou transparent. b) Implémentation à verrou avec priorité.
c) Implémentation à base de registre avec une horloge qui peut être arrêtée.

La figure 2.13 montre trois formes populaires d'implémentation d'un arbitre à deux ports où chaque port d'entrée est échantillonné par l'activation de son signal de requête (R) et où le port de reconnaissance (A) correspondant est activé en fonction des requêtes (Najibi et al., 2005; Taylor et al., 2000). Pour une requête simple (1 seul port d'entrée activé), le signal de reconnaissance associé est activé pour permettre le partage d'une ressource précise. Un problème peut survenir pour des requêtes simultanées ou rapprochées (transitions rapides de « 00 » à « 11 »). Dans ces circonstances, des oscillations, résultat de la métastabilité, peuvent se produire sur les sorties.

La figure 2.13a et la figure 2.13b sont des implémentations de verrous transparents basés sur deux LUT avec deux et trois chemins de rétroaction respectivement (Najibi et al., 2005). Ces circuits présentent une limitation similaire à celle du $Tecm_{MIN}$ de l'ECM, que l'on appelle $Tarb_{MIN}$ pour l'arbitre. Dans le même esprit, les PP&RS (non montrées) ont confirmé que l'arbitre est fonctionnel quand cette limitation est respectée et que les demandes rapides et rapprochées (par rapport à $Tarb_{MIN}$) causent des oscillations à la sortie de l'arbitre. Ces

oscillations demeurent tant et aussi longtemps que les requêtes demeurent dans leur état initial (la durée des oscillations peut être très longue, voire tendre vers l'infini).

La solution traditionnelle pour ces arbitres est d'utiliser un filtre analogique pour résoudre la métastabilité. À titre d'exemple, l'arbitre de Seitz (Seitz, 1980) permet d'attendre la résolution de métastabilité avant de produire des sorties franches. Cependant, dans un FPGA synchrone, un tel filtre analogique n'est pas disponible et ne peut pas être implémenté avec les ressources internes du FPGA. Dans ces circonstances, la seule solution est d'exploiter, à nouveau, la faculté de résolution rapide de la métastabilité des registres du FPGA.

L'arbitre basé sur quatre registres (Taylor et al., 2000) est illustré dans la figure 2.13c. Cet arbitre est plus tolérant à la métastabilité puisque les registres résolvent la métastabilité rapidement et l'horloge qui peut s'arrêter permet de masquer certaines sources de métastabilité. Bien que l'implémentation à base de registre exige plus de ressources (quatre LUT et quatre registres) et qu'elle soit plus lente qu'un verrou transparent, elle produit beaucoup moins d'erreurs fonctionnelles liées à la résolution de métastabilité.

2.4 Générateur d'horloge d'un port de communication GALS unidirectionnel

En regard à l'objectif global de permettre le support GALS minimal pour les FPGA synchrones, cette section présente une solution courante intégrant les composants de base pour établir des communications GALS dans un FPGA synchrone existant. Ce genre d'application convient parfaitement aux conceptions principalement synchrones avec peu de liens asynchrones sur de longs liens de communication.

Une application traditionnelle illustrant la fonctionnalité des composants proposés est une variation du module du générateur d'horloge (Seitz, 1980) d'un port de communication GALS (Moore et al., 2002). Comme la littérature le rapporte, le générateur d'horloge, dont la génération de l'horloge peut être arrêtée, constitue la source principale de métastabilité pour un port de communication GALS. La figure 2.14 illustre un générateur d'horloge pour un port de communication GALS unidirectionnel dont la génération peut être arrêtée.

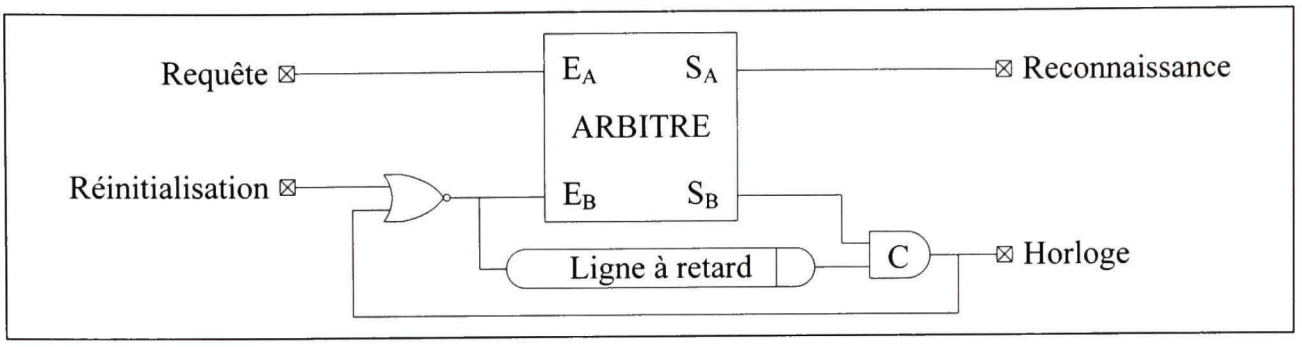


Figure 2.14 Générateur d'horloge pour communication GALS unidirectionnel.

La génération du signal d'horloge peut être suspendue (niveau bas) de deux façons : par l'activation (niveau haut) de signal de remise à zéro « Réinitialisation » ou par une « Requête » (niveau haut sur le port E_A de l'arbitre). Dans le premier cas, aucun autre front montant sur le signal d'horloge n'est généré. Dans le second cas, la génération est arrêtée et un signal d'acquittement est activé (un niveau haut sur le port « S_A » de l'arbitre) de façon à permettre l'échange de données sur le port de communication GALS. Le lecteur intéressé trouvera une description complémentaire sur les générateurs d'horloge dans (Seitz, 1980).

Pour l'implémentation d'un générateur d'horloge qui peut être arrêté, la ligne à retard peut être implémentée à l'aide d'une seule LUT. Cependant, cet élément doit être ajusté (nombre de LUT) en fonction de la fréquence désirée en considérant les limitations d'opération des autres éléments du design. Une ligne à retard à une seule LUT atteint une fréquence d'horloge maximale et respecte habituellement la limitation du $Tecm_{MIN}$. Cette hypothèse doit nécessairement être vérifiée par PP&RS pour une implémentation donnée. L'ECM est implémenté à l'aide d'une seule LUT également parce qu'il est utilisé dans le mode asynchrone fondamental (la durée du retard incluant le routage est plus grande que $Tecm_{MIN}$). Finalement, l'arbitre est implémenté à l'aide de registres pour permettre la résolution rapide de la métastabilité. Le temps de résolution est assez rapide qu'il est considéré comme négligeable pour des signaux d'horloge en dessous des 200 MHz. Les PP&RS ont confirmé, encore une fois, la fonctionnalité et les limitations de l'arbitre.

Le tableau 2.2 résume les paramètres d'implémentation des circuits présentés à la figure 2.7, à la figure 2.9 et à la figure 2.13. Pour chaque circuit, les ressources nécessaires sont

présentées (nombre de LUT et de registres) tout en rappelant les limitations fonctionnelles et celles relatives aux performances de chacun d'eux.

Tableau 2.2 Résumé des paramètres d'implémentation

Circuit	Ressources		Limitations	
	LUT	Registres	Fonctionnelles	Performances
figure 2.7a	≥ 1	0	Aucune limite	Granularité du retard
figure 2.9a	1	0	Impulsion $\geq Tecm_{MIN}$	Précision de $Tecm_{MIN}$
figure 2.9b	2	1	Impulsion minimum	Performance des registres
figure 2.13a	2	0	Impulsion $\geq Tarb_{MIN}$	Précision de $Tarb_{MIN}$
figure 2.13b	2	0	Impulsion $\geq Tarb_{MIN}$	Précision de $Tarb_{MIN}$
figure 2.13c	4	4	Zone défendue	Performance des registres

2.5 Conclusion et direction du prochain chapitre

La méthodologie d'implémentation de composants proposée fournit les composants dédiés pour permettre l'appui minimal aux conceptions GALS dans les FPGA synchrones actuels.

Les résultats montrent que peu de composants sont nécessaires pour qu'un FPGA augmente sa versatilité par des communications asynchrones. La technologie courante de FPGA peut soutenir les échanges d'informations asynchrones. Un port de communication GALS a été utilisé, comme exemple, pour confirmer que l'ECM permet des transferts GALS dans un FPGA synchrone commercial. Naturellement, une telle réalisation possède des limites, mais les limites fonctionnelles peuvent toutes être contournées de façon à créer un environnement GALS viable dans un FPGA synchrone.

Même si l'élimination de la métastabilité représente l'objectif idéal, il est impossible de l'atteindre. Par conséquent, nos travaux futurs sont orientés, d'une part, vers une nouvelle architecture de FPGA permettant un masquage plus efficace de la métastabilité et d'autre part, vers une version ASIC d'un circuit de masquage d'aléas temporels intégré comme ressources locales aux futurs FPGA permettant de masquer les cas problématiques avant qu'ils ne se produisent.

CHAPITRE 3

Architecture nativement GALS pour FPGA

Ce chapitre est une traduction libre de l'article « From synchronous to GALS : A new architecture for FPGAs » publié dans la revue « microelectronics journal » (Gagné, Belzile et Thibeault, 2009e).

3.1 Introduction

La complexité croissante des circuits intégrés, IC, (**I**ntegrated **C**ircuit) dont nous avons été témoins au cours des dernières décennies, a forcé les concepteurs à adopter des approches simples et automatisées pour faciliter la conception et la vérification d'IC. Parmi ces approches, les méthodologies de conception synchrones, avec leur ensemble complexe d'outils, ont largement dominé puisqu'elles permettent des simplifications temporelles qui justifient l'enthousiasme envers ce type de conceptions (Friedman, 2001). Avec les capacités d'intégration (densité, surface, etc.) et la fréquence du signal d'horloge atteignable pour les technologies CMOS, les méthodologies de conception synchrones sont confrontées à la tâche toujours plus complexe de distribution du signal d'horloge dans tout le circuit (Friedman, 2001; Matzke, 1997). Cette tâche est d'autant plus difficile lorsqu'on considère les problèmes de bruit tout en gardant la puissance à un niveau acceptable (Smith, 2007). Par conséquent, les préoccupations concernant la convenance de ces méthodologies pour les futurs dispositifs augmentent (Matzke, 1997; Smith, 2007). L'ITRS a récemment identifié les systèmes GALS (Chapiro, 1985) en tant qu'alternative viable à ces problèmes. Dans la mesure où les outils de conception asynchrone ne sont pas aussi matures que ceux des méthodologies de conception synchrones (Hauck, 1995), une méthodologie GALS semble être un bon compromis permettant l'utilisation locale des outils et techniques synchrones sur les îlots synchrones tout en utilisant une approche asynchrone pour les communications globales entre ces îlots.

Les réalisations custom et semi-custom, y compris les circuits intégrés dédiés ASIC, ont jusqu'ici été les cibles préférées pour les conceptions GALS et les conceptions asynchrones.

Cependant, les FPGA émergent en tant que moyen alternatif pour le prototypage (Najibi et al., 2005). Ils augmentent en popularité pour les conceptions synchrones alors que le nombre de conceptions ASIC décroit (LaPerdus, June 11, 2007). Gartner/Dataquest a récemment estimé qu'il y avait 25 fois plus de nouvelles conceptions basées sur des FPGA que de conceptions basées sur des ASIC (Chen et Wong, 2008). Dans beaucoup de cas, les FPGA ont un accès plus rapide aux nouvelles technologies comparativement aux ASIC (Morris, December 6, 2005). C'était le cas pour la technologie 90nm d'IBM où Xilinx était leur premier client (Global semiconductor forum, 2003). C'est cet accès rapide, aux dernières technologies, qui a fait des FPGA, les premiers IC exposés aux défis des technologies inférieurs au micron, les mêmes que ceux adressés par les méthodologies GALS.

Plusieurs éléments sont requis pour faciliter la transition des méthodologies de conception synchrone à celle des méthodologies GALS. Dans ce chapitre, nous présentons une nouvelle architecture de FPGA transparente aux conceptions synchrones, exigeant un niveau de changements minimum et masquant la métastabilité pour les communications asynchrones. En effet, nous proposons d'ajouter des composants asynchrones à une architecture classique de FPGA synchrone pour soutenir les conceptions GALS. Les modifications proposées sont mineures et peuvent facilement être incorporées aux outils de conception automatisée, EDA, (**E**lectronic **D**esign **A**utomation) allégeant, du même coup, le fardeau des concepteurs.

Ce chapitre est divisé en neuf sections. La section 3.2 donne une vue d'ensemble de l'état de l'art dans le domaine des FPGA GALS. La section 3.3 présente les conditions pour soutenir des liaisons asynchrones dans des architectures synchrones. L'architecture proposée est présentée, analysée et simulée dans les sections 3.4, 3.5 et 3.6 respectivement. La section 3.7 présente l'application complète d'un générateur d'horloge pouvant être arrêté. Les défis d'implémentation sont présentés à la section 3.8. Finalement, la section 3.9 fournit des conclusions et les orientations pour les travaux futurs.

3.2 État de l'art

Dans cette section, nous passerons en revue la situation actuelle en portant une attention particulière à la tâche de masquage de la métastabilité lors de transferts asynchrones. Cette revue sera faite dans un contexte de compatibilité et de changements minimums pour les architectures existantes de FPGA.

Il existe deux tendances importantes dans le domaine des FPGA GALS. La première approche, celle du chapitre précédent, consiste à employer des méthodes asynchrones et à les appliquer directement aux FPGA existants. Il faut pour cela, a) émuler les éléments asynchrones, comme l'ECM, (Quoc Thai et al., 2002) ou l'arbitre (Najibi et al., 2005) dans des dispositifs synchrones, ou b) utiliser la désynchronisation comme la logique à retard de phase dans un contexte synchrone (Traver, Reese et Thornton, 2001). Dans ce dernier cas, des ajustements précis de phase des signaux sont nécessaires pour assurer des transferts corrects entre les sections mésochrones. Les solutions de cette première tendance allouent des transferts asynchrones, mais ces transferts sont sujets à des contraintes de synchronisation très serrées. Évidemment, satisfaire ces contraintes présente des défis sérieux pour les outils d'EDA et pour la communauté de concepteurs. En ce sens, Xilinx documente la fréquence maximale d'opération de ses bascules de mémoire, pour lesquelles, statistiquement, le temps de résolution de la métastabilité est négligeable pour des transferts entre domaines d'horloge (la valeur τ de leurs bascules) (Alfke, 2002). Dans toutes les circonstances, l'ajout de latence et de retards d'opération imputables à la synchronisation limite les performances d'une telle méthodologie dans un contexte de conception à multiples horloges.

La seconde tendance est de développer une nouvelle architecture de FPGA possédant nativement les éléments de base pour implémenter des circuits asynchrones. Cette tendance possède un très grand potentiel, mais présente également des défis plus importants. La première génération d'architecture de FPGA asynchrone (Payne, 1996) a été basée sur la logique à retard de phase. Plus récemment, d'autres architectures, basées sur des hypothèses temporelles plus complexes comme le modèle QDI (Teifel et Manohar, 2004; Wong, Martin

et Thomas, 2003), ont été proposées. Ce modèle est une simplification du modèle DI. Ils proposent tous des techniques pour traiter la signalisation asynchrone pour permettre ou améliorer les systèmes asynchrones. Cependant, ils ont besoin d'un nouvel ensemble d'outils d'EDA, de nouvelles connaissances techniques et d'un placement et routage très précis pour se conformer à certaines hypothèses temporelles (sauf pour le modèle DI). De plus, une utilisation générique de ces architectures exige une quantité d'efforts significative et mène dans la plupart des cas à des améliorations marginales. Finalement, même si certaines de ces architectures sont semblables aux architectures synchrones (Teifel et Manohar, 2004), elles ne peuvent pas être facilement exploitées par les concepteurs synchrones de FPGA, ce qui ralentit leur acceptation. Par exemple, Achronix, le producteur de FPGA connu comme celui produisant les FPGA les plus rapides au monde, a adopté une approche avec des circuits basés sur un pipeline QDI à grains fins en accord avec l'architecture de Teifel pour FPGA asynchrones (Teifel et Manohar, 2004). La solution d'Achronix est proche des besoins du concepteur. Cependant, conformément à ce que nous verrons dans les prochaines sections, nous proposons une approche intermédiaire. Cette approche intermédiaire est intéressante pour simplifier les hypothèses temporelles QDI qui sont difficiles à supporter de manière transparente dans un FPGA nativement synchrone. Bref, nous proposons une architecture programmable capable de soutenir des transferts asynchrones dans une architecture de FPGA fondamentalement synchrone.

3.3 Requis pour une architecture GALS

Dans cette section, nous regardons les éléments requis pour masquer la métastabilité dans les communications asynchrones. Le premier module constitutif exigé est l'ECM, lequel est essentiel pour pratiquement tous les protocoles de poignée de main (Muller et Bartky, 1959; Sparsø et Furber, 2001). Cet élément et sa table de vérité apparaissent à la figure 2.8. Il est à noter que nous employons l'implémentation Berkel de l'ECM avec les outils standards CMOS (Shams, Ebergen et Elmasry, 1998), puisque cet élément est adapté à nos besoins.

Fondamentalement, l'ECM effectue une opération « ET » logique sur des événements : sa sortie est ajustée à la valeur de ses entrées lorsque celles-ci sont identiques, alors que sa

sortie est maintenue à sa dernière valeur lorsque les entrées diffèrent. Étant donné que l'implémentation de cet élément est basée sur un verrou transparent, les conditions nécessaires pour masquer la métastabilité sont également nécessaires à son bon fonctionnement (les conditions seront présentées en détail à la section 3.5). Il est bien connu que l'ECM peut attendre la résolution de métastabilité lorsqu'il est utilisé de pair avec un protocole de communication (protocoles de poignée de main) (Chattopadhyay et Zilic, 2005; Muller et Bartky, 1959; Sparsø et Furber, 2001). Dans ces circonstances, le protocole peut soutenir des retards de communication incluant les retards aléatoires du routage dans un FPGA commercial (Quoc Thai et al., 2002; Traver, Reese et Thornton, 2001). Ainsi, dans un contexte de protocole de communication, l'ECM offre la possibilité d'établir des circuits de communications asynchrones. L'ajout de cet élément représente un changement minimal à l'architecture d'un FPGA pour permettre des transferts asynchrones.

Un autre élément stratégique exigé pour certaines fonctionnalités évoluées des communications asynchrones est un MUTEX (Sparsø et Furber, 2001). Il convient de noter que pour ce chapitre, nous emploierons la dénomination EME (**É**lément **M**utuellement **E**xclusif) pour faire référence au MUTEX proposé. Ce vocable est crucial pour distinguer notre travail (construction et résolution de la métastabilité) de ce qui a déjà été proposé dans la littérature. En conséquence, pour augmenter l'efficacité et la fonctionnalité de l'architecture proposée, nous proposons d'ajouter un circuit spécialisé qui combine deux ECM et quelques portes logiques pour former un EME. Pour le circuit formé, les ECM ont besoin d'un P&R précis pour garantir la synchronisation des signaux d'EME, ce qui doit être fait au moment de la conception. Ce circuit est donc peu propice aux applications P&R automatisées. Tous les détails au sujet d'EME et des équations sous-jacentes seront traités en détail à la section 3.5.

Une fois l'ECM et l'EME disponibles, ils peuvent être utilisés pour implémenter des composants plus évolués, tels que les arbitres et les synchroniseurs (Moore et al., 2002; Sparsø et Furber, 2001). Quelques autres éléments de base tels que la ligne à retard, un oscillateur local et un détecteur de front peuvent s'avérer utiles dans un circuit asynchrone (Moore et al., 2002; Seitz, 1980). Cependant, il est inutile d'ajouter de tels composants à l'architecture puisqu'ils peuvent être implémentés avec les ressources existantes du FPGA

(Najibi et al., 2005). À ce stade, il convient de noter que d'autres MUTEX asynchrones ont été proposés dans la littérature (Seitz, 1980; Semiat et Ginosar, 2003; Van Berkel, Huberts et Peeters, 1995). Nous avons choisi de proposer notre EME pour deux raisons principales malgré sa complexité : la pluralité fonctionnelle de l'EME/ECM permet à la fois l'arbitrage et le support de protocole de communication, ce qui réduit au minimum la quantité de changement. De plus, l'ECM peut être diffusé dans tout le FPGA, alors que la fonction d'arbitrage peut être limitée à une partie de ce dernier. De cette façon, il sera possible de mettre en application des familles de FPGA ciblant différentes applications basées sur le nombre d'arbitres disponibles. En outre, il doit être clair qu'une telle architecture ne vise pas particulièrement les applications d'arbitrage asynchrone (Sproull, Sutherland et Molnar, 1994; Varshavsky et Marakhovsky, 2002). Cependant, l'EME proposé, possédant une probabilité de métastabilité extrêmement basse (dont les calculs seront présentés plus tard), est disponible si cette fonctionnalité est requise par une application.

Pour le reste de la thèse, nous supposerons que la fréquence et la phase du signal d'horloge de réception ne sont pas fixes par rapport à la fréquence et la phase d'émission de données. Cependant, nous supposerons que la fréquence du signal d'horloge de réception est plus grande que le débit attendu sur un canal de communication de sorte que toutes les émissions de données puissent être récupérées correctement.

3.4 Architecture proposée

Dans cette section, nous discuterons en détail de l'architecture proposée. Nous regarderons en particulier la façon dont l'ECM et l'EME peuvent être incorporés au FPGA. Étant donné les différentes architectures offertes par les divers fournisseurs de FPGA et afin de maintenir le caractère général du travail, nous considérons l'architecture de FPGA du type île. Cette architecture générale est illustrée à la figure 3.1.

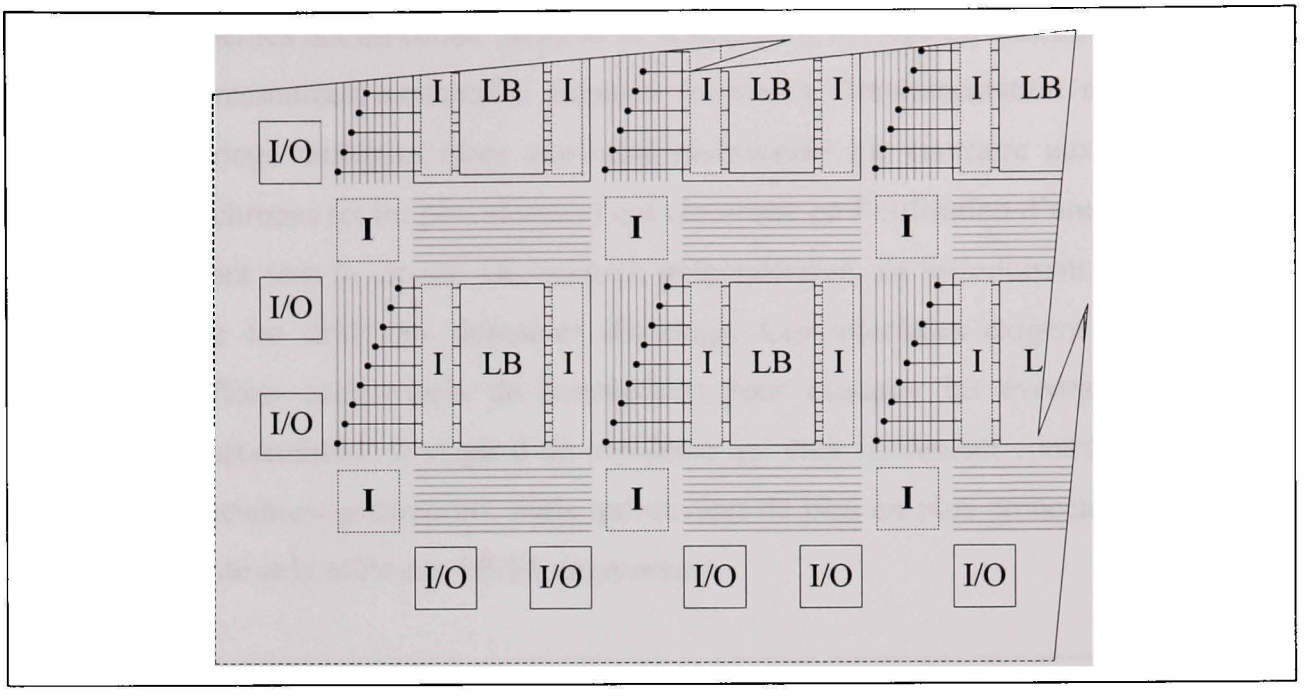


Figure 3.1 Coin inférieur gauche d'un FPGA du type île. Les FPGA de Xilinx sont construits suivant cette philosophie et possèdent l'avantage d'être extensibles.

La figure 3.1 illustre le coin inférieur gauche d'une matrice de blocs logiques, LB, (**L**ogic **B**lock) configurables entourés par des liens d'interconnexions, I, (**I**nterconnections) locaux et globaux. Ces canaux d'interconnexions permettent d'accéder à un LB ou une autre ressource déterminée. Les ports d'I/O sont, pour une technologie plus ancienne, répartis uniformément sur le périmètre du FPGA alors qu'ils peuvent également être insérés dans la matrice pour les technologies haute performance actuelles.

Bien que les FPGA modernes incorporent également d'autres structures encore plus dédiées comme de la mémoire, RAM, (**R**andom **A**ccess **M**emory), des entrées/sorties spécialisées, des modules de traitement de signaux, DSP, (**D**igital **S**ignal **P**rocessing) ou même des unités centrales de traitement, CPU, (**C**entral **P**rocessing **U**nit), l'architecture de base constituée de LB entourés par un réseau flexible et programmable d'interconnexions demeure le cœur du FPGA. Un défi particulièrement difficile à résoudre pour ces FPGA de hautes performances et les ASIC actuels est de fournir une horloge haute vitesse à l'échelle du circuit avec une très faible obliquité (Friedman, 2001; Matzke, 1997).

Les FPGA modernes ont un certain nombre de domaines d'horloge où chacun d'eux possède des chemins (ressources) consacrés. Ceux-ci favorisent l'implémentation de systèmes à domaines d'horloge multiples alors que cette philosophie est contraire aux méthodes de conception synchrones (et les plus simples) qui consistent en l'utilisation d'une seule horloge monophasée pour tout le circuit. De manière prépondérante, ils introduisent des interfaces naturelles entre les différents domaines d'horloge. Ces interfaces exigent une attention particulière (efforts additionnels de conception) pour masquer les éventuels problèmes associés à la métastabilité. Il s'agit d'un problème qui était facilement contrôlable pour les dispositifs d'anciennes générations, mais qui devient de plus en plus préoccupant à mesure que la complexité et la taille des FPGA augmentent.

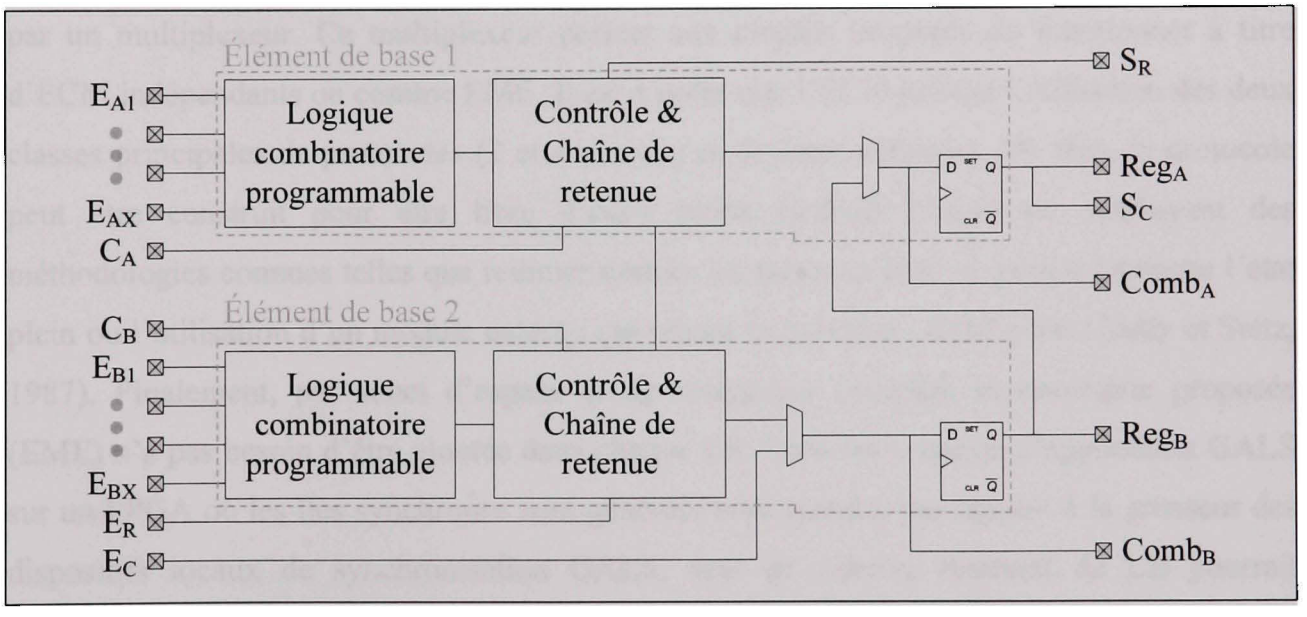


Figure 3.2 LB configurable typique de FPGA.

La figure 3.2 illustre un LB typique de FPGA. Le LB se compose d'un ou de plusieurs éléments de base. Une configuration populaire pour un élément de base est de contenir une unité de logique combinatoire, une chaîne d'addition et une bascule de sortie. Des multiplexeurs sont habituellement présents pour diriger les signaux vers ces éléments en accord avec la description matérielle. La figure 3.2 illustre deux de ces éléments de base par LB. Le choix du nombre d'éléments de base dans un LB représente un compromis au chapitre de la taille et des performances. De plus, chaque élément de base a des ressources dédiées

pour résoudre une série de problèmes logiques. Encore ici, la quantité de ressources intégrées à l'élément de base influe sur la taille et les performances du LB d'un FPGA. Il convient de noter que cette configuration est propre à chaque fabricant.

Dans l'objectif de minimiser les perturbations à apporter à l'architecture synchrone, nous proposons d'ajouter deux ECM et quelques autres portes logiques pour permettre les transferts de données asynchrones (figure 3.3). Les éléments ajoutés à l'élément de base du LB sont dépeints en gris foncé. Les éléments clés sont les deux ECM, dénotés comme des portes logiques « ET » avec l'étiquette « C ». Chaque ECM possède deux entrées, une qui vient directement de l'extérieur (l'entrée C_A ou l'entrée C_B) qui est normalement employée pour une chaîne d'addition ou pour un registre à décalage, alors que l'autre est commandée par un multiplexeur. Ce multiplexeur permet aux circuits proposés de fonctionner à titre d'ECM indépendants ou comme EME. Il est à noter que l'ECM permet l'utilisation des deux classes principales de protocoles (2 et 4 phases) et de leurs variantes. De plus, le protocole peut être construit pour être libre d'états morts (« dead lock ») en employant des méthodologies connues telles que redimensionner les tampons pour ne jamais atteindre l'état plein ou l'utilisation d'un module externe qui résout ce problème d'état plein (Dally et Seitz, 1987). Finalement, par souci d'espace et de ressources utilisées, la circuiterie proposée (EME) n'a pas besoin d'être ajoutée dans chaque LB. Dans un contexte d'application GALS sur un FPGA où les îles synchrones sont généralement grandes par rapport à la grosseur des dispositifs locaux de synchronisation GALS, seul un nombre restreint de LB pourrait contenir l'EME ce qui réduirait davantage le nombre de transistors requis pour le changement et donc minimiserait les impacts sur l'architecture.

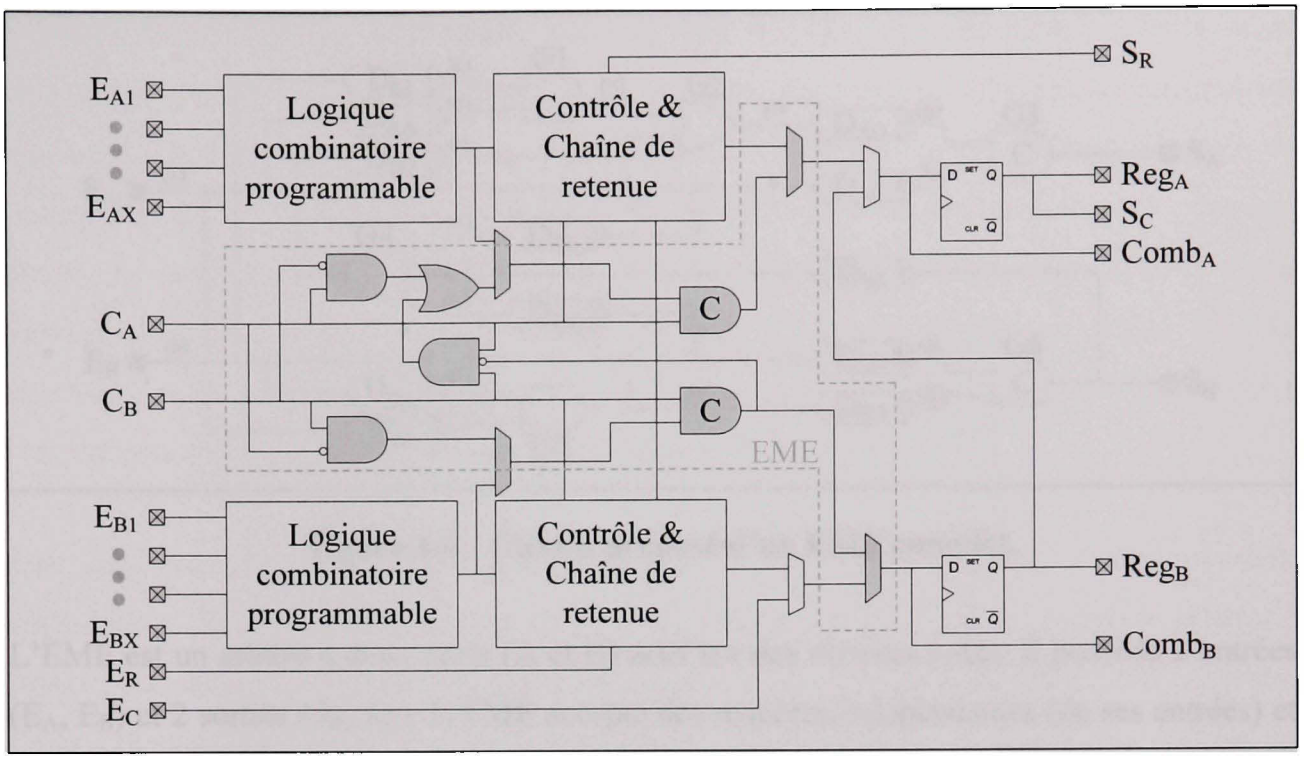


Figure 3.3 LB configurable typique de FPGA avec ECM (les composants proposés (ajoutés) sont colorés en gris et l'EME est encadré par un trait pointillé).

Dans le mode de fonctionnement indépendant, chaque ECM est autonome et possède son propre élément de mémoire. Les bascules du LB ne sont donc pas nécessaires pour garder l'état de l'élément.

Avec une telle architecture, un arbitre à deux entrées avec synchronisation locale de type « poignée de main » peut facilement être construit sur une architecture possédant nativement des EME (Sparsø et Furber, 2001), et ce, sans aucune autre considération. Un simple synchroniseur GALS basé sur cette architecture peut également être construit (Moore et al., 2002). Le synchroniseur GALS sera présenté et analysé, à titre d'exemple, à la section 3.7.

3.5 Analyse des contraintes de synchronisation

Dans cette section, nous analyserons les contraintes de synchronisation des EME incluant l'analyse de l'ECM. L'architecture présentée à la figure 3.3 est détaillée à la figure 3.4.

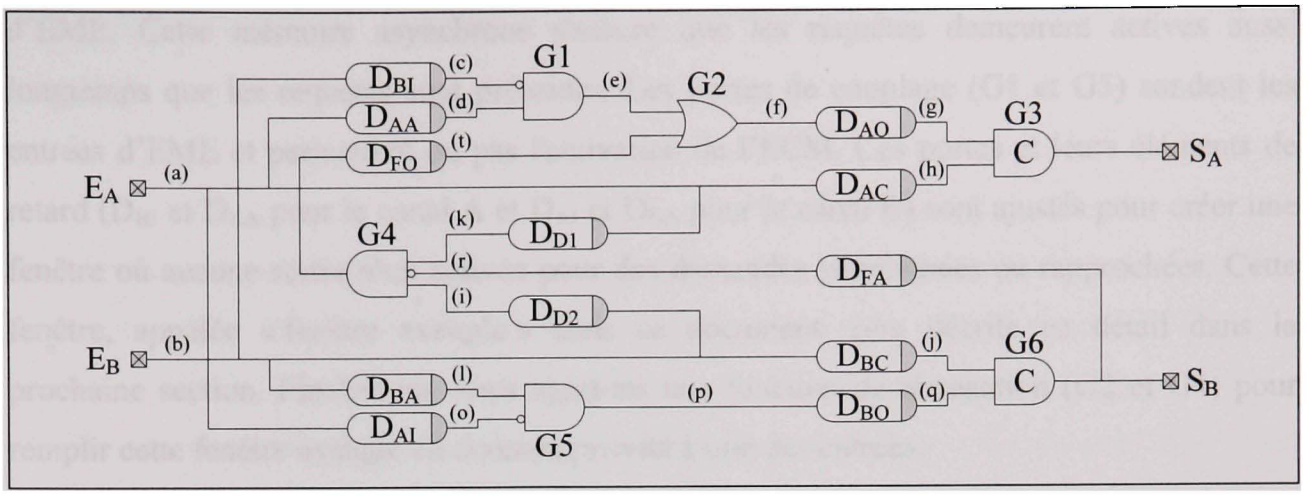


Figure 3.4 Circuit proposé d'un EME complet.

L'EME est un arbitre à deux ports (A et B) actif sur des niveaux hauts. Il possède 2 entrées (E_A, E_B) et 2 sorties (S_A, S_B). L'EME accepte des requêtes indépendantes (de ses entrées) et concède des droits (l'octroi de demandes) selon la règle d'arbitration du premier entré, premier servi. Ici, l'octroi d'une demande signifie laisser passer le signal d'entrée vers la sortie correspondante. En tout temps, un seul droit doit être consenti (tout au plus, une sortie peut être activée). L'EME est conçu pour être employé dans un contexte de protocole. Une requête ne peut pas être désactivée par l'expéditeur avant qu'elle ne soit octroyée par l'EME.

L'EME se compose de 6 portes (G1 à G6) où les opérations sont considérées comme instantanées. Leurs retards respectifs ont été combinés avec les retards de routage et exprimés en tant qu'éléments de retard externes (un ovale avec une partie grise). Notez que les multiplexeurs de sortie (ceux reliés aux sorties des ECM, figure 3.3) peuvent être ignorés pour cette analyse parce qu'ils influencent seulement la latence de sortie et non pas la fonctionnalité. De plus, les multiplexeurs incorporés (dont les sorties sont les entrées des ECM, figure 3.3) n'apparaissent pas, non plus, car leurs retards d'opération ont été intégrés dans les éléments de retard D_{BI}, D_{AA}, D_{FO} pour le multiplexeur supérieur, MUX, (**MU**ltiple**X**er) et D_{BA} et D_{AI} pour le MUX inférieur.

Cet agrégat de portes peut être fractionné en trois fonctions différentes : couplage, rétroaction et mémorisation. Les ECM (G3 et G6) sont employés pour mémoriser les signaux de sortie

d'EME. Cette mémoire asynchrone s'assure que les requêtes demeurent actives aussi longtemps que les requêtes sont présentes. Les portes de couplage (G1 et G5) sondent les entrées d'EME et permettent ou pas l'activation de l'ECM. Ces portes et leurs éléments de retard (D_{BI} et D_{AA} pour le canal A et D_{AI} et D_{BA} pour le canal B) sont ajustés pour créer une fenêtre où aucune sortie n'est activée pour des demandes simultanées ou rapprochées. Cette fenêtre, appelée « fenêtre aveugle » dans ce document, sera décrite en détail dans la prochaine section. Finalement, nous ajoutons une fonction de rétroaction (G2 et G4) pour remplir cette fenêtre aveugle en donnant priorité à une des entrées.

Une fois connecté en EME, le circuit proposé établit la priorité entre deux requêtes asynchrones. Si deux demandes arrivent presque simultanément, l'EME accorde la priorité à l'entrée du canal A ce qui permet de réduire radicalement la probabilité de métastabilité. Cette organisation explique également pourquoi l'EME contient seulement un chemin de rétroaction. Le concept de priorisation d'une entrée par rapport à l'autre est l'une des conditions essentielles pour concevoir un EME correct par construction (Chakraborty, Mekie et Sharma, 2003). Pour le reste de cette analyse, nous étudierons le circuit proposé lorsque les ECM sont employés de façon indépendante pour ensuite les étudier lorsqu'ils sont employés pour former un EME complet.

3.5.1 Analyse de l'élément C de Muller

Étant donné que l'ECM est basé sur un verrou transparent, la métastabilité peut se produire. Ce phénomène apparaît lorsqu'une séquence de signaux d'entrées forçant un changement d'état (les deux entrées à 1 pour une activation ou les deux entrées à 0 pour une remise à zéro) est assez longue pour amorcer un changement, mais sans être assez longue pour le compléter. En ce sens, nous définissons $\text{IMPULSION}_{\text{ACTIVATION}}$ comme la durée d'une telle séquence, D comme le temps minimum d'une séquence pour amorcer un changement d'état et $D + \text{Tecm}_{\text{MIN}}$ comme le temps minimum d'une séquence pour accomplir correctement le changement d'état. Les équations suivantes décrivent complètement le comportement de l'ECM :

$$IMPULSION_{ACTIVATION} < D \quad (1a)$$

$$D \leq IMPULSION_{ACTIVATION} \leq D + Tecm_{MIN} \quad (1b)$$

$$IMPULSION_{ACTIVATION} > D + Tecm_{MIN} \quad (1c)$$

La première équation (équation 1a) décrit le cas où une séquence d'entrées est trop courte pour amorcer le changement de mémoire et qui ne peut pas causer la métastabilité. La seconde équation (équation. 1b) décrit la fenêtre potentielle de métastabilité, dont la durée est représentée par le paramètre $Tecm_{MIN}$. Finalement, la troisième équation (équation 1c) décrit le cas où une séquence est assez longue pour amorcer et terminer correctement le changement d'état. Même si, traditionnellement, les réalisations de l'ECM peuvent engendrer des états métastables, ces éléments fonctionnent correctement lorsqu'utilisés dans un contexte de protocole. Dans ces circonstances, le retard de rétroaction engendré par les signaux liés à un protocole de communication est plus grand, par construction, que le paramètre de $Tecm_{MIN}$. Ainsi, l'utilisation de l'ECM en mode indépendant est viable, par construction, pour l'architecture proposée, lorsque déployée dans un tel contexte. Il est à noter que l'implémentation Berkel de l'ECM a été choisie pour ses caractéristiques de vitesse et de symétrie (Shams, Ebergen et Elmasry, 1998).

Dans le même ordre d'idées, la prochaine sous-section démontrera que nous pouvons construire un EME dont la probabilité de métastabilité est fortement réduite lorsqu'il est employé dans un contexte de protocole. Le protocole assure le respect de l'impulsion d'entrée minimum pour l'ECM (l'entrée ne peut être enlevée avant que l'arbitre active le signal de sortie approprié) où le routage global dans la matrice d'interconnexion du FPGA n'influence pas le fonctionnement (logique insensible aux retards).

3.5.2 Analyse de l'élément mutuellement exclusif

Pour assurer la complétude de l'analyse d'EME, cette analyse s'applique aux 2 portes de couplage (G1 et G5). Chaque porte de couplage peut produire 4 types différents de sortie : 1) aucune impulsion d'activation, 2) une impulsion d'activation trop courte pour amorcer une opération (équation 1a), 3) une impulsion d'activation causant la métastabilité (équation 1b),

et 4) une impulsion d'activation assez longue pour terminer l'activation de l'ECM (équation 1c). En théorie, la combinaison de ces types de sorties pour deux éléments de couplage devrait mener à 16 cas possibles. Ces cas sont récapitulés dans le tableau 3.1, où l'index de chaque symbole t_x correspond au nœud possédant la même lettre que la figure 3.4 ($\uparrow$ signifie une transition montante et $\downarrow$ une transition descendante).

Tableau 3.1 Résumé des paramètres de vérification d'EME

Cas	Transition	Commentaire (s)
Avec : $t_1\uparrow \geq t_o\uparrow$		"p" initialement à 0
C1	$t_d\uparrow \geq t_c\uparrow$	"e" initialement à 0
C2	$0 < t_c\uparrow - t_d\uparrow < D$	Pas d'activation, métastabilité sur S_A
C3	$D \leq t_c\uparrow - t_d\uparrow \leq D + Tecm_{MIN}$	Métastabilité potentielle sur S_A
C4	$D + Tecm_{MIN} < t_c\uparrow - t_d\uparrow$	Activation sur S_A
Avec : $0 < t_o\uparrow - t_1\uparrow < D$		Pas d'activation, métastabilité sur S_B
C5	$t_d\uparrow \geq t_c\uparrow$	"e" initialement à 0
C6	$0 < t_c\uparrow - t_d\uparrow < D$	Pas d'activation, métastabilité sur S_A
C7	$D \leq t_c\uparrow - t_d\uparrow \leq D + Tecm_{MIN}$	Métastabilité potentielle sur S_A
C8	$D + Tecm_{MIN} < t_c\uparrow - t_d\uparrow$	Activation sur S_A
Avec : $D \leq t_o\uparrow - t_1\uparrow \leq D + Tecm_{MIN}$		Métastabilité potentielle sur S_B
C9	$t_d\uparrow \geq t_c\uparrow$	"e" initialement à 0
C10	$0 < t_c\uparrow - t_d\uparrow < D$	Pas d'activation, métastabilité sur S_A
C11	$D \leq t_c\uparrow - t_d\uparrow \leq D + Tecm_{MIN}$	Métastabilité potentielle sur S_A
C12	$D + Tecm_{MIN} < t_c\uparrow - t_d\uparrow$	Activation sur S_A
Avec : $D + Tecm_{MIN} < t_o\uparrow - t_1\uparrow$		Activation sur S_B
C13	$t_d\uparrow \geq t_c\uparrow$	"e" initialement à 0
C14	$0 < t_c\uparrow - t_d\uparrow < D$	Pas d'activation, métastabilité sur S_A
C15	$D \leq t_c\uparrow - t_d\uparrow \leq D + Tecm_{MIN}$	Métastabilité potentielle sur S_A
C16	$D + Tecm_{MIN} < t_c\uparrow - t_d\uparrow$	Activation sur S_A

En raison des différents retards et de l'interdépendance des entrées aux portes de couplage, notre analyse indique que 7 des 16 cas sont physiquement possibles (accentués en caractères gras). De plus, notre analyse a confirmé que 3 de ces 7 cas sont équivalents (C2, C5 et C6), parce qu'une impulsion d'activation trop courte (équation 1a) produit le même effet que l'absence d'une impulsion. Par conséquent, seuls 5 cas doivent être étudiés pour faire une analyse complète, soit les cas : C13, C9, C5-C6-C2, C3 et C4. Une fois traduit en ordre

d'arrivée des requêtes sur E_A et E_B (respectivement $t_A\uparrow$ et $t_B\uparrow$) et après avoir remplacé D par le retard correspondant, ces 5 cas couvrent le spectre entier des valeurs $t_B\uparrow$-$t_A\uparrow$ (tableau 3.2). Cet ordre d'analyse correspond à une progression sur l'échelle $t_B\uparrow$-$t_A\uparrow$ où la couverture complète de spectre est une indication que notre analyse est rigoureuse.

Tableau 3.2 Ordonnancement des cas selon l'axe $t_B\uparrow$-$t_A\uparrow$

Cas	Transition	S_A	S_B
C13	$t_B\uparrow - t_A\uparrow < D_{AI}\uparrow - D_{BA}\uparrow - Tecm_{MIN} - D_{BO}\uparrow$	Inactive	Active
C9	$D_{AI}\uparrow - D_{BA}\uparrow - Tecm_{MIN} - D_{BO}\uparrow$ $\leq t_B\uparrow - t_A\uparrow \leq$ $D_{AI}\uparrow - D_{BA}\uparrow - D_{AO}\uparrow$	Inactive	Métastabilité
C5-C6-C2	$D_{AI}\uparrow - D_{BA}\uparrow - D_{AO}\uparrow$ $\leq t_B\uparrow - t_A\uparrow \leq$ $D_{AO}\uparrow - D_{AA}\uparrow - D_{BI}\uparrow$	Active via rétroaction	Inactive
C3	$D_{AO}\uparrow + D_{AA}\uparrow - D_{BI}\uparrow$ $\leq t_B\uparrow - t_A\uparrow \leq$ $D_{AO}\uparrow + Tecm_{MIN} + D_{AA}\uparrow - D_{BI}\uparrow$	Métastabilité	Inactive
C4	$t_B\uparrow - t_A\uparrow > D_{AO}\uparrow + Tecm_{MIN} + D_{AA}\uparrow - D_{BI}\uparrow$	Active (directe)	Inactive

Ces 5 scénarios de cas ont été analysés. Nous mettrons l'emphase sur les aspects importants liés à chacun des scénarios, en commençant par ceux sans métastabilité. Pour l'analyse, quelques hypothèses sont faites : 1) les demandes sont basées sur l'utilisation de protocoles de communication (une requête ne peut pas être enlevée avant que la sortie associée ne soit activée), 2) l'ECM réagit tel que décrit à la sous-section 3.5.1 et 3) les circuits sont considérés instantanés, mais le retard associé est rapporté aux éléments de retard précédents. Ces hypothèses ont été validées et ont été employées à titre de contraintes de conception des éléments de retard.

Cas C13

Pour ce cas, la sortie de la porte G1 de couplage reste à 0 tandis que la sortie de la porte de couplage G5 produit une impulsion plus grande que D + $Tecm_{MIN}$. En conséquence, la sortie

S_B est activée la première. Pour assurer un comportement approprié d'EME, nous devons vérifier que la sortie S_A ne puisse pas être activée avant la sortie S_B. Notre analyse confirme ce comportement, lorsque les contraintes énumérées au tableau 3.3 sont respectées.

Tableau 3.3 Contraintes de conception pour le cas C13

No.	Contrainte
1	$D_{BA}\downarrow + D_{BO}\downarrow + D_{FO}\uparrow > D_{BI}\downarrow$
2	$D_{D1}\uparrow + D_{FO}\uparrow > D_{AA}\uparrow$
3	$D_{AA}\uparrow + D_{AO}\uparrow > D_{AC}\uparrow$
4	$D_{BI}\downarrow + D_{AO}\uparrow > D_{BA}\downarrow + D_{BO}\downarrow$
5	$D_{AI}\uparrow + D_{BO}\downarrow + D_{FA}\downarrow > D_{D1}\uparrow$
6	$D_{BC}\downarrow + D_{FA}\downarrow + D_{FO}\uparrow > D_{BI}\downarrow$
7	$D_{BI}\downarrow + D_{AO}\uparrow > D_{BC}\downarrow$

Cas C5-C6-C2

Pour ce cas, les sorties des portes de couplage (G1 et G5) demeurent à 0 ou produisent des impulsions inférieures à D. Il s'agit du scénario de la fenêtre aveugle où le chemin de rétroaction est utilisé pour activer la sortie du canal A. Pour assurer le comportement approprié d'EME, nous devons vérifier que la sortie S_B ne puisse pas être activée avant la sortie S_A. Notre analyse confirme ce comportement lorsque les contraintes énumérées au tableau 3.4 sont respectées.

Tableau 3.4 Contraintes de conception pour les cas C5-C6-C2, C4 et C3

No.	Contrainte
1	$D_{BC}\uparrow < D_{BA}\uparrow + D_{BO}\uparrow$
2	$D_{AC}\downarrow < D_{AA}\downarrow + D_{AO}\downarrow$
3	$D_{D1}\downarrow + D_{FO}\downarrow > D_{AA}\downarrow$
4	$D_{AI}\downarrow + D_{BO}\uparrow > D_{D1}\downarrow + D_{FO}\downarrow + D_{AO}\downarrow$

Cas C4

Pour ce cas, la sortie de la porte de couplage G5 demeure à 0 tandis que la sortie de la porte de couplage G1 produit une impulsion supérieure à D+ $Tecm_{MIN}$. En conséquence, la sortie S_A est activée en premier, et ce, de façon directe (sans utiliser la rétroaction). Pour assurer un comportement approprié d'EME, nous devons également vérifier que la sortie S_B ne peut pas être activée avant la sortie S_A. Étant donné que le retrait de la requête sur le canal A exploite les mêmes chemins que ceux du cas précédent, l'analyse confirme donc un comportement correct pour l'EME lorsque les mêmes contraintes (tableau 3.4) sont respectées.

Cas C3

Pour le cas, la sortie de la porte de couplage G5 demeure à 0 tandis que la sortie de la porte de couplage G1 produit une impulsion causant la métastabilité. Même si le cas C3 provoque la métastabilité sur le canal A, cet état est transitoire puisque le chemin de rétroaction (b-i-s-t-f-g, figure 3.4) permet de résoudre la métastabilité en fournissant une impulsion d'activation assez longue dans un court laps de temps après que l'ECM ait atteint la métastabilité. Selon notre analyse, la métastabilité ne peut pas durée plus longtemps que $((D_{D2} + D_{FO}) - (D_{BI}))$. Nous avons vérifié, à l'aide de simulations, que l'EME continue à fonctionner correctement et ne pose pas de problème au niveau du protocole de communication utilisé, et ce, même si l'ECM demeure métastable ou si sa sortie converge vers un '0' logique ou vers un '1' logique. Pour assurer ce comportement, le multiplexeur de sortie doit interpréter la métastabilité sur l'ECM comme un '0' logique. Ceci représente une contrainte de conception additionnelle. Cette contrainte peut être facilement rencontrée en dimensionnant convenablement les transistors du MUX. Étant donné que le cas C3 est analogue au cas C2, les contraintes du tableau 3.4 doivent être respectées afin d'obtenir le comportement décrit.

Cas C9

Pour ce cas, la sortie de la porte de couplage G1 demeure à 0 tandis que la sortie de la porte de couplage G5 produit une impulsion causant la métastabilité. Le problème du cas C9 est

plus complexe parce que l'absence de rétroaction n'aide en rien à la résolution de la métastabilité. De plus, la métastabilité doit être interprétée par deux circuits différents, ce qui peut mener à un état mort (« dead lock ») ou une fausse reconnaissance sur le canal B. Ce problème d'interprétation peut être résolu en dimensionnant correctement la porte de rétroaction (G4) et la sortie du MUX du canal B, de façon à interpréter la métastabilité comme un niveau '0' logique. Dans ces circonstances, le cas C9 devient similaire au cas C5 et l'état de métastabilité n'a pas d'effets pervers. Cependant, si la résolution de métastabilité converge vers un niveau '1' logique, dans ce cas, il existe une très faible probabilité pour laquelle les sorties S_A et S_B deviennent simultanément actives. Pendant que la porte de rétroaction (G4) interprète l'état métastable comme un niveau '0' logique avant que la métastabilité ne soit résolue, G4 envoi une impulsion sur le canal A pour activer S_A à l'aide de l'ECM. Il est à noter que la métastabilité doit durer plus longtemps que le temps pris par l'impulsion de rétroaction pour créer l'activation du canal A, mais moins longtemps que le temps pris par le canal A pour désactiver sa sortie. Même si les conditions décrites précédemment peuvent être remplies, leur probabilité d'occurrence est très basse. Par des simulations SPICE et des valeurs de retard extraites à partir d'une topologie (décrite après), nous avons estimé que la durée de fenêtre de métastabilité (exprimée selon l'axe $t_B\uparrow$-$t_A\uparrow$) est au moins 400 fois plus petite que celle de l'ECM, exprimé par $Tecm_{MIN}$ (0.075ps et 31ps respectivement, pour analyse au pire cas). Au besoin, cette fenêtre pourrait encore être abaissée en employant les techniques présentées au prochain chapitre.

3.6 Méthodologie de simulation

Le circuit proposé a été présenté et analysé dans les sections précédentes. Dans cette section, nous simulerons le comportement à différents niveaux d'abstraction. Nous ferons d'abord une PP&RS, une simulation logique qui tient compte des retards extraits, pour montrer le comportement général de l'EME. Nous ferons, par la suite, une simulation SPICE complète pour vérifier l'exactitude des résultats obtenus.

3.6.1 Extraction des retards

La figure 3.5 illustre la topologie de l'EME utilisée pour l'extraction des retards.

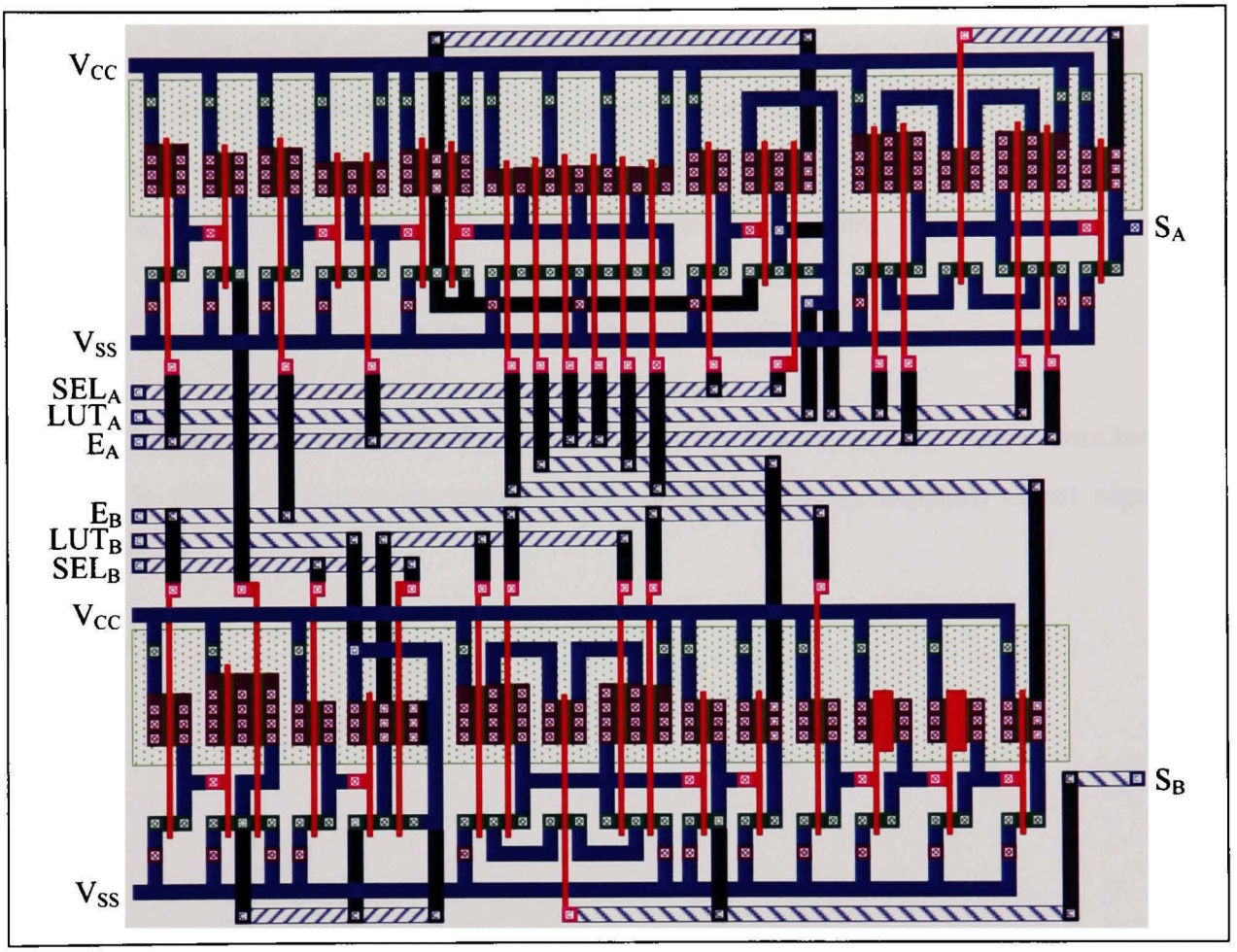


Figure 3.5 Topologie non optimisée d'EME, mais qui respecte les contraintes.

La technologie utilisée pour ce circuit complètement custom repose sur la bibliothèque TSMC à 0.18µm supportant les modèles SPICE BSIM3.2. Cette bibliothèque est adéquate pour nos besoins, même si des bibliothèques plus récentes permettraient de créer des circuits plus performants avec de plus petits retards, cette réduction n'implique aucun changement architectural et l'analyse demeure exacte. Concernant le processus d'extraction des retards, nous choisissons l'implémentation Berkel pour les ECM. Conformément aux explications précédentes, nous avons enlevé les multiplexeurs de sortie parce qu'ils n'affectent pas la

fonctionnalité, seulement la latence. De plus, nous avons exécuté quelques optimisations booléennes pour réduire des retards combinatoires. Il est à noter qu'aucun effort particulier n'a été consacré afin d'optimiser cette topologie. Par conséquent, une optimisation au niveau de la surface pourrait être possible augmentant du coup la vitesse d'opération et la densité de la topologie. Tous les retards extraits sont présentés dans le tableau III.1 (voir l'annexe III). Pour le reste de cette section, nous considérerons seulement les données du processus pour des transistors « N » rapides et des transistors « P » rapides (« FF process corner ») pour les exemples parce qu'elles mènent à des marges de retard plus petites.

3.6.2 Simulation logique après placement et routage

La figure 3.6 illustre les résultats PP&RS incluant les retards extraits. Cette simulation est basée sur le fichier standard de retards, SDF, (**S**tandard **D**elay **F**ormat) et est séparée en quatre sections (a, b, c et d).

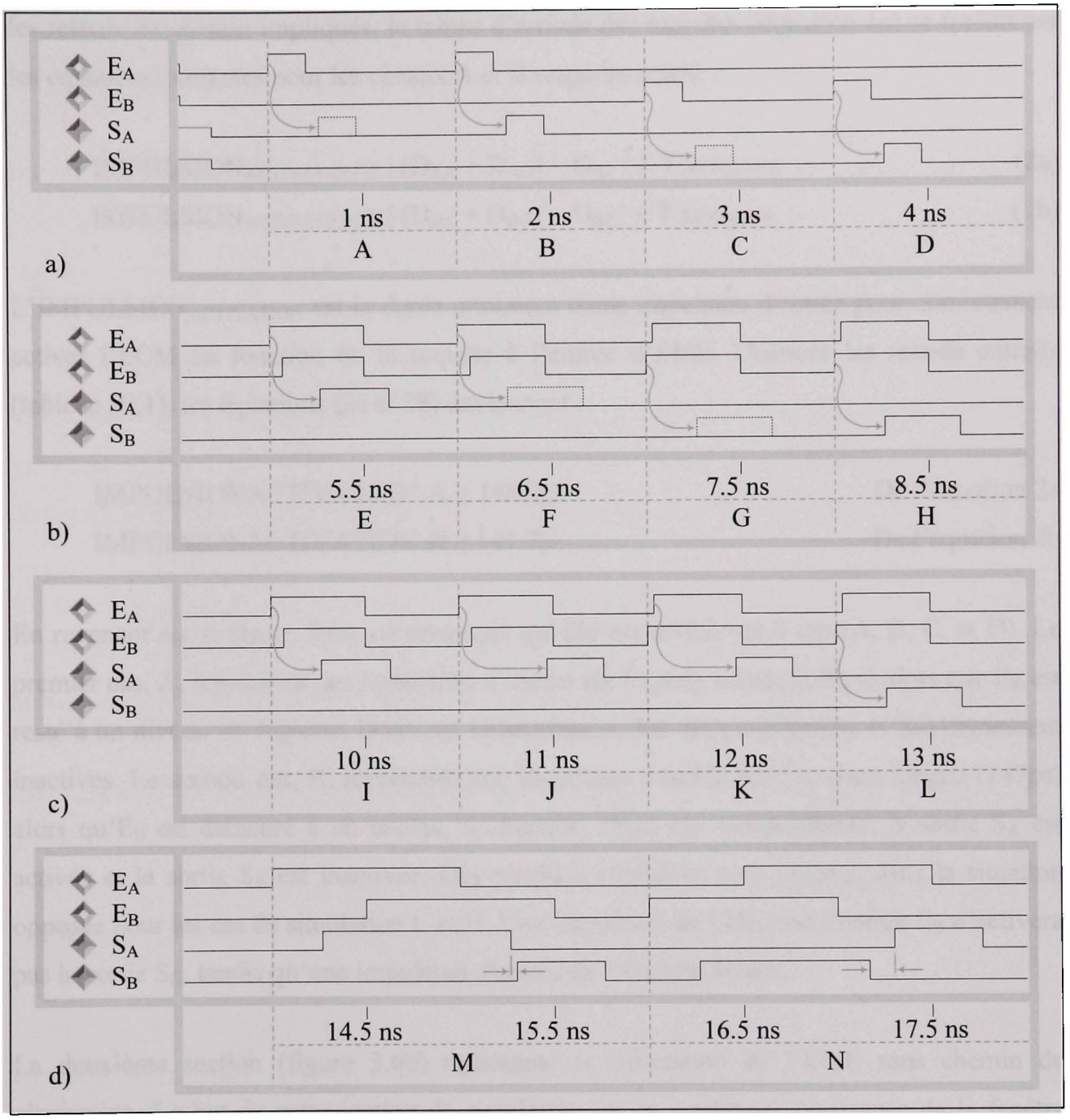


Figure 3.6 PP&RS de l'architecture proposée.
a) PP&RS de l'ECM. b) PP&RS de l'EME sans rétroaction.
c) PP&RS de l'EME complet. d) PP&RS sans chevauchement des sorties.

La première section (figure 3.6a) représente la simulation des ECM en mode indépendant. Deux aspects différents de leur comportement y sont illustrés : lorsque l'impulsion d'activation est trop courte pour lancer l'opération (équation 1a) et lorsque l'impulsion d'activation est assez longue pour lancer et terminer l'opération (équation 1c). En considérant

les retards du chemin impliqués, le temps d'arrivée des requêtes (équation 1c) se traduit par les équations suivantes pour les canaux A et B respectivement :

$$IMPULSION_{ACTIVATION} \geq | (D_{AA} + D_{AO}) - D_{AC} | + Tecm_{MIN\text{-}A} \quad (2a)$$

$$IMPULSION_{ACTIVATION} \geq | (D_{BA} + D_{BO}) - D_{BC} | + Tecm_{MIN\text{-}B} \quad (2b)$$

L'$IMPULSION_{ACTIVATION}$ est la durée minimum d'une impulsion d'entrée pour correctement activer l'ECM en fonction de la requête à l'entrée d'EME. Utilisant les retards extraits (tableau III.1), les équations (2a et 2b) deviennent :

$IMPULSIONACTIVATION\text{-}A \geq 146.9ps$ De l'équation 2a

$IMPULSIONACTIVATION\text{-}B \geq 141.7ps$ De l'équation 2b

En revenant sur la figure 3.6a, on remarque qu'elle est divisée en 4 cas (A, B, C, et D). Le premier cas, A, représente une impulsion d'entrée sur E_A trop courte (146ps) alors que E_B est resté à un niveau '0' logique. Dans ces circonstances, les deux sorties (S_A et S_B) demeurent inactives. Le second cas, B, représente une impulsion d'entrée sur E_A assez longue (147ps) alors qu'E_B est demeuré à un niveau '0' logique. Dans ces circonstances, la sortie S_A est activée et la sortie S_B est inactivée. Des résultats similaires sont obtenus dans la situation opposée pour les cas de simulation C et D. Une impulsion de 141ps sur l'entrée E_B n'activera pas la sortie S_B; tandis qu'une impulsion d'entrée de 142ps l'activera.

La deuxième section (figure 3.6b) représente la simulation de l'EME sans chemin de rétroaction. Le but de cette section de simulation est de confirmer l'existence de la fenêtre aveugle. Cette section de simulation est également divisée en 4 cas (E, F, G, et H). Basé sur les conditions d'entrées $t_B\uparrow$-$t_A\uparrow$ des cas C5-C6-C2 (tableau 3.2) et les retards extraits, la fenêtre aveugle commence avec le cas C5 et se termine avec le cas C2 avec les valeurs suivantes de $t_B\uparrow$-$t_A\uparrow$:

$$t_B\uparrow - t_A\uparrow > -86.4ps$$
$$t_B\uparrow - t_A\uparrow < 76.8ps$$

Le cas E montre un front montant sur l'entrée E_A se situant à 77ps (valeur arrondie) avant E_B, juste après l'extrémité de la fenêtre aveugle, où les conditions d'activation sont remplies et la sortie S_A est activée. Le cas F montre un front montant sur l'entrée E_A se situant à 76ps avant E_B, ce qui correspond à l'extrémité de la fenêtre aveugle (aucune activation de la sortie S_A). Le cas G montre un front montant sur l'entrée E_B se situant à 86ps avant E_A, ce qui correspond au commencement de la fenêtre aveugle (aucune activation de la sortie S_B). Finalement, le cas H montre un front montant sur l'entrée E_B se situant à 87ps (valeur arrondie) avant E_A, avant le commencement de la fenêtre aveugle, qui active la sortie S_B.

La troisième section (figure 3.6c) représente la simulation de l'EME avec le chemin de rétroaction. Les 4 cas (I, J, K, et L) utilisent les mêmes stimuli que ceux de la seconde section. La présence du chemin de rétroaction élimine la fenêtre aveugle (ensembles J et K).

La quatrième et dernière section (figure 3.6d) contient 2 cas, M et N. Ces deux cas de simulation illustrent l'absence de chevauchement des sorties pour des transitions de S_A vers S_B (cas M) ou S_B vers S_A (cas N) lorsque les contraintes de conception sont respectées (tableau 3.3 et tableau 3.4).

Le tableau 3.5 récapitule les données extraites à partir des PP&RS. Les deux premières colonnes représentent respectivement les conditions de simulation et la référence au cas de simulation de la figure 3.6. Finalement, un commentaire récapitule le résultat.

Tableau 3.5 Résumé des PP&RS

Variation(s) (ps)	Référence	Commentaire(s)
$E_A = 146$	A	S_A absent, E_A trop court
$E_A = 147$	B	Ok
$E_B = 141$	C	S_B absent, E_B trop court
$E_B = 142$	D	Ok
E_A 77 avant E_B	E	Ok
E_A 76 avant E_B	F	S_A absent, E_B trop près d'E_A
E_B 86 avant E_A	G	S_B absent, E_A trop près d'E_B
E_B 87 avant E_A	H	Ok
E_A 77 avant E_B	I	Ok
E_A 76 avant E_B	J	S_A remplit la fenêtre aveugle
E_B 86 avant E_A	K	S_A remplit la fenêtre aveugle
E_B 87 avant E_A	L	Ok
$S_A \rightarrow S_B$	M	Ok, pas de chevauchement $S_A\downarrow \rightarrow S_B\uparrow$
$S_B \rightarrow S_A$	N	Ok, pas de chevauchement $S_B\downarrow \rightarrow S_A\uparrow$

Comme nous pouvons voir, la PP&RS fournit un environnement viable de vérification. Cependant, nous devons effectuer une simulation SPICE pour considérer les variations de paramètres et la métastabilité. Les modèles de synchronisation SPICE sont suffisants pour établir un modèle pour la détection de la métastabilité. Il convient toutefois de noter que l'absence de métastabilité, pour une simulation SPICE, ne signifie pas que la conception est libre d'états métastables.

3.6.3 Simulation générale de circuits électroniques analogiques

La figure 3.7 illustre la simulation SPICE de la solution complète d'EME. En fait, nous réutilisons la plupart des cas de la PP&RS (dans un ordre différent) pour démontrer que la topologie illustrée à la figure 3.5 fonctionne tel qu'attendu.

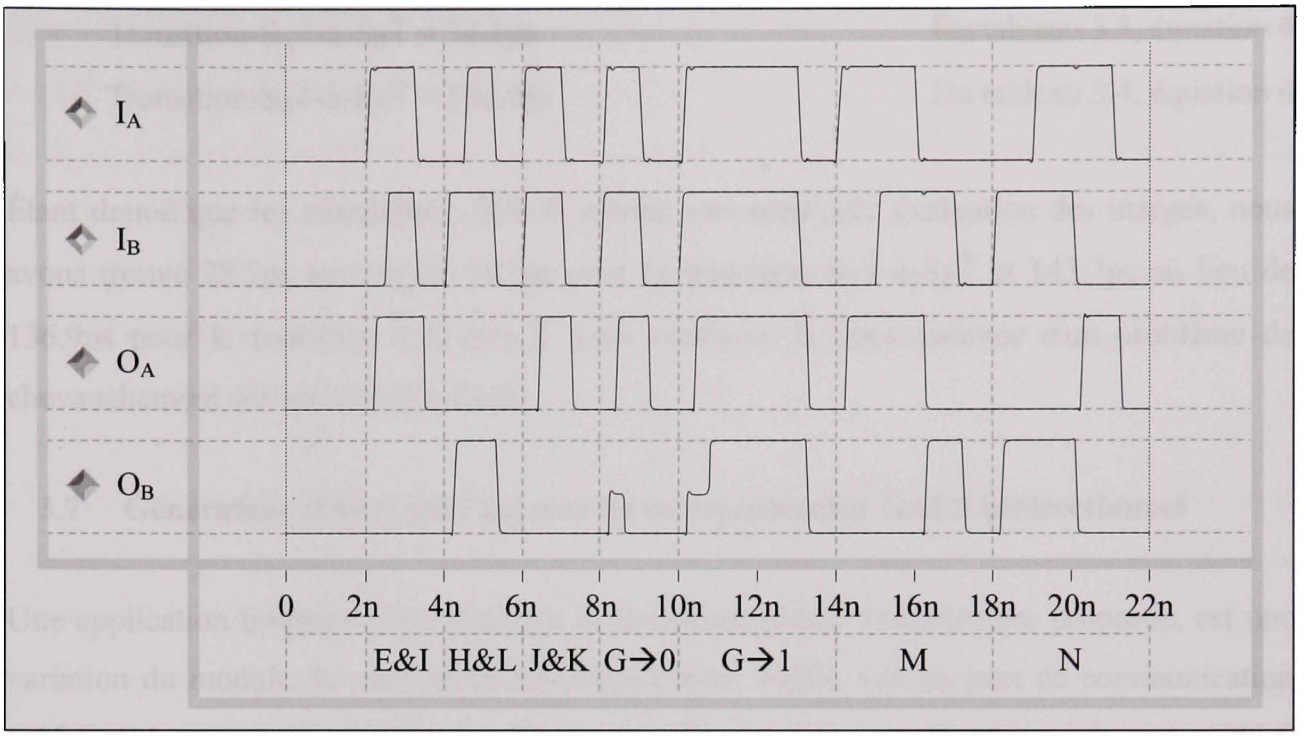


Figure 3.7 Simulation SPICE d'EME complet.
Probabilité de métastabilité non nulle, car les problèmes n'ont pas été tous masqués par la fenêtre aveugle.

Les cas de simulation A à D ont été simulés, mais ils ne sont pas illustrés étant donné que les résultats sont identiques à ceux de la figure 3.6. Les trois premiers cas de simulation (E&I, H&L et J&K) réagissent comme prévu. Le cas F (non illustré) a été ajusté pour créer la métastabilité correspondant au scénario C4 (tableau 3.2). Les résultats ont confirmé que la métastabilité a été masquée. Le cas G a été ajusté pour créer la métastabilité correspondant au scénario C9. Tel que présenté précédemment, lorsque la résolution de métastabilité finit avec un niveau '0' logique (G→0), il n'y a aucun problème. Cependant, lorsque la résolution de la métastabilité finit avec un niveau '1' logique (G→1), l'EME échoue si la métastabilité dure plus longtemps que le temps nécessaire pour activer OA par l'entremise du chemin de rétroaction (177ps).

Finalement, la simulation est complétée par les cas M et N. Des cas basés sur les équations correspondantes (tableau 3.3, équation 4 et tableau 3.4, équation 4) et sur les retards extraits, les marges suivantes sont respectivement prévues :

Transition-$S_A\downarrow$-à-$S_B\uparrow$ = 34.1ps — Du tableau 3.3, équation 4

Transition-$S_B\downarrow$-à-$S_A\uparrow$ = 136.9ps — Du tableau 3.4, équation 4

Étant donné que les simulations SPICE offrent une meilleure évaluation des marges, nous avons trouvé 28.5ps au lieu de 34.1ps pour la transition $S_A\downarrow$-à-$S_B\uparrow$ et 143.7ps au lieu de 136.9ps pour la transition $S_B\downarrow$-à-$S_A\uparrow$. Ceci confirme la non-existence d'un problème de chevauchement sur les sorties d'EME.

3.7 Générateur d'horloge d'un port de communication GALS bidirectionnel

Une application traditionnelle, illustrant la fonctionnalité de l'architecture proposée, est une variation du module du générateur d'horloge (Seitz, 1980), soit un port de communication GALS (Moore et al., 2002). La figure 3.8 illustre un port de communication GALS bidirectionnel.

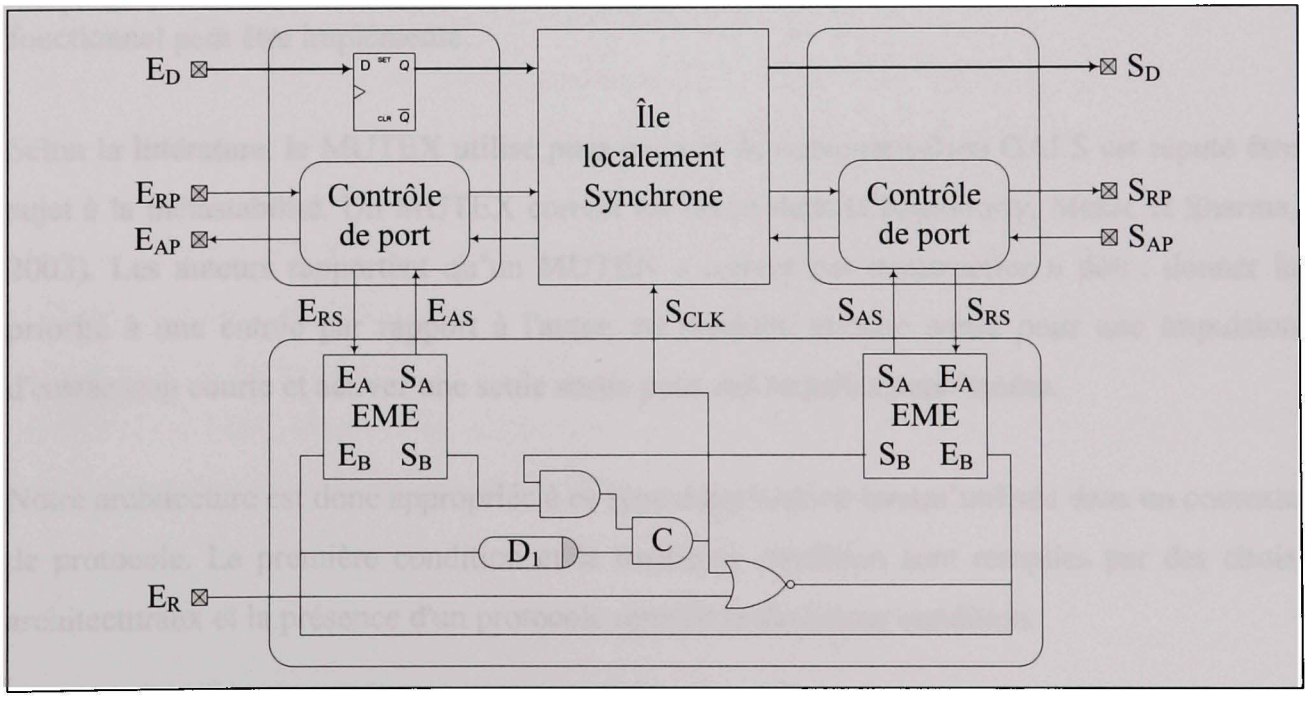


Figure 3.8 Port de communication GALS bidirectionnel.

Le port de communication GALS est une application typique qui permet la synchronisation de données transférées à haut débit. Ce module se compose d'une île localement synchrone

entourée par deux contrôleurs de ports supportant les transferts asynchrones (données (D), requête (RP) et reconnaissance (AP)). Le générateur d'horloge pouvant être arrêté fournit une horloge locale qui s'arrête pendant les transferts sur les ports. Selon la littérature, le générateur d'horloge est la source principale de métastabilité, principalement due au verrou transparent asynchrone mal activé à cause de l'effet de course des requêtes simultanées aux entrées de l'arbitre. Dans ces circonstances, nous nous concentrerons donc exclusivement sur ce module pour l'application de l'architecture proposée.

Le générateur d'horloge exige au moins un EME et un ECM pour des transferts unidirectionnels de données. Le port de communication de la figure 3.8 supporte des transferts bidirectionnels, ainsi nous avons besoin de deux contrôleurs de port et de deux EME. Le contrôleur de port est également une partie critique pour une conception GALS comportant de la logique asynchrone complexe et le plus souvent des hypothèses temporelles (dues aux transferts de données par paquets, au temps d'insertion de l'arbre de diffusion du signal d'horloge, aux machines d'états asynchrones, etc.). Malgré cela, un contrôleur fonctionnel peut être implémenté.

Selon la littérature, le MUTEX utilisé pour un port de communication GALS est réputé être sujet à la métastabilité. Un MUTEX correct est décrit dans (Chakraborty, Mekie et Sharma, 2003). Les auteurs rapportent qu'un MUTEX « correct par construction » doit : donner la priorité à une entrée par rapport à l'autre; ne produire aucune sortie pour une impulsion d'entrée trop courte et activer une seule sortie pour des requêtes simultanées.

Notre architecture est donc appropriée à ce type d'application lorsqu'utilisée dans un contexte de protocole. La première condition et la troisième condition sont remplies par des choix architecturaux et la présence d'un protocole remplit la deuxième condition.

3.7.1 Fonctionnement du générateur d'horloge

Le générateur d'horloge produit un signal d'horloge (S_{CLK}) tant qu'aucun contrôleur ne demande l'arrêt du générateur pour faire un transfert asynchrone. Chaque contrôleur de port

communique avec le module générateur du signal d'horloge par l'intermédiaire de deux fils. Le premier signal est une entrée utilisée pour demander l'arrêt de l'horloge (RS). Le deuxième signal, reconnaissance d'arrêt (AS), est un signal d'acquiescement (le signal de sortie) permettant de signaler que l'horloge est arrêtée. Du moment que le signal d'acquiescement est activé, aucun autre front montant additionnel ne sera généré par le module d'horloge; un front descendant peut cependant se produire.

Le générateur d'horloge possède un autre signal d'entrée. Ce signal (E_R) permet la remise à zéro de l'ECM et de forcer le signal d'horloge à un niveau '0' logique.

3.7.2 Modèle de simulation du générateur d'horloge

Pour établir le modèle de simulation du générateur d'horloge, nous avons fusionné deux modèles de simulation. Étant donné que l'ECM et l'EME n'existent pas dans les FPGA commerciaux, nous employons le modèle temporel SPICE pour ces éléments et le modèle après P&R pour tous les autres éléments (logique, ligne à retard et le routage entre les éléments). Le modèle combiné de cosimulation VHDL-SDF/SPICE est illustré à la figure 3.9 où les composants SPICE sont colorés en gris.

Dans ce modèle, il y a trois composants asynchrones et huit triangles qui représentent de simples tampons. Ces derniers servent uniquement à la simulation; ils garantissent le respect des contraintes de synchronisation SPICE dans le modèle de simulation final basé sur un modèle après P&R. Finalement, les entrées et les sorties d'ECM sont ajustées à 0ps et 94ps pour être cohérentes avec la simulation SPICE.

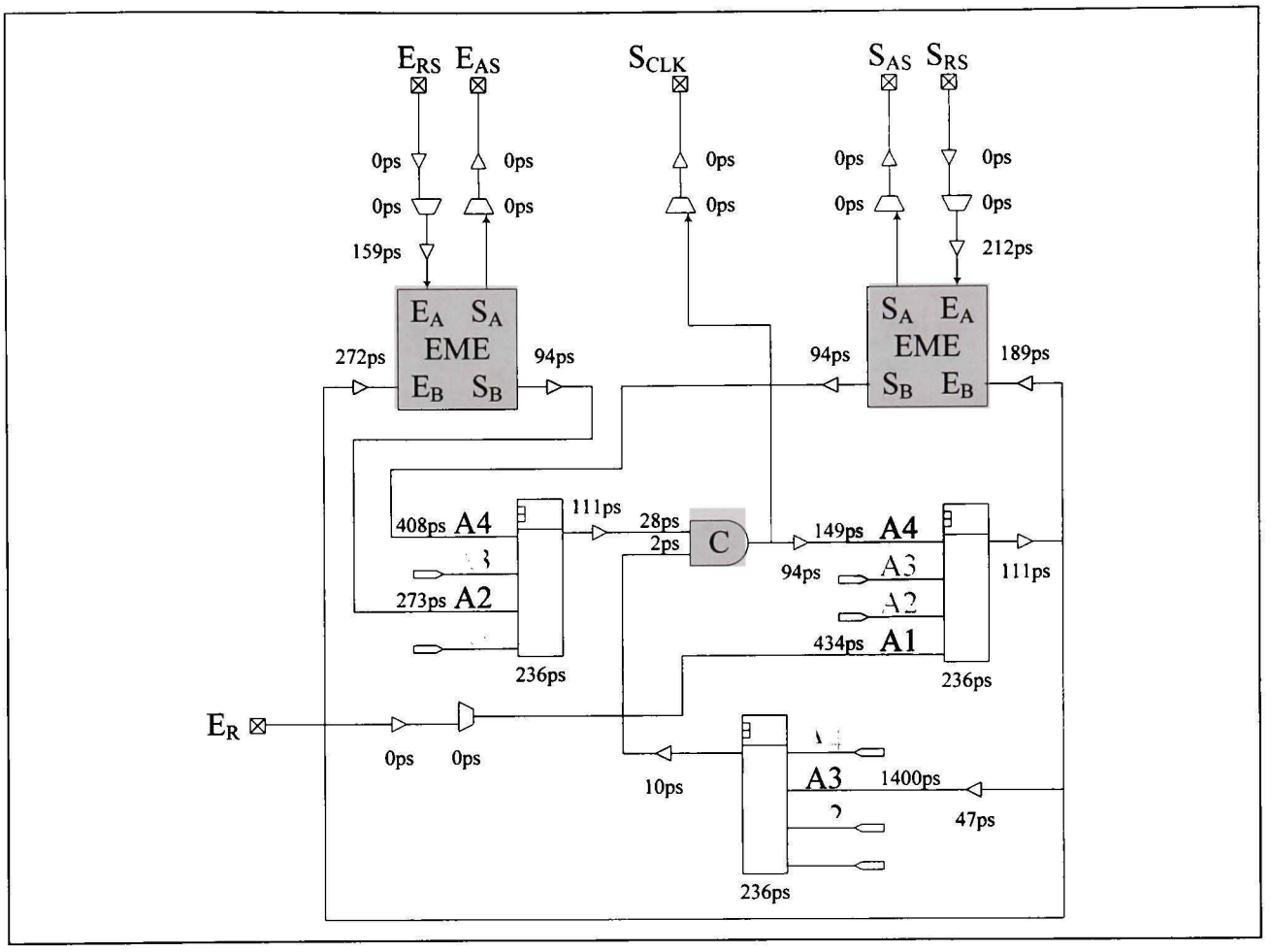


Figure 3.9 Modèle de cosimulation VHDL-SDF/SPICE d'un générateur d'horloge.

Les résultats de simulations basées sur le modèle de synchronisation de la figure 3.9 sont illustrés à la figure 3.10. Les résultats montrent que l'échelle de temps est approximativement 10 fois celle pour l'EME. Cette extension de l'échelle temporelle est due au routage du FPGA. Ainsi, le circuit proposé ne limitera pas les performances du FPGA.

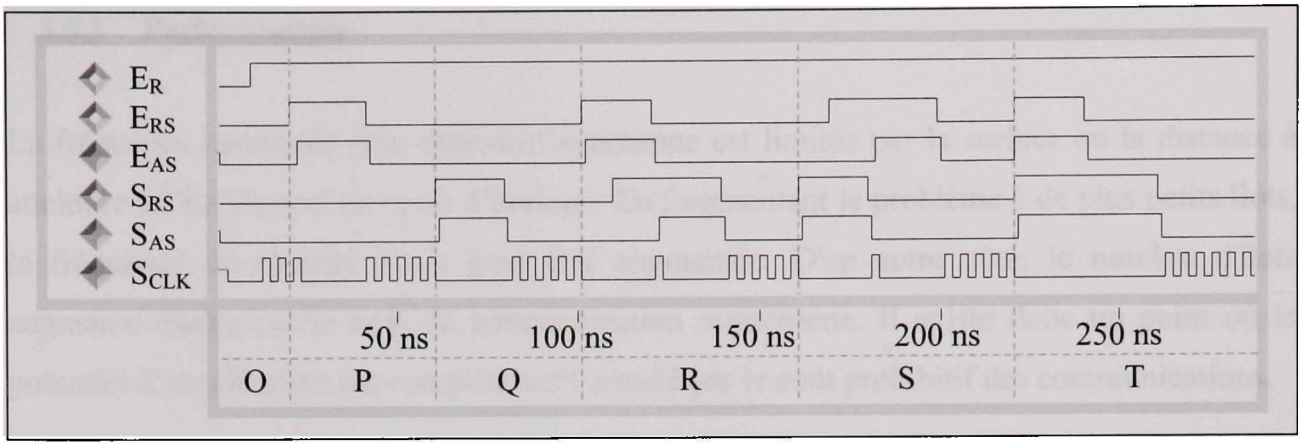


Figure 3.10 Résultats de simulation du modèle combiné du générateur d'horloge.

Les cas de simulation O à T montrent que le générateur d'horloge produit des signaux conformes, même pour des requêtes simultanées. Il est à noter que les cas R et S de simulation montrent que la génération du signal d'horloge continue conformément aux attentes malgré le chevauchement de requêtes. Ce comportement est correct puisqu'aucun signal de reconnaissance n'est accordé pour ce front montant de l'horloge. Si ce comportement est non désiré, le concepteur doit utiliser un autre type de générateur d'horloge puisqu'il s'agit d'une caractéristique du générateur et non pas d'un mauvais comportement.

3.8 Discussion

Une solution complète devrait inclure deux composants clés. Le premier composant est un outil logiciel capable de convertir et d'optimiser, automatiquement, le code conventionnel qui devrait être traduit en un modèle asynchrone. La recherche dans ce domaine est prometteuse et des solutions viables émergent (Smith, 2007). La conclusion présente également des pistes de solution. Le deuxième composant est l'innovation présentée dans ce chapitre : une architecture programmable de FPGA à laquelle nous avons greffé un ensemble minimal d'éléments de base supportant les transferts asynchrones.

3.8.1 Performances

La fréquence maximale d'un dispositif synchrone est limitée par la surface ou la distance à atteindre à l'intérieur d'un cycle d'horloge. En fragmentant le problème à de plus petits îlots, la fréquence de chacun d'eux peut être augmentée. D'un autre côté, le nombre d'îlots augmente également le coût de communication asynchrone. Il existe donc un point où le potentiel d'accélération est complètement annulé par le coût prohibitif des communications.

Pour des applications basées sur les FPGA où le routage limite les performances globales, le compromis est de permettre les systèmes GALS avec un nombre relativement restreint d'îlots synchrones. Une stratégie simple, en ce sens, est de permettre tous les domaines d'horloge présents naturellement dans l'application, sans les fragmenter davantage.

3.8.2 Impacts de variation des paramètres de fabrication

Une analyse complète d'impacts de variation des paramètres de fabrication a prouvé la robustesse de l'architecture proposée. L'architecture proposée est robuste et devrait rester utilisable même avec de grandes variations des paramètres du processus de fabrication pour les raisons suivantes. Nous avons employé des équations linéaires qui rendent le modèle viable pour de grandes variations comme celles associées à un échange entre des transistors « N » et « P » rapides et transistors « N » et « P » lents où l'impact est uniformément réparti. Nous avons utilisé des portes à tampon double avec des caractéristiques identiques pour réduire au minimum la sensibilité aux transitions (les fronts montants et les fronts descendants) communes aux échanges entre transistors « N » lents et « P » rapides et des transistors « N » rapides et « P » lents. Finalement, étant donné que les communications sont insensibles aux retards et considérant la taille de chaque îlot synchrone, les variations locales sur le dé ne devraient pas être un problème.

Si cependant un problème se posait durant la conception, un réglage des retards pour agrandir la fenêtre aveugle et masquer les événements problématiques corrigerait les impacts de variation des paramètres attribuable au processus de fabrication.

3.8.3 Limitations

Les conceptions nécessitant des hypothèses de synchronisation, comme les arbitres multi-entrées QDI (Felicijan, Bainbridge et Furber, 2003), ne peuvent pas être implémentées directement sur l'architecture proposée. Des analyses de synchronisations et configurations additionnelles peuvent être nécessaires pour ces conceptions. Par exemple, des lignes à retard peuvent être employées pour rencontrer certaines contraintes de synchronisation. Alternativement, les outils de P&R peuvent être utilisés pour assortir les retards critiques. L'analyse et la faisabilité de ces solutions sont en dehors de la portée de cette thèse.

3.8.4 Évaluation de la surface et du nombre de transistors

En partant de la topologie d'EME et du modèle SPICE, nous avons extrapolé la taille du circuit incluant les multiplexeurs (figure 3.3). Dans ces circonstances, la taille ne dépasse pas 30.8µm x 41.7µm ce qui représente 1284µm^2 pour chaque EME. Cette taille correspond à 6.2 % de la surface du LB (appelées **C**onfigurable **LB**, CLB) d'un FPGA Virtex-E de Xilinx basé sur un procédé de fabrication de 0.18µm (Weaver, Hauser et Wawrzynek, 2004). Une optimisation d'EME nous permettrait de réduire encore ce rapport. Cette proportion est beaucoup plus petite en fonction du nombre total de transistors dans le FPGA. Nous estimons que les ECM et leurs circuits périphériques, tels que représentés à la figure 3.3, exigent approximativement 90 transistors. Dans le plus grand Virtex-E FPGA, il y a 129 792 éléments de base et un total de 210 millions de transistors, ce qui correspond à 2.8 % du compte de transistor (chaque LB d'un Virtex-E contient quatre éléments de base).

De ce fait, le coût en nombre de transistors pour ajouter le circuit proposé à chaque LB est mineur. Pour une technologie plus récente, la famille Virtex-4 construite en 90nm, le V4LX200 atteint 1 milliard de transistors. Ce dispositif possède 178 176 éléments de base (chaque LB d'un Virtex-4 contient quatre éléments de base). Ainsi, des 1 milliards de transistors seuls 8 millions seraient associés aux EME, ce qui représente une augmentation de 0.8 %. Il est à noter qu'un plus faible taux peut être atteint, si nous modifions une quantité limitée d'éléments de base au lieu de modifier tous les LB. Ainsi, même si le rapport relatif

de la surface demeure relativement constant d'une technologie à l'autre, l'impact du nombre de transistors diminue. Étant donné que plus de transistors sont disponibles, ces derniers sont généralement utilisés à d'autres fins que pour l'implémentation de LB supplémentaires.

3.9 Conclusions et travaux futurs

L'architecture proposée surmonte les difficultés liées à l'émulation de composantes asynchrones sur FPGA nativement synchrone. De plus, elle offre des composants dédiés pour permettre un support minimal aux conceptions GALS pour FPGA commerciaux.

Les résultats de simulation montrent que l'ajout de peu de transistors à un LB de FPGA permet d'augmenter significativement la souplesse de ce dernier. Nous avons montré que les technologies courantes de FPGA peuvent synchroniser des données asynchrones sous condition de changements minimaux. Un générateur d'horloge pouvant être arrêté a été utilisé, à titre d'exemple, pour prouver le concept et pour prouver que l'ajout des ECM et des EME permettent des opérations GALS dans les FPGA commerciaux.

Cette légère augmentation au niveau des transistors améliore de façon considérable le potentiel de performance de l'architecture sous-jacente. Nous avons noté, en particulier, une probabilité dramatiquement réduite de métastabilité pour les opérations asynchrones. Le lecteur doit comprendre que même si l'élimination complète de la métastabilité représente la situation idéale, il est impossible de l'éliminer entièrement.

Quelques éléments de vérification et d'essai ont été discutés, mais une recherche complète est un secteur d'activités futures. De la même manière, l'intégration d'outils et des exemples plus complexes peuvent également faire partie de travaux futurs pour rendre cette architecture plus accessible aux concepteurs GALS. Par contre, le prochain chapitre mettra de l'avant une innovation majeure dans le domaine du masquage complet de la métastabilité.

CHAPITRE 4

Masquage des aléas temporels

Pour aborder le masquage des aléas temporels, il faut laisser de côté les FPGA GALS et regarder le problème des aléas temporels de façon globale sans être contraint par une architecture donnée.

4.1 Introduction

Une unité de masquage des aléas temporels est un module à une entrée et une sortie masquant les transitions de fréquences élevées considérées nuisibles pour le circuit en aval, tout en conservant les signaux de fréquences moins élevées intacts. À la manière d'un filtre passe-bas, la sortie du module copie le signal d'entrée lorsque la fréquence de ce dernier est inférieure à un seuil déterminé. Dans le cas contraire, le signal est filtré empêchant les transitions de fréquences trop élevées à l'entrée du module d'être reproduites à sa sortie. Le seuil choisi, représentant la « fréquence de coupure » du « filtre », sert de pivot de décision de filtrage et est ajusté en fonction de l'application à laquelle elle s'applique. Il convient toutefois de noter que le module présenté n'agit pas à l'instar d'un filtre en fréquence, mais bien à titre de filtre de largeur d'impulsion.

Le problème fondamental pour le masquage des aléas temporels réside à la fréquence de coupure (F_c, figure 4.1a ou zone B, figure 4.1b), dans le passage de l'état passant (zone A, figure 4.1) à l'état bloquant (zone C, figure 4.1) et vice versa (Gargour, Ramachandran et Bensoussan, 1993). Idéalement, ce passage doit être franc : la transition doit se faire sur un intervalle de temps infiniment petit où F_a tends vers F_b (Li et Nowrouzian, 2002). Un tel filtre est référencé par l'appellation « mur de brique ». Il peut être réalisé en multipliant le signal d'entrée par une fenêtre rectangulaire (figure 4.1a) dans le domaine fréquentiel ou par la convolution avec un sinus cardinal dans le domaine temporel. La moindre divergence, par rapport à cet idéal, engendre la création de la zone B (figure 4.1b) et le problème d'interprétation des seuils aux interfaces A-B et B-C.

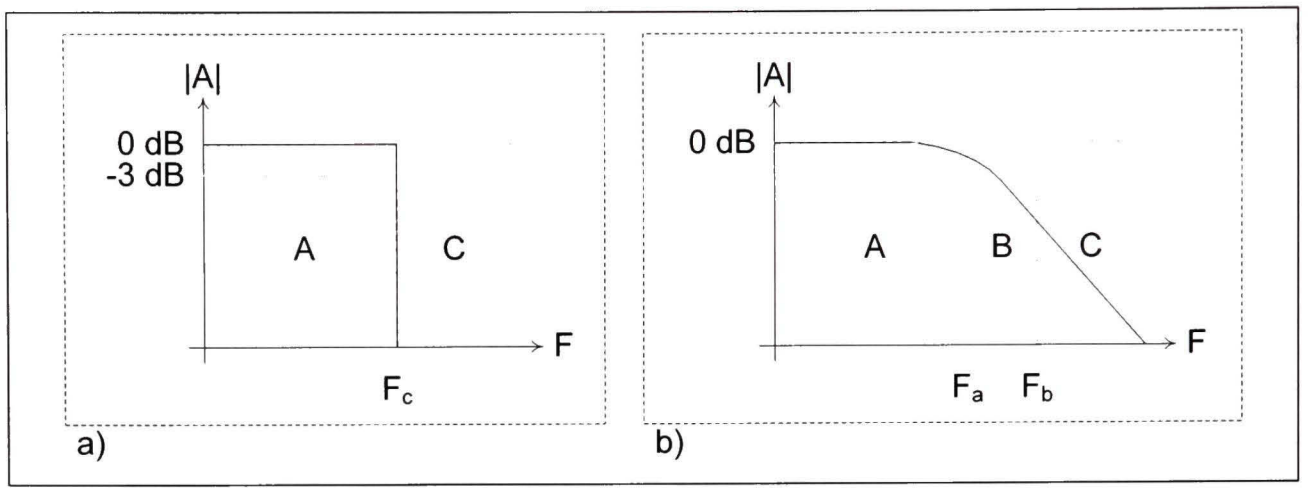


Figure 4.1 Réponse en fréquence.
a) Filtre idéal. b) Filtre passe-bas théorique.

Pour illustrer la difficulté d'effectuer un filtrage idéal, il suffit de prendre l'exemple du sinus cardinal : une fonction théoriquement infinie. Ainsi, pour compenser l'utilisation du sinus cardinal (représentation finie) dans le filtrage, il faudrait avoir un historique infini du signal et prédire son évolution jusqu'à la fin des temps. Un filtre causal peut dépendre de l'historique d'un signal et de son présent, mais jamais de son futur. Dans ces circonstances, il est impossible d'obtenir ces informations et donc de faire un filtrage idéal (Chen, 2003; Gargour, Ramachandran et Bensoussan, 1993). Cependant, il est possible de simuler une utilisation non causale d'un filtre en ajoutant un élément de retard sur le signal de façon à le retarder et ainsi « connaître son futur ». Cette approche n'est pas toujours possible, mais lorsque l'application le permet, le filtrage peut être amélioré. Nonobstant cette amélioration, un filtre du type « mur de brique » nécessite tout de même un gain infini (Rabaey, Chandrakasan et Nikolic, 2003). Pour des composants réels, qui ont un gain fini, l'atteinte d'un tel gain passe nécessairement par une rétroaction positive : le principe d'emballement connu sous le nom de l'effet Larsen (Schlarmann, Malik et Geiger, 2002).

Les limites (filtrage, gain, etc.) dans le domaine du masquage des aléas temporels sont usuellement contournées par l'hypothèse que le signal à filtrer est un bruit dont la fréquence est beaucoup plus élevée (un ordre de grandeur) que celle du signal porteur d'informations utiles. Les concepteurs forcent ainsi le signal à demeurer dans la zone A et le bruit dans la

zone C (Chen, 2003; Gargour, Ramachandran et Bensoussan, 1993). Cette hypothèse limite la portée des applications et exacerbe le problème de la logique asynchrone qui permet une interarrivée arbitrairement petite de signaux.

Le masquage d'aléas devrait, idéalement, permettre l'utilisation de l'ECM sans protocole de communication et ainsi accélérer les transferts d'informations. Malheureusement, dans un contexte de logique asynchrone, les signaux peuvent difficilement être contraints. Les agencements de la logique combinatoire donnent naissance à des signaux de durées arbitraires qui ne respectent pas l'hypothèse temporelle (bruit/signal) et qui provoquent la métastabilité (Kinniment, 2007). La figure 2.12 illustre ce phénomène pour un ECM où il est évident qu'un filtrage capacitif ne permet pas le masquage de la métastabilité.

L'objectif de ce chapitre consiste à créer un nouveau circuit de masquage des aléas temporels adapté aux besoins de la logique asynchrone. En ce sens, nous proposons une analyse révélant les caractéristiques requises à la réalisation d'un circuit soutenant les applications traditionnelles (hypothèses temporelles) et les nouvelles applications asynchrones. Cette analyse est basée sur des conditions similaires à celles présentées au chapitre précédent pour construire un EME sans métastabilité (Chakraborty, Mekie et Sharma, 2003). Bien que nous proposions la réalisation d'un filtrage parfait en éliminant tous les aléas temporels, cet objectif ne sera pas atteint dans un contexte général, mais plutôt dans un contexte de masquage des imperfections. Il est à noter que l'élimination des aléas signifie les supprimer complètement du circuit alors que nous proposons de protéger certains signaux pour produire une sortie exempte d'aléas, c'est pourquoi nous utilisons le terme « masquage ».

Pour ce faire, nous passerons en revue certains circuits typiques connus sous le vocable « circuits de suppression d'aléas temporels ». De ces informations, nous ciblerons, par la suite, les lacunes de ces circuits par rapport aux besoins de la logique asynchrone. Puis nous proposerons une solution fonctionnelle à ce problème tout en circonscrivant les limites.

Ce chapitre est divisé en six sections. La seconde section passe en revue les circuits liés à la résolution de la métastabilité pour les circuits possédant de la mémoire et ceux de

suppression des aléas temporels afin d'établir les lacunes latentes aux hypothèses de conception. La troisième section présente l'analyse de la revue de la littérature de façon à présenter les différents éléments de solution tout en ciblant les caractéristiques requises pour assurer la fonctionnalité dudit circuit. Le circuit proposé est présenté à la quatrième section. Les cinquième et sixième sections présentent les limitations et les conclusions.

4.2 Revue de littérature

Mead et Conway ont établi que la métastabilité ne pourra pas être évitée pour des éléments de mémoire, seul le masquage des signaux problématiques peut s'avérer utile en pratique (Männer, 1988; Mead et Conway, 1980). Dans cette optique, plusieurs adaptations aux bascules D illustrent ce principe. Les brevets américains 4 929 850, 4 999 528 et 5 999 029 en sont que quelques exemples (Breuninger, 1990; Keech, 1991; Nguyen et Schultz, 1999). Conformément aux attentes, ils diminuent les probabilités de métastabilité. Cependant, le gain fini de ces solutions empêche un masquage complet. Ainsi, la logique séquentielle s'avère peu attrayante pour le masquage d'aléas temporels puisqu'en cas de métastabilité, le résultat de l'opération est incertain, et ce, pour une durée indéterminée (Männer, 1988).

Le problème est moins complexe pour l'ECM, le pendant asynchrone de la bascule D synchrone, puisqu'il permet de masquer complètement la métastabilité par l'utilisation de protocoles de communication (Sparsø et Furber, 2001; Sutherland, 1989). Toutefois, le protocole fréquemment utilisé pour ce masquage nécessite un temps de rétroaction qui influe sur les performances générales et peut être prohibitif en temps d'opération. Bref, la solution ne passe manifestement pas par un élément séquentiel pour des questions d'efforts, de performances et de la qualité de la solution (Ginosar, 2003).

Dans le domaine des circuits combinatoires, les brevets américains 5 760 612, 6 249 141, 6 356 101 et 6 894 540 illustrent le masquage logique des signaux à filtrer (Ali et Dharne, 2005; Aspacio, Bharadwaj et Nguyen, 2001; Erstad, 2002; Ramirez, 1998). Ce masquage peut être produit à l'aide d'un module composé d'une porte logique à deux entrées dont les entrées sont assignées à un signal donné et à une version retardée de ce même signal. Pour

une porte logique « ET », le signal d'entrée subit une érosion à une impulsion d'entrée positive. Par impulsion positive, nous entendons une transition rapide à un niveau logique haut alors que le signal était et retournera à un niveau logique bas. En ce sens, l'aléa positif est une impulsion positive dont le cycle actif est inférieur à la largeur d'impulsion minimum souhaitée (fréquence de coupure du filtre). Dans les mêmes circonstances, le signal subira une dilatation pour des impulsions et des aléas négatifs. Pour le circuit logique « OU », il s'agit d'une situation similaire, mais de polarité inverse. Selon les besoins, d'autres circuits logiques, tels que le circuit logique « XOR », peuvent être utilisés. La figure 4.2, expliquée en détail plus loin, illustre la simulation logique d'une partie essentielle des modules de filtrage logique : le circuit d'érosion/dilatation. Il est à noter que le signal interne S_D est illustré pour simplifier la compréhension du processus de filtrage.

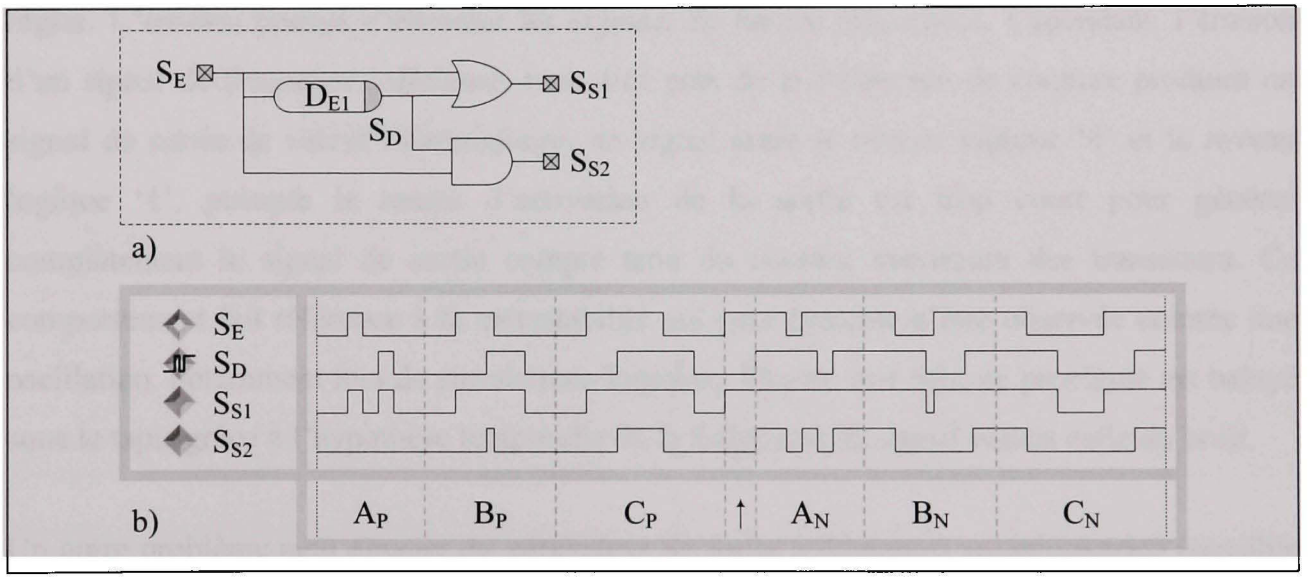


Figure 4.2 Deux types de circuit utilisés pour le filtrage logique. a) Circuit. b) Simulation logique.

L'état de l'art, dans ce domaine, se restreint à l'agencement de quelques-uns de ces modules dans le but de supprimer les hautes fréquences à l'aide d'une érosion et reconstruire le signal à l'aide d'une dilatation. Cet ordre permet, dans un premier temps, de masquer les aléas et de reconstruire le signal à l'image du signal original. En ce sens, les topologies séries (5 760 612), parallèles (6 356 101), XOR (6 249 141) et simplifiées (6 894 540) représentent

bien l'ensemble de l'état de l'art (Ali et Dharne, 2005; Aspacio, Bharadwaj et Nguyen, 2001; Erstad, 2002; Ramirez, 1998).

De retour à la figure 4.2, on y retrouve trois cas de simulation logique notés A à C, pour des transitions positives (P) et négatives (N). Les cas A représentent des aléas (à masquer), les cas C des impulsions (à conserver) et les cas B des signaux limites (un cas potentiellement problématique si le signal ne peut pas compléter la transition amorcée). De plus, le cas A_P du signal S_{S1} et le cas A_N du signal S_{S2} illustrent que les signaux générés oscillent lorsqu'un aléa temporel est présenté à l'entrée. Ainsi, les cas A et B sont critiques lors de la construction d'un circuit de masquage d'aléas temporels, nous y reviendrons sous peu.

Malheureusement, les solutions de l'état de l'art ne font que déplacer le problème sans le régler. L'érosion permet d'éliminer les signaux de hautes fréquences. Cependant, l'érosion d'un signal de fréquence inférieure, mais très près de la fréquence de coupure produira un signal de sortie de valeur intermédiaire, un signal entre le niveau logique '0' et le niveau logique '1', puisque le temps d'activation de la sortie est trop court pour générer complètement le signal de sortie compte tenu du courant maximum des transistors. Ce comportement fait référence à la métastabilité qui peut également être observée comme une oscillation, notamment lors de simulations logiques. Encore une fois, ce problème est balayé sous le tapis grâce à l'hypothèse temporelle de la fréquence du signal versus celle du bruit.

Un autre problème peut émaner du générateur de sortie à dilatation autorégulatrice (sensible aux niveaux plutôt qu'aux fronts). Cette fonction autorégulatrice permet la génération d'un signal de sortie de même durée que le signal d'entrée (maintenu par les capacités parasites accomplissant la fonction de dilatation), le tout rendant superflue la dilatation explicite du signal. Ainsi, un signal problématique (laissé dans un état intermédiaire) est conservé par l'effet de mémorisation (haute impédance ('Z')) suivant chaque impulsion d'entrée. La figure 4.3 illustre la simulation SPICE identifiant ce problème d'activation de la sortie propre au circuit formant l'état de l'art. Il est à noter que les conditions de simulations SPICE sont identiques à celles du chapitre précédent, et ce, pour les mêmes motivations. De plus, la projection de la courbe S_S (ligne pointillée) représente le signal de sortie souhaité.

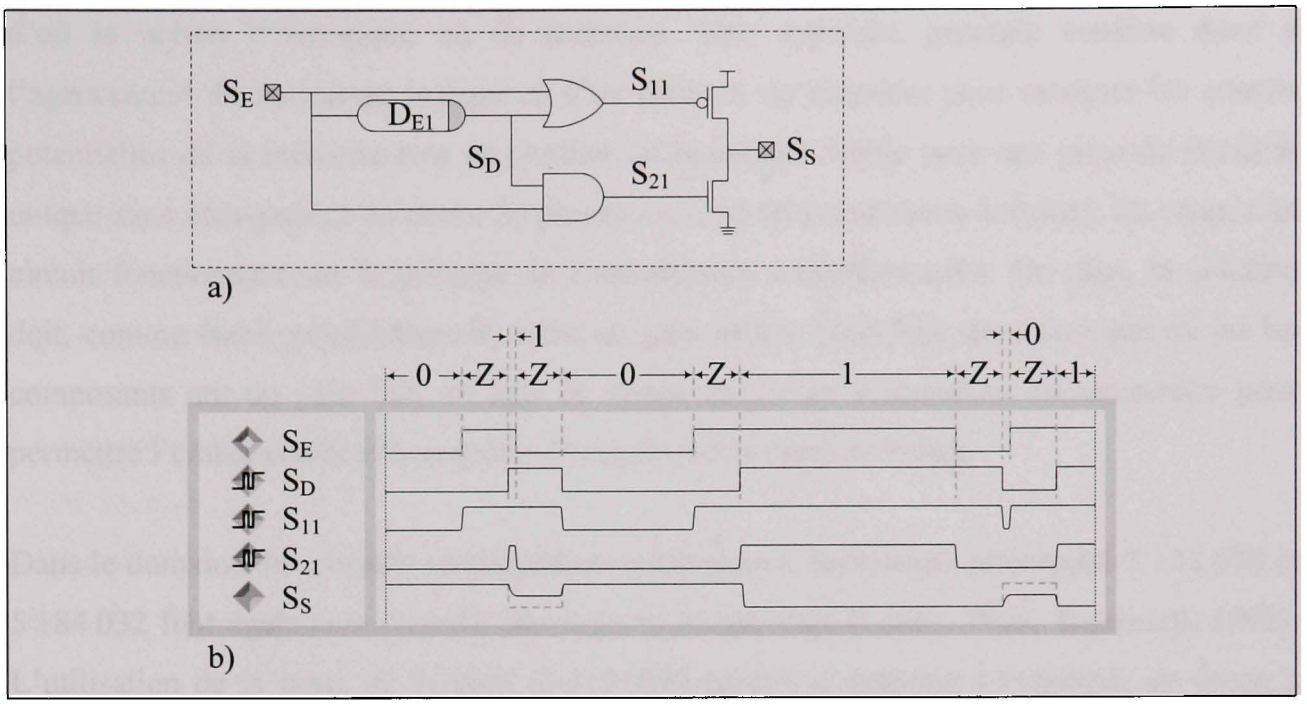


Figure 4.3 Problème d'activation de la sortie.
a) Circuit du brevet américain 6 356 101.
b) Simulation SPICE avec stimulus idéal pour ce circuit.

La figure 4.3 illustre l'incertitude potentielle émanant de l'étape de dilatation pour reconstruire le signal de sortie S_S. Pour une reconstruction, sans problème, du signal S_S tout en demeurant indépendant de la nature du signal d'entrée, il faut que l'impulsion érodée ait une durée égale ou supérieure à celle de la dilatation. Dans les faits, plus le circuit aura d'étages de dilatation (possédant de petites durées de dilatation) et plus il sera tolérant aux signaux arbitraires à son entrée. Ainsi, une approche basée sur une dilatation en plusieurs étages avec une progression exponentielle des durées de dilatation réduira la probabilité de métastabilité, mais sans jamais la masquer complètement.

Dans un contexte de logique asynchrone, ce genre de solution ne pourra pas être considéré comme viable, tant et aussi longtemps qu'il sera impossible de garantir une sortie déterministe pour un signal arbitraire à l'entrée d'un circuit de masquage d'aléas temporels. Ceci oblige la conversion d'une logique à trois niveaux ('0', '1', et niveau intermédiaire) en logique à deux niveaux ('0' et '1') où les niveaux intermédiaires sont convertis en niveau logique. Cette conversion nécessite une prise de décision basée sur l'évolution des signaux,

d'où la notion d'historique ou de mémoire. Une approche générale consiste donc à l'agencement du masquage logique et d'un élément de mémoire pour masquer les erreurs potentielles de la mémoire tout en gardant un historique viable pour une prise de décision unique sans changement en cours de processus. Une telle contrainte invalide, du coup, tout circuit fonctionnant sur le principe de l'anticipation « feedforward ». De plus, la solution doit, comme établi précédemment, avoir un gain infini. Toutefois, dans un contexte où les composants ont un gain fini, un laps de temps minimum d'opération est nécessaire pour permettre l'emballement et la création d'un gain infini après ce temps.

Dans le domaine des circuits combinatoires analogiques, les brevets américains 5 113 098 et 5 184 032 font également ressortir des aspects intéressants (Leach, 1993; Teymouri, 1992). L'utilisation de la porte de Schmitt (5 113 098) permet d'exploiter l'hystérésis de façon à supprimer les très hautes fréquences (Teymouri, 1992). La lacune de cette solution est la lenteur d'opération pouvant introduire un état intermédiaire à la sortie, une lenteur engendrée par le diviseur de potentiel nécessaire pour créer l'hystérésis. En conséquence de quoi, des signaux viables, mais rapides, n'ont pas un temps d'opération suffisant, ce qui se traduit par un état intermédiaire à la sortie. La figure 4.4 illustre une porte CMOS à hystérésis et sa simulation SPICE à différents types d'entrées.

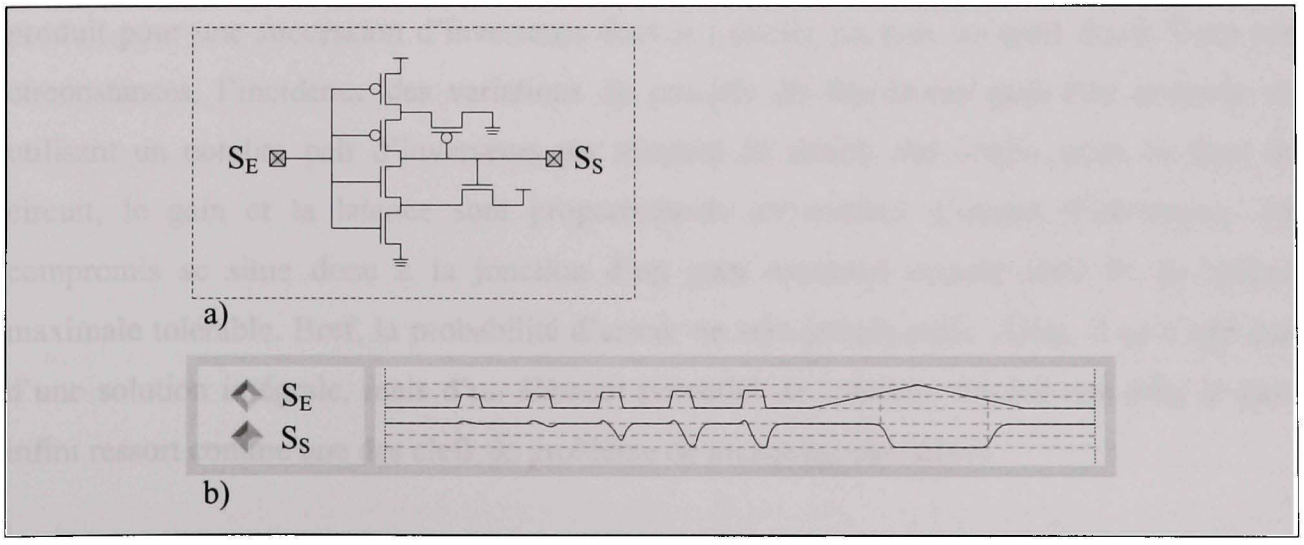


Figure 4.4 Porte de Schmitt.
a) Circuit du brevet américain 5 113 098.
b) Simulation SPICE avec stimulus idéal pour ce circuit.

La simulation SPICE de la figure 4.4 illustre la lenteur de la porte lorsque l'entrée est proche du seuil de transition. Une situation analogue se produit pour des impulsions de pleine amplitude dont leurs largeurs s'approchent de celles des aléas (non montrée). Ainsi, la porte de Schmitt déplace le problème sans le régler. Dans cette optique, seul le masquage logique peut être salvateur. De plus, la notion de largeur minimum d'impulsions, réintroduite par cet exemple, confirme son utilité dans notre quête de masquage des aléas temporels.

Le second brevet (5 184 032) présente une forme alternative de masquage réalisée à l'aide d'une succession d'inverseurs asymétriques (Leach, 1993) de façon à minimiser l'impact de la granularité du retard dans le circuit. Le retard est attribuable à l'asymétrie des inverseurs plutôt qu'au retard dû à l'activation d'un inverseur. Cette solution permet d'illustrer que l'asymétrie et le gain de chaque étage peuvent contribuer à l'augmentation de la probabilité d'avoir un signal de sortie compatible avec les niveaux logiques tout en augmentant la granularité dudit retard. Par exemple, l'agencement d'un transistor fort (« T_F ») et d'un transistor faible (« T_f ») permet de créer un inverseur déséquilibré. Lorsque le transistor « P » est plus fort que le transistor « N », l'inverseur possède un seuil bas. Le seuil haut est atteint dans les conditions inverses. Ainsi, il est possible de produire une dilatation à une impulsion positive et une érosion à une impulsion négative à l'aide une succession d'inverseurs dont la valeur des seuils alterne si le premier inverseur possède un seuil bas. L'effet inverse se produit pour une succession d'inverseurs dont le premier possède un seuil élevé. Dans ces circonstances, l'incidence des variations du procédé de fabrication peut être atténuée en utilisant un nombre pair d'inverseurs par élément de retard. Par contre, pour ce type de circuit, le gain et la latence sont proportionnels au nombre d'étages d'inverseurs. Le compromis se situe donc à la jonction d'un gain maximal compte tenu de la latence maximale tolérable. Bref, la probabilité d'erreur ne sera jamais nulle. Ainsi, il ne s'agit pas d'une solution intégrale, mais d'un élément potentiel de solution. Encore une fois, le gain infini ressort comme une des clefs du problème de masquage des aléas.

4.3 Analyse des requis

La revue de littérature a permis d'établir certaines caractéristiques nécessaires à la réalisation d'un circuit de masquage des aléas temporels dans un contexte asynchrone. Le tableau 4.1 résume ces caractéristiques.

Tableau 4.1 Caractéristiques nécessaires pour le masquage d'aléas temporels

No.	Caractéristiques
1	Masquage logique
2	Gain le plus élevé possible, idéalement un gain infini
3	Temps d'opération minimum, en cas de gain fini
4	Historique, mémoire

Tel que présenté précédemment, le masquage logique est nécessaire pour compenser les imperfections du circuit. Le gain infini instantané n'est, de fait, pas réalisable, c'est pourquoi il est nécessaire de garantir un temps d'opération minimum pour amorcer l'emballement. Finalement, l'historique est vital au bon fonctionnement du circuit pour assurer l'inertie des prises de décisions. Sans mémoire, si les entrées changent, les décisions peuvent changer et créer des erreurs potentielles.

Des caractéristiques secondaires peuvent s'ajouter à cette liste de caractéristiques globales pour une implémentation précise. Par exemple, le gain infini peut être assuré par une boucle d'emballement. Dans cette situation, trois caractéristiques secondaires seraient la présence de plusieurs signaux de prise de décision, une indépendance de la sortie par rapport à l'entrée et un emballement avant l'activation de la sortie pour assurer l'inertie de la prise de décisions.

En évaluant les circuits typiques trouvés lors de la revue de littérature par rapport aux caractéristiques nécessaires pour le masquage d'aléas temporels, nous avons construit un tableau récapitulatif. Ce tableau illustre les points faibles et les points forts de chaque circuit pour nous aider à la conception d'un circuit fonctionnel de masquage des aléas temporels

dans un environnement asynchrone. La colonne « Résultat » correspond au pas d'incrément nécessaire créant la métastabilité à la sortie pour chacun des circuits.

Tableau 4.2 Résumé des circuits de l'état de l'art

Circuit	Caractéristiques				Résultats	
	Masquage logique	Gain tendant vers l'infini	Temps minimum	Mémoire	Nombre de transistors	Incrément sur l'entrée pour observer la métastabilité à la sortie
5 760 612	X				76	1ps
6 249 141	X			X	35	10ps
6 356 101	X			X	26	10ps
6 894 540	X			X	18	5ps

Ce tableau exprime que l'état de l'art n'est pas adapté à des signaux arbitraires. Dans le meilleur cas, la métastabilité est observable en variant la largeur d'impulsion du signal d'entrée de 1ps. Ce résultat nous informe sur le fait que ces circuits ne sont pas adaptés à la logique asynchrone. Par contre, en regroupant les caractéristiques identifiées précédemment, il est possible d'implémenter un circuit de masquage des aléas temporels pour un signal d'entrée complètement arbitraire, et ce, sans contraintes sur le signal d'entrée.

4.4 Solution proposée

Cette section est composée de deux sous-sections. La première (4.4.1) présente pédagogiquement les éléments de conception de la solution finale. Les résultats intermédiaires ainsi présentés, fondamentalement indépendants de la solution finale, permettent au néophyte de comprendre chaque choix de conception. Le lecteur aguerri aurait toutefois avantage à passer directement à la seconde sous-section (4.4.2), soit celle mettant en lumière la solution proposée.

4.4.1 Éléments de conception

Un circuit de masquage des aléas temporels doit avoir un temps minimum d'opération et un très haut gain. Le concept du temps minimum d'opération alloue du temps pour l'emballement de façon à créer une divergence lorsque la métastabilité est éminente. Les inverseurs généralisés peuvent être utiles à cette fin. Un inverseur généralisé est un inverseur (dans ce cas à deux entrées) pour lequel le transistor « P » (-) et le transistor « N » (+) peuvent être activés indépendamment. Un inverseur généralisé avec hystérésis possède, en plus, la fonctionnalité de la porte de Schmitt du brevet américain (3 984 703) (Morgensen, 1976). Il est à noter que le concept d'inverseur peut être étendu : un inverseur généralisé à quatre entrées possède deux transistors « P » et deux transistors « N » contrôlables de façon indépendante et ainsi de suite. La figure 4.5 illustre le principe de création d'un temps minimum d'opération à l'aide d'inverseurs généralisés à hystérésis où le générateur du signal de sortie possède quatre transistors au lieu de deux. Ce changement par rapport au circuit de la figure 4.3 permet de considérer tout déséquilibre entre les inverseurs généralisés (court-circuit franc de la source) et la nature des aléas (positifs et négatifs). Dans ces circonstances et avec une condition d'emballement, il est préférable d'utiliser un générateur à 4 transistors.

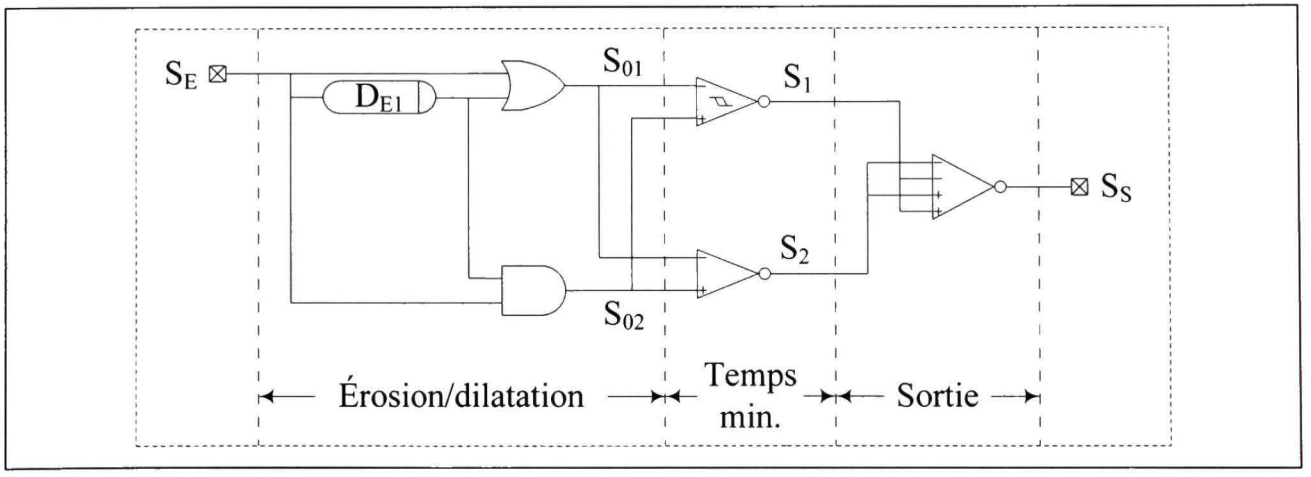


Figure 4.5 Masquage des aléas temporels à temps d'opération minimum.

Un inverseur généralisé à hystérésis est relativement lent par rapport à un inverseur généralisé normal. Ainsi, une fois jumelés, ces inverseurs détectent, à la manière d'un

détecteur de pente, toute divergence et donc toute situation potentiellement problématique pour la sortie. De plus, ils assurent une meilleure stabilité des signaux produits pour deux raisons : a) au maximum un seul signal pourra prendre une valeur intermédiaire (inverseur normal ou à hystérésis), b) il y a un second étage d'isolation, de la sortie par rapport à l'entrée, introduit par ces inverseurs, une particularité qui sera forte utile pour améliorer le gain de la solution finale. Le circuit de la figure 4.5 permet d'atteindre des résultats comparables aux meilleurs résultats (1ps) de l'étude de l'état de l'art, et ce, avec une fraction du nombre des transistors. Ainsi, il est clair que le temps d'opération minimum est un élément clé. La Figure 4.6 illustre la simulation SPICE du diagramme de la figure 4.5.

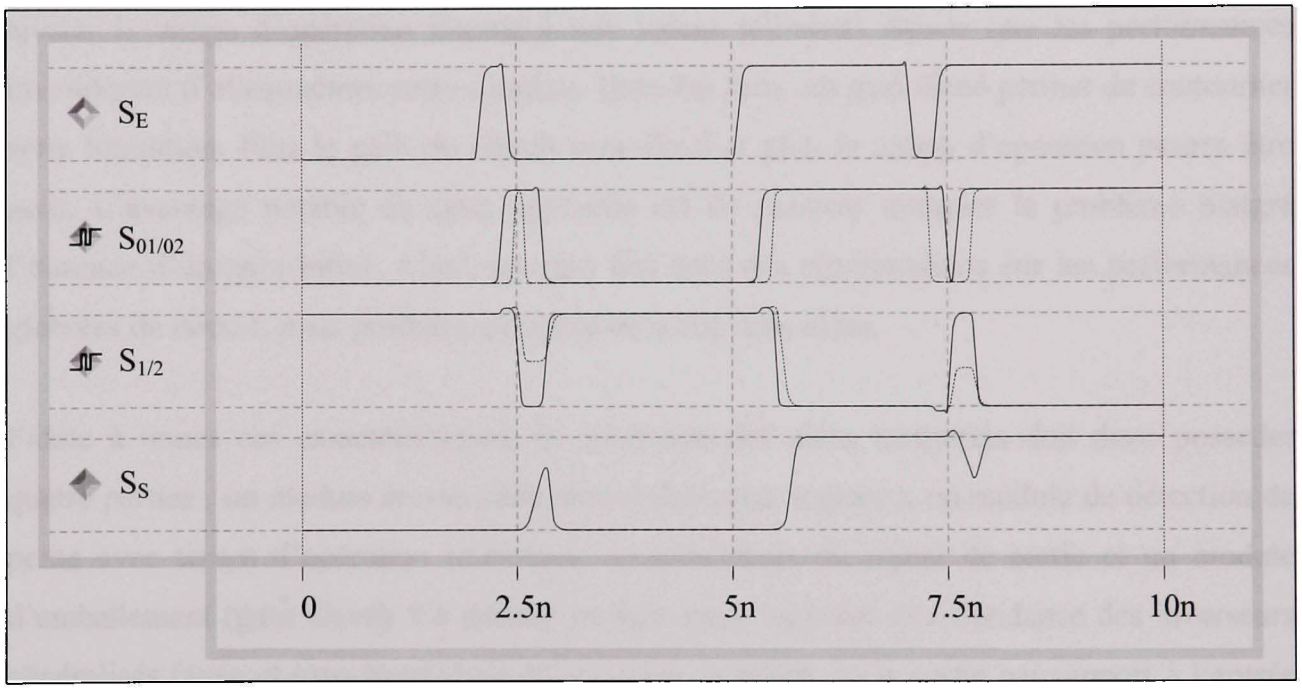


Figure 4.6 Simulation SPICE du masquage des aléas temporels à temps d'opération minimum.

La simulation de la figure 4.6 présente six courbes : le signal d'entrée (S_E), les deux signaux après les érosions négatives et positives (S_{01}/S_{02}), les deux signaux après les inverseurs généralisés simples et à hystérésis (S_1/S_2) et le signal de sortie (S_S). L'avantage de cette approche est de transformer le décalage temporel des signaux érodés en une divergence de niveau. Cette divergence n'est plus relative à la largeur d'impulsion du signal d'entrée, mais à un temps minimum d'opération (fixée lors de la conception du circuit). De cette façon, il

est possible de garantir un temps minimum pour les opérations internes. Durant ce temps, la sortie est déconnectée du signal d'entrée. Cette indépendance permet la création d'un signal de sortie valide, même si le signal d'entrée est tel qu'il provoquerait un aléa.

Malheureusement, le résultat de simulation indique clairement la création d'aléas à la sortie (S_S), même si le circuit jouit d'un temps d'opération. La solution est simple : le temps d'opération alloué au circuit étant insuffisant, il faut l'augmenter. Cependant, une augmentation à cet effet engendre une diminution des performances. De plus, en théorie, la métastabilité peut être très longue, à la limite tendre vers l'infini. Ainsi, pour assurer la viabilité du circuit, et ce, indépendamment de la nature des signaux d'entrées, il faudrait ajuster le temps d'opération interne à une valeur tellement élevée que les performances invalideront d'elles-mêmes cette solution. Dans les faits, un gain élevé permet de contourner cette limitation. Plus le gain du circuit sera élevé et plus le temps d'opération pourra être petit. L'avantage notable de cette approche est de pouvoir masquer le problème malgré l'absence d'un gain infini. Ainsi, un gain fini aura des répercussions sur les performances globales du circuit, mais produira un signal de sortie sans aléas.

Fidèle à toutes ces caractéristiques, le masquage des aléas temporels doit donc posséder quatre parties : un module érosion/dilatation (masquage logique), un module de détection de pente avec temps d'opération minimum, un générateur du signal de sortie et un module d'emballement (gain élevé). Ce dernier module peut exploiter la redondance des inverseurs généralisés (avec et sans hystérésis) et la double isolation de la sortie par rapport à l'entrée pour assurer à la fois la mémorisation et un gain très élevé. En ce sens, la redondance crée un détecteur de pente pour observer les transitions et agir dans le même sens (emballement). À titre d'exemple, le circuit d'emballement force un niveau logique '1' dans le cas de la détection d'un front montant pour une transition positive et inversement pour un niveau logique '0'. La figure 4.7 illustre un diagramme regroupant ces éléments.

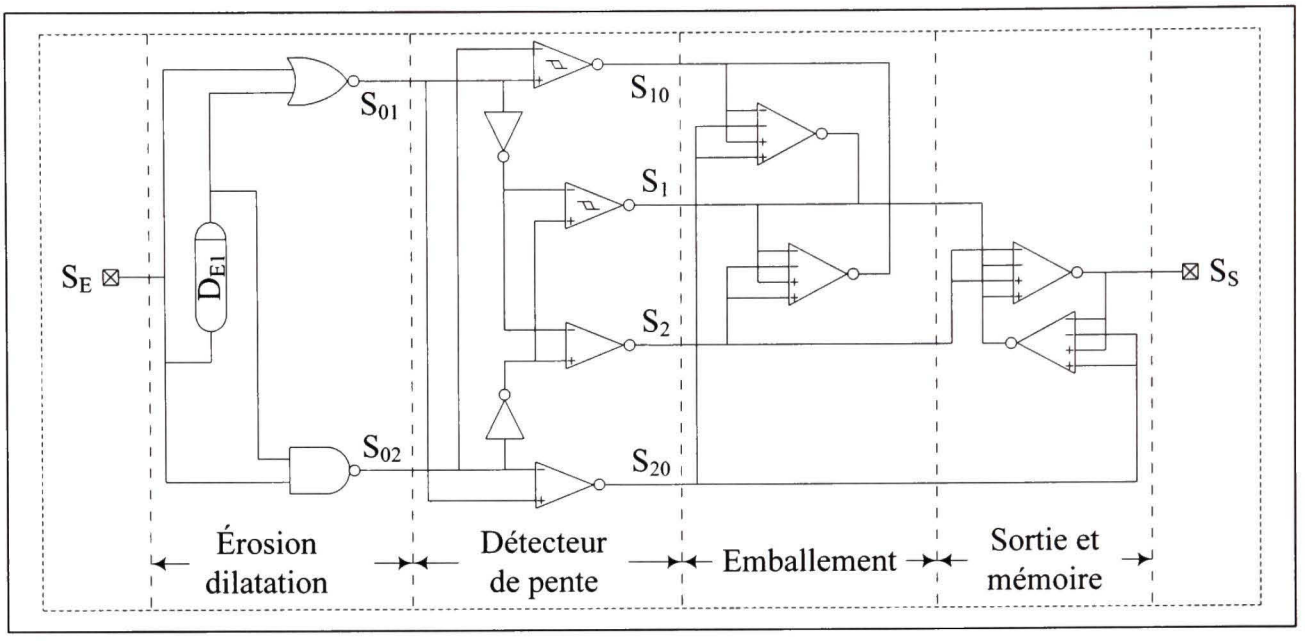


Figure 4.7 Diagramme de masquage des aléas temporels.

L'unité de masquage des aléas temporels doit stabiliser les signaux (S_1/S_2) qui générèrent le signal de sortie à l'aide de deux signaux de rétroaction alors que ces signaux sont dans un état intermédiaire. Le diagramme compte quatre inverseurs généralisés pour réaliser la fonction de détecteur de pente dans la mesure où il est non déterministe d'inverser un signal laissé dans un état intermédiaire (impossible d'assurer l'inertie du processus de prise de décision). Cette solution n'est pas intéressante malgré l'économie potentielle de transistors qu'elle engendre, puisque les signaux à inverser peuvent être dans un état intermédiaire ce qui créerait une incohérence entre les quatre signaux pour le module d'emballement. Pour le diagramme proposé, les signaux S_1 et S_2 sont identiques aux signaux S_{10} et S_{20} respectivement, mais de polarités inverses. De plus, les signaux S_2 et S_{20} ont toujours des cycles actifs supérieurs aux signaux S_1 et S_{10}. Ainsi, par construction, les signaux S_2 et S_{20} ne peuvent pas créer un problème à la sortie ce qui explique l'absence d'un circuit d'emballement pour ces signaux. De plus, l'inertie d'une transition impose la polarité de la rétroaction ce qui empêche toutes formes d'aléas de sortie tant et aussi longtemps que le temps minimum d'opération interne est respecté.

Une autre boucle de rétroaction est formée entre le signal S_1 et la boucle de mémoire conditionnelle couplant la sortie. Cette boucle conditionnelle aux signaux S_2 et S_{20} accélère

encore le circuit en ajoutant du gain lorsque la sortie commence la transition. Le concept de la boucle de mémoire conditionnel accélère également le circuit puisque la sortie n'a pas à combattre la source de courant d'un inverseur faible normalement implanté dans une boucle de mémoire à deux inverseurs en boucle (concept du brevet américain 6 249 141) (Aspacio, Bharadwaj et Nguyen, 2001). Dans le but de mieux comprendre et de motiver les éléments du diagramme illustré à la figure 4.7, la figure 4.8 illustre sa simulation SPICE. Il est à noter que pour bien visualiser l'effet de l'emballement, le retard d'érosion a été retiré du circuit et il a été remplacé par le signal S_D. Ce signal a été modelé en fonction des besoins de la simulation. Cependant, ce modelage n'a aucune incidence sur les résultats du circuit.

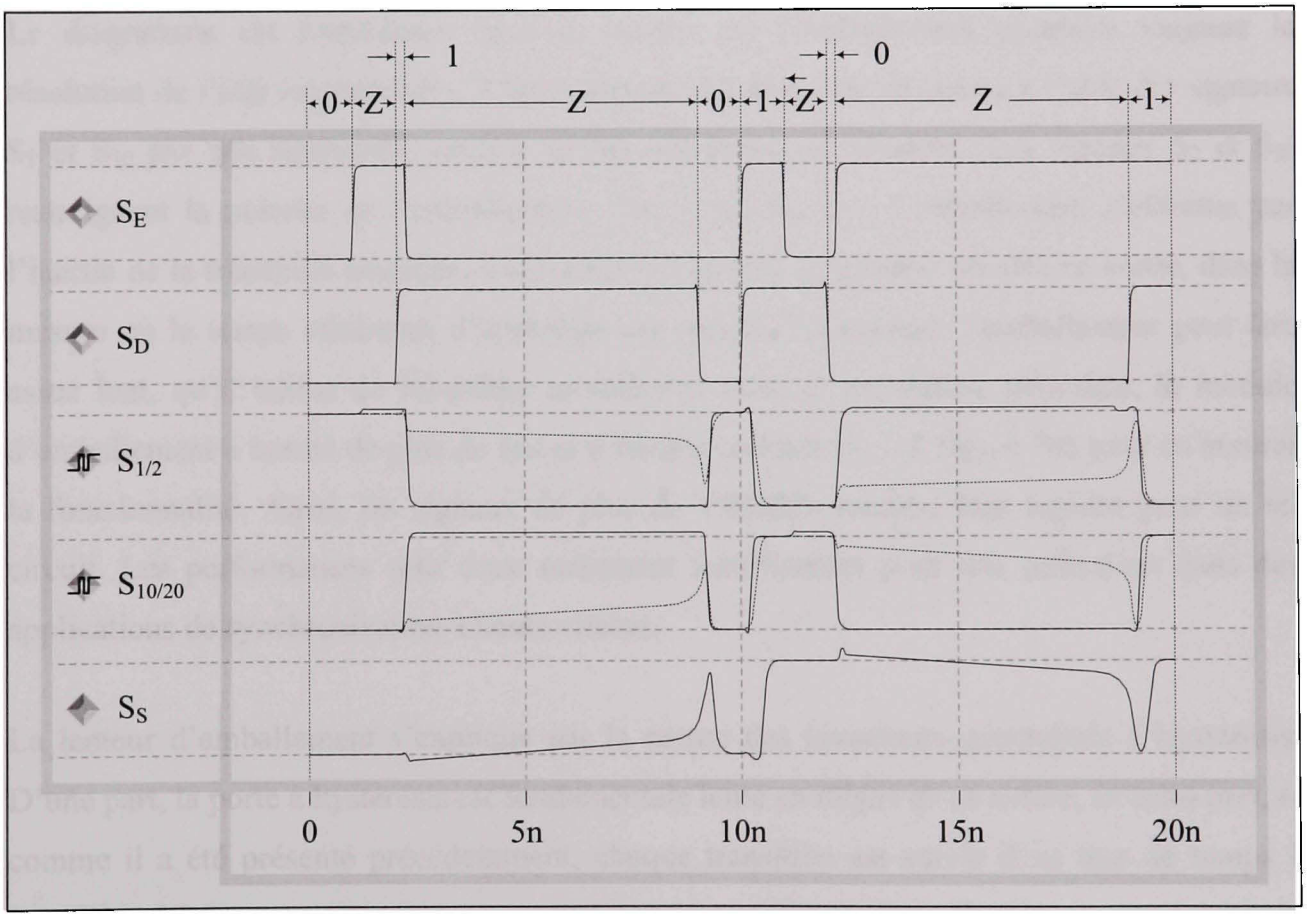


Figure 4.8 Simulation SPICE du diagramme de masquage des aléas temporels.

La simulation SPICE montre que le circuit est conforme aux attentes. Les signaux d'entrées forment l'équivalent d'impulsions érodées à 205ps (positive et négative) qui sont présentées aux inverseurs généralisés. Ces derniers génèrent les signaux S_2 et S_{20} des signaux francs qui

définissent le mode d'opération du circuit. Ils génèrent également les signaux S_1 et S_{10} de polarité inverse (lignes pointillées) et dans des états intermédiaires menant à des transitions. La légère pente de ces signaux illustre l'effet de l'emballement. La pente relativement constante entre les temps 2.5ns à 7.5ns et entre les temps 12.5ns et 17.5ns, attribuable au module d'emballement, change brusquement lorsque la rétroaction du signal de sortie est activée. Au terme de l'emballement, le signal de sortie est généré et demeure à son niveau pour le reste du temps où elle est déconnectée (haute impédance). Il est à noter que le signal S_D a été choisi pour mettre en évidence le temps minimum d'opération interne en fonction du gain d'emballement.

Le diagramme est fonctionnel dans la mesure où l'emballement garantira toujours la résolution de l'état intermédiaire. L'emballement s'amorce de lui-même à l'aide des signaux S_1 et S_{10} par une rétroaction croisée lorsqu'une pente est détectée. Les signaux S_2 et S_{20} restreignent la polarité de l'emballement. Dans tous les cas, l'emballement s'effectue par l'inertie de la transition amorcée, il est donc impossible de générer un aléa en sortie, dans la mesure où le temps minimum d'opération est garanti. Cependant, l'emballement peut être assez lent, qu'il inhibe de lui-même la solution. Pour la simulation présentée, le module d'emballement a besoin de plus de 5ns et il faudrait ajuster D_{E11} et D_{E21} à 7ns pour en assurer la fonctionnalité. Ainsi, les signaux de plus de 150MHz seraient trop rapides pour un tel circuit. Les performances sont donc nettement insuffisantes pour une utilisation dans des applications de synchronisation à haute vitesse.

La lenteur d'emballement s'explique par la nature des inverseurs généralisés à hystérésis. D'une part, la porte à hystérésis est fondamentale lente en raison de sa nature. D'autre part, et comme il a été présenté précédemment, chaque transition est suivie d'un laps de temps à haute impédance pour réserver un temps minimum d'opération. Durant ce court laps de temps, les sources sont déconnectées et les signaux demeurent stables (chargés par les capacités parasites). Si d'aventure les signaux S_1 et S_{10} sont laissés dans un état intermédiaire, le gain très faible des portes logiques couramment utilisées, dans ces circonstances, peut expliquer la relative lenteur du processus. Il est donc impossible d'espérer

un niveau de performance intéressant avec une telle configuration. Finalement, les concepts de tire-haut et de tire-bas du module d'emballement ralentissent également le processus.

Une façon d'accélérer le circuit est de remplacer le module d'emballement par un module plus performant. En ce sens, l'amplificateur différentiel est un exemple pertinent. La réponse de ce type de circuit est beaucoup plus rapide. Le prix à payer est une consommation statique de courant qui peut être limitative dans les circuits fonctionnant sur piles ou les applications où plusieurs de ces circuits sont requis. Bref, c'est l'application qui dicte la composition finale du circuit. Dans certains cas où une consommation élevée d'énergie serait intolérable, la réponse temporelle sera plus longue sans toutefois compromettre le comportement général.

Bref, tous les éléments de la solution ont été identifiés, il ne reste plus qu'à les agencer ensemble correctement pour former la solution finale, le circuit proposé.

4.4.2 Circuit proposé

En gardant en tête les éléments requis présentés à la section précédente, nous proposons un circuit de masquage des aléas temporels connectant ces éléments en créant une séparation évidente entre le temps minimum d'opération et le module d'emballement. Cette approche permet l'atteinte rapide d'un gain très élevé, et ce, sans consommation statique de puissance. La figure 4.9 illustre le circuit proposé de masquage des aléas temporels.

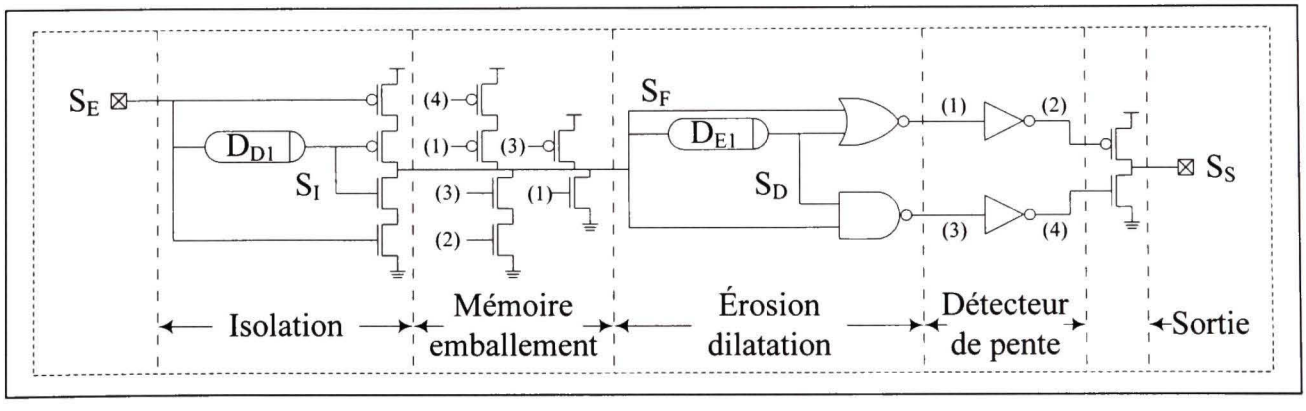


Figure 4.9 Circuit proposé de masquage des aléas temporels.

Ce circuit est composé de cinq modules connectés en série. Le module d'isolation, à gauche du module « emballement », garantit un temps d'opération minimum après chaque transition de l'entrée, et ce, pour un laps de temps correspondant à la valeur de D_{D1}. La sortie de ce premier module est mise à haute impédance pour la durée de D_{D1} après chaque transition. Le second module identifié « emballement » utilise des signaux retardés et de polarités inverses d'un détecteur de pentes et assurer l'emballement. Ce détecteur de pente est réalisé grâce à l'ajout d'inverseurs après les modules de dilatation/érosion pour détecter les transitions et activer la rétroaction positive qui se traduit par un gain infini après l'emballement. Finalement, le générateur du signal de sortie nécessite deux transistors étant donné que le module d'emballement est localisé au milieu du circuit et que le signal d'emballement (S_F) ne sert pas à la génération du signal de sortie. Ainsi, les principes présentés au diagramme de la figure 4.7 demeurent les mêmes.

Il existe plusieurs formes de réalisation du circuit d'emballement pour atteindre un gain élevé. En oubliant les formes qui consomment un courant statique (puisque ce type de circuit invalide, à lui seul, cette solution pour les applications fonctionnant sur piles et les applications d'intégration d'échelle), la solution à base de rétroaction positive demeure la seule alternative viable. L'aspect contre-intuitif de la solution est d'inclure les modules d'érosion dans la rétroaction de façon à créer deux étages de rétroaction. La première boucle sert d'élément de mémorisation et elle est formée des nœuds (1) et (3). La seconde boucle est utilisée à la manière d'un détecteur de pente et permet d'observer un changement et d'assurer un emballement rapide lorsque nécessaire. Il est à noter que cette seconde boucle agit comme élément polarisant pour chaque transition d'entrée valide qui devrait être transposée à la sortie. De plus, pour assurer la fonctionnalité du module d'emballement, les transistors des deux boucles doivent être du même ordre de grandeur. Malgré tout, le courant nécessaire pour la polarisation est inférieur au courant de la première boucle (environ 66 %) ce qui en fait une bonne solution, et ce, même pour les conceptions de faible puissance.

Étant donné que les deux boucles de rétroaction sont intercalées, la condition d'emballement sera toujours atteinte avant la condition d'activation de la sortie. De plus, étant donné que le chemin d'emballement passe par le retard d'érosion (D_{E1}), D_{D1} doit être plus grand que D_{E1}.

Naturellement, cette dépendance entre les boucles de rétroaction apporte des avantages de taille, mais complique également la conception du circuit. Cependant, une approche de conception en deux étapes suffit pour rendre la conception assez simple. Dans un premier temps, il faut construire le circuit tel quel, mais sans raccorder la sortie de l'élément de retard D_{D1}. Ainsi, le circuit de masquage des aléas temporels aura deux signaux d'entrés : le signal original S_E relié à la borne d'entrée de l'élément de retard D_{D1} et un autre signal que nous appellerons S_I relié à l'endroit où devrait se connecter la sortie de l'élément de retard. Dans ces circonstances, il est possible de générer des stimuli de façon à simuler un retard aussi long que souhaité pour établir le temps d'emballement et de trouver une valeur pour D_{D1} qui assure une largeur d'impulsion minimum du signal de sortie. La figure 4.10 illustre la simulation SPICE du circuit de la figure 4.9 raccordée tel que décrit précédemment. Il est à noter que pour cette simulation le signal S_F qui est la sortie du premier et du second module est illustré pour visualiser l'emballement du circuit.

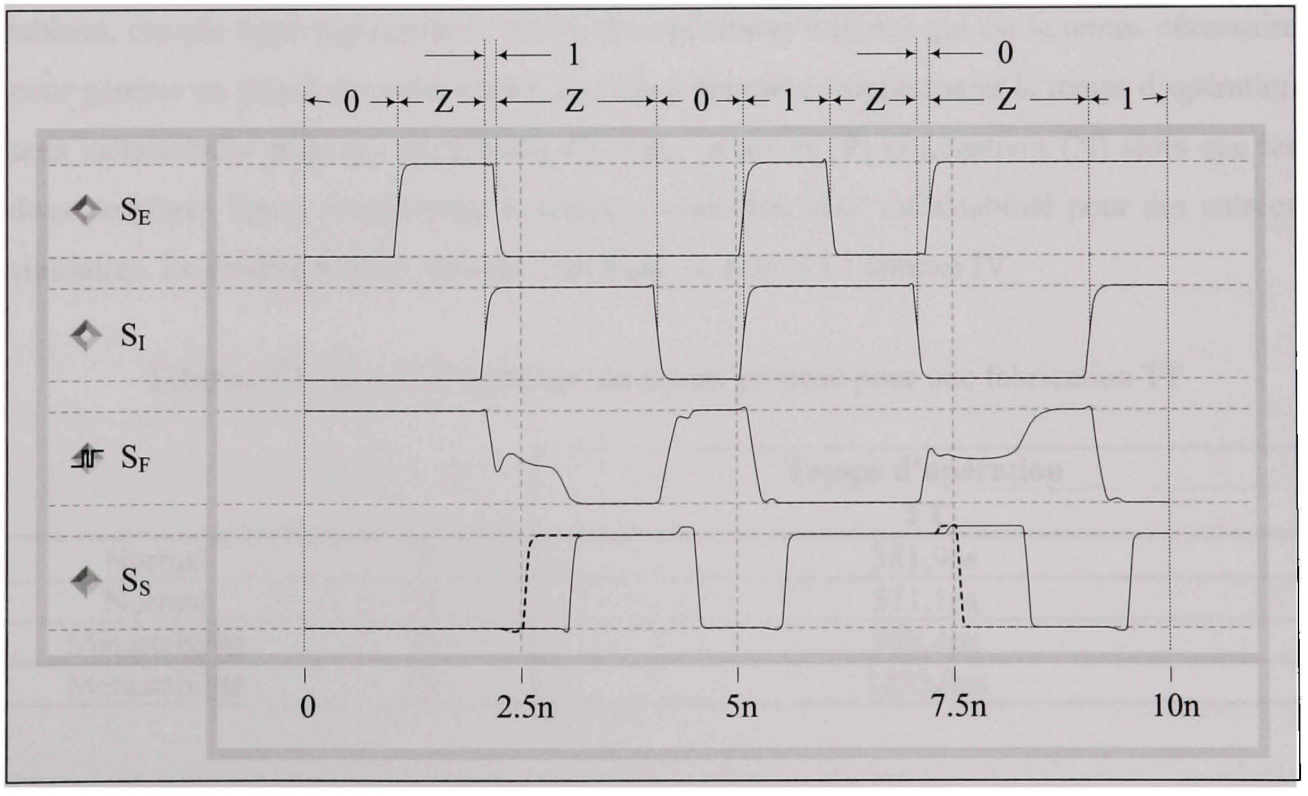


Figure 4.10 Simulation SPICE du circuit proposé de masquage des aléas temporels avec signal SI contrôlé de façon externe.

La simulation de la figure 4.10 illustre cinq courbes : les entrées S_E et S_I, le signal interne d'emballement S_F et deux versions du signal de sortie S_S où le signal S_I n'est pas une version retardée de S_E, mais une entrée contrôlée de façon externe pour les besoins de la simulation. Pour le signal de sortie, la ligne noire représente la sortie obtenue (le pire cas pour des impulsions positives et négatives) et la ligne pointillée représente la sortie si l'emballement avait été instantané. Les résultats montrent que le circuit répond à nos attentes. Un autre résultat intéressant que cette simulation ne permet de visualiser est le comportement du circuit à une impulsion d'entrée de 10fs plus courte que celle utilisée pour cette simulation. Dans ces circonstances, le circuit ne produit pas d'impulsions à la sortie, et ce, sans créer d'artéfacts comme ceux présentés à la figure 4.6. Bref, le seul bémol de ce circuit est le temps d'emballement qui n'est pas instantané affectant du coup la latence.

Le tableau 4.3 illustre les données pertinentes extraites de cette simulation SPICE basée sur des transistors « N » et « P » typiques, d'où l'appellation Typique-Typique (TT). Pour ce tableau, chaque ligne représente le temps des opérations internes qui est le temps nécessaire pour générer un signal de sortie viable. Les deux premières représentent le temps d'opération sans métastabilité pour des impulsions d'entrées positives (P) et négatives (N) alors que les deux dernières lignes représentent le temps d'opération avec métastabilité pour des entrées similaires. Le modèle SPICE, de cette topologie, se trouve à l'annexe IV.

Tableau 4.3 Temps d'opération du circuit proposé pour une fabrication TT

		Temps d'opération
		TT
Normal	P	381,9ps
Normal	N	371,1ps
Métastabilité	P	958,4ps
Métastabilité	N	1455,0ps

En ajustant D_{DI} en fonction de ces données et du temps pour générer une impulsion de sortie minimum, la simulation SPICE de la figure 4.11 confirme que le circuit est fonctionnel pour un signal d'entrée arbitraire.

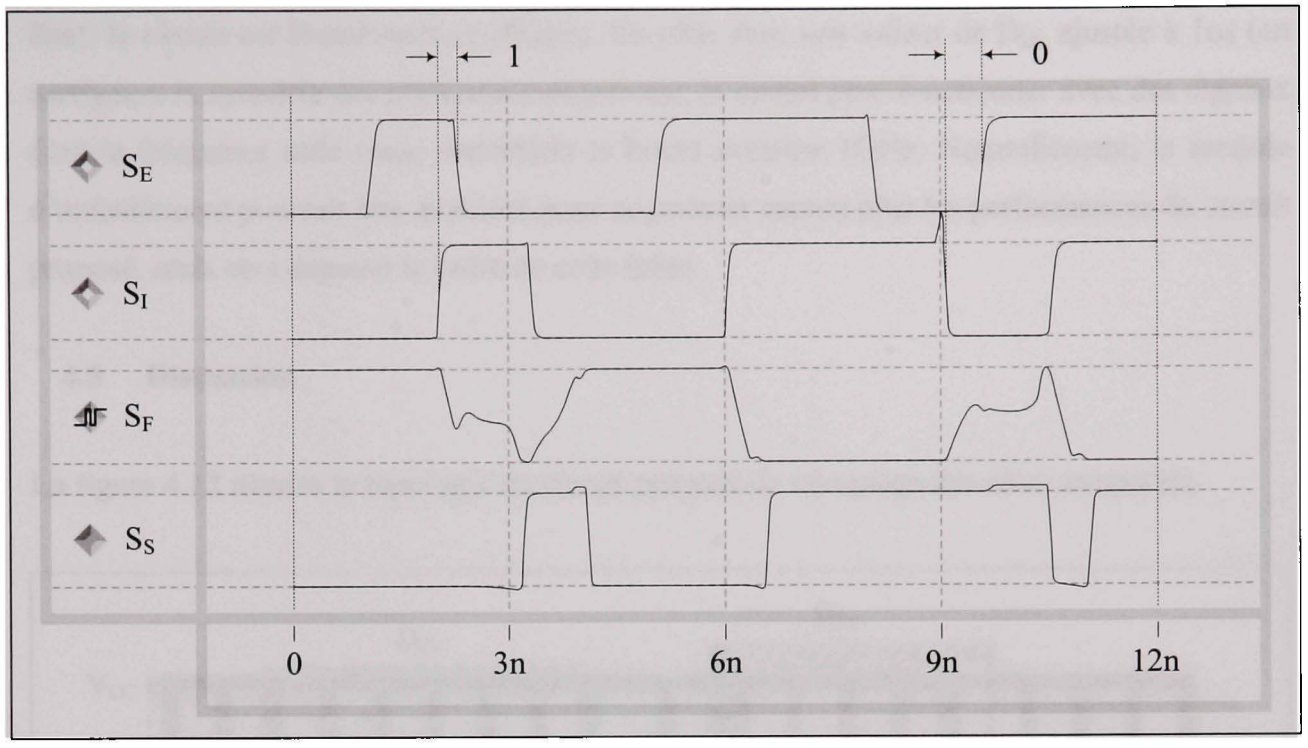


Figure 4.11 Simulation SPICE du circuit proposé de masquage des aléas temporels.

La simulation de la figure 4.11 illustre quatre courbes : Le signal d'entrée S_E et sa version retardée S_I, le signal interne d'emballement S_F et celui de sortie S_S. Le signal S_I est retardé de la valeur D_{DI} qui est ajustée en fonction des données extraites de la figure 4.10. Dans ces circonstances, le signal de sortie réagit selon les attentes. De plus, une impulsion d'entrée de 10fs plus courte que celle utilisée pour cette simulation ne produit pas d'impulsions à la sortie, et ce, toujours sans créer d'artéfacts. Autre fait à noter, l'impulsion positive est de largeur inférieure à celle de l'impulsion négative pour cette simulation, cet effet est dû à la nature des transistors. Ainsi, en fonction des besoins, il est possible d'ajuster la taille des transistors (force des transistors « N » par rapport à la force des transistors « P ») du circuit pour assurer une meilleure symétrie des signaux. De plus, il suffit d'augmenter D_{DI} si l'impulsion de sortie n'a pas une largeur suffisante pour le circuit en aval. Cependant, la simulation montre que le circuit est fonctionnel, et ce, sans aucune optimisation au niveau des transistors, ce qui signifie que le circuit pourrait être réalisé à l'aide de librairie CMOS standard.

Bref, le circuit est fonctionnel et efficace. De plus avec une valeur de D_{D1} ajustée à 1ns (en corrigeant la symétrie des impulsions négatives), le circuit peut fonctionner avec des signaux dont la fréquence utile (sans considérer le bruit) avoisine 1GHz. Naturellement, le module d'emballement pourrait être amélioré pour augmenter encore plus les performances du circuit proposé, mais ceci dépasse le cadre de cette thèse.

4.5 Discussion

La figure 4.12 illustre la topologie du circuit proposé de masquage des aléas temporels.

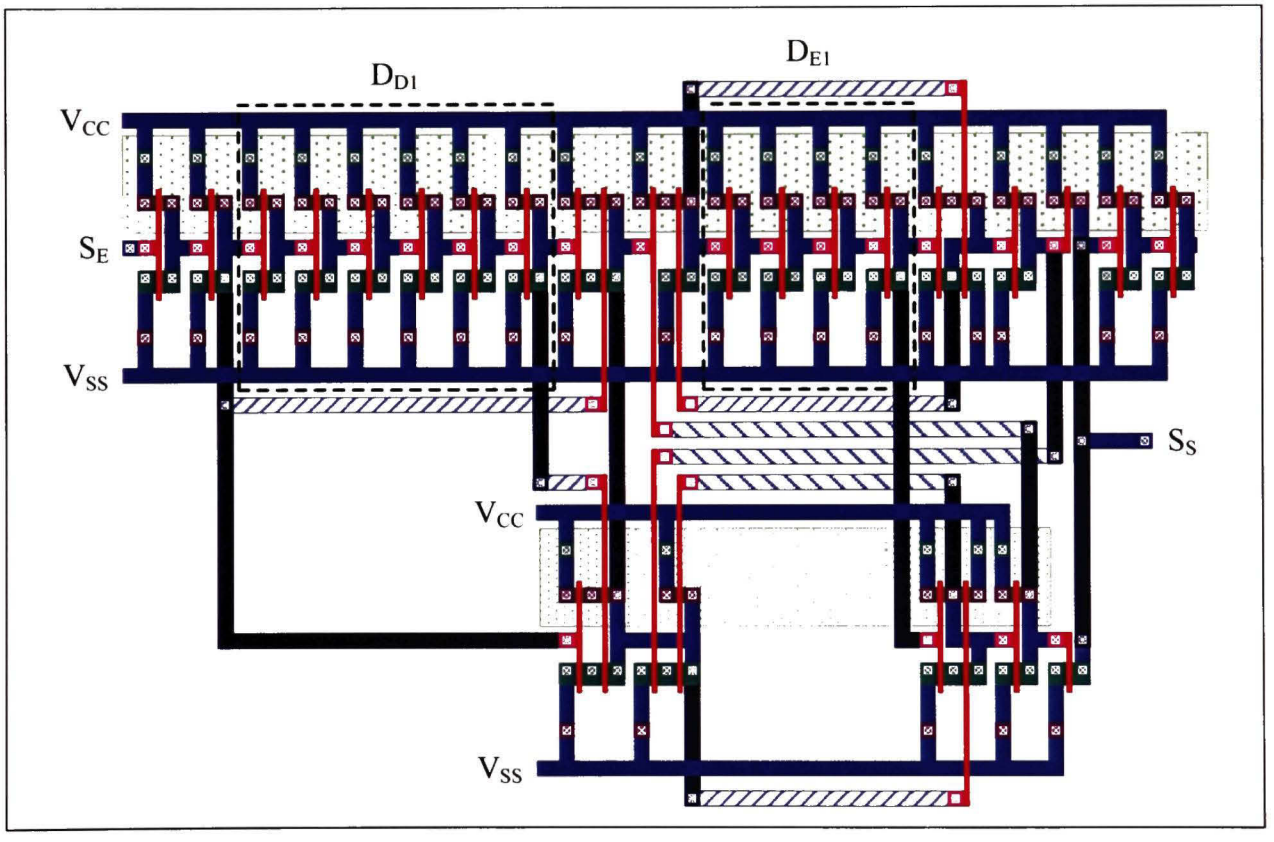


Figure 4.12 Topologie du circuit proposé de masquage des aléas temporels.

La topologie du circuit est illustrée à titre d'exemple de réalisation. Quelques optimisations booléennes ont été faites pour réduire les retards dans le circuit, mais aucun effort particulier n'a été fait pour optimiser la topologie elle-même. Ainsi, une optimisation au chapitre de la vitesse d'opération et de la surface est encore possible. La technologie utilisée pour cette

conception personnalisée est la librairie TSMC 0.18µm avec des modèles SPICE BSIM 3.2. Cette librairie est adéquate pour ces simulations, car, même si une librairie plus récente permettrait d'atteindre de meilleures performances, la réduction à l'échelle ne génère pas de changements d'architecture et l'analyse demeure exacte.

4.5.1 Impacts des variations de fabrication

De la topologie de la figure 4.12, nous avons fait varier les paramètres SPICE pour les différents procédés de fabrication et la température de façon à établir la robustesse du circuit. Le tableau 4.4 illustre les différents résultats pour les variations liées au procédé de fabrication. La table possède quatre colonnes de données, une pour chaque limite considérant que la vitesse des transistors « N » et « P » est : Rapide-Rapide (FF), Lente-Rapide (SF), Rapide-Lente (FS) et Lente-Lente (SS). Le tableau 4.5 illustre, quant à lui, les variations liées à la température. La table possède trois colonnes de données, une pour chaque température considérée : -40°C, 27°C et 85°C. La température médiane correspond à une température normale utilisée par défaut alors que les autres températures correspondent aux écarts standards pour le grade industriel.

Tableau 4.4 Temps d'opération du circuit proposé pour les limites de fabrication

		Limites de fabrication			
		FF	**SF**	**FS**	**SS**
Normal	P	308,2ps	382,9ps	383,9ps	472,5ps
Normal	N	286,2ps	382,3ps	368,1ps	463,2ps
Métastabilité	P	983,9ps	1066,0ps	1051,7ps	1425,0ps
Métastabilité	N	1095,0ps	1253,0ps	1298,0ps	1571,0ps

Tableau 4.5 Temps d'opération en fonction de la température du circuit proposé de fabrication TT

		Température		
		-40°C	**27°C**	**85°C**
Normal	R	347,9ps	381,9ps	417,3ps
Normal	F	327,2ps	371,1ps	419,8ps
Métastabilité	R	932,3ps	958,4ps	998,2ps
Métastabilité	F	1104,0ps	1455,0ps	1544,0ps

Une analyse complète des différentes variations du procédé de fabrication a prouvé la robustesse du circuit proposé. Ledit circuit est robuste et demeura fonctionnel même pour de grandes variations de fabrication pour les raisons suivantes. Nous avons utilisé des éléments de retard partagés au lieu de retards distribués pour rendre le circuit viable devant de grandes variations telles que celles rencontrées pour les cas extrêmes de procédés de fabrication SS et FF où les opérations sont affectées similairement. Nous avons utilisé des tampons doubles avec des caractéristiques comparables de façon à diminuer l'impact des variations propres aux cas extrêmes de fabrication SF et FS. Finalement, les variations locales sur dé ne devraient pas être un problème étant donné que la synchronisation transparente se comporte de la même façon que la logique insensible aux retards qui n'est pas affectée par de telles variations. Si d'aventure un problème était détecté durant la phase de conception, le réglage des retards permettrait d'augmenter la « fréquence de coupure » et de masquer les événements potentiellement problématiques corrigeant ainsi les problèmes associés aux différentes variations attribuables au procédé de fabrication visé.

4.5.2 Estimation de la surface et du nombre de transistors

À partir de la topologie du circuit proposé, la taille peut être extrapolée. Dans ces circonstances, la surface totale est incluse dans un rectangle de 26.0μm x 41.8μm où le circuit n'occupe pas plus de 720.0μm^2 de cet espace pour chaque circuit de masquage des aléas temporels. La topologie est construite à l'aide de 56 transistors : un tampon double à 4 transistors à l'entrée du circuit pour simuler les caractéristiques réelles du signal d'entrée, un autre tampon double à 4 transistors à la sortie pour simuler une charge réelle, 20 transistors

pour implémenter les différents éléments de retard et 28 transistors pour implémenter la fonctionnalité de masquage d'aléas. En fait, le nombre de transistors relatif aux éléments de retard dépend de l'application et de la méthode d'implémentation desdits retards. Finalement, la logique de masquage des aléas temporels pourrait être implémentée avec seulement 24 transistors, mais la topologie présentée possède un double circuit d'isolation pour assurer une meilleure symétrie des signaux. Ainsi, étant donné que le circuit proposé n'a pas de consommation statique de puissance et qu'il est très petit, il pourrait être distribué à l'échelle d'un circuit entier pour un usage interne et sur chacun des ports d'entrée/sortie, et ce, pour un coût inférieur à 1 % du nombre total de transistors. Par exemple, pour un circuit de 1 milliard de transistors, 1 % des transistors représente 17'857 modules de masquage d'aléas temporels.

4.5.3 Limitations

Par construction, le circuit de la figure 4.9 possède trois limites.

Une première limite due au module d'isolation qui se comporte comme un circuit d'érosion. Pour assurer la largeur d'impulsion du signal de sortie, la valeur de D_{DI} doit être supérieure à la somme du temps d'emballement et de la largeur minimum d'impulsion de sortie requise. Ainsi, une impulsion d'entrée inférieure à D_{DI} est complètement érodée (éliminée), et ce, même si cette dernière est plus grande que la valeur de D_{EI} (érosion). Cette limitation doit donc être considérée lors de la conception pour ne pas nuire à la fonctionnalité du circuit.

Une seconde limite est associée à la conception des éléments de retard. Plus les impulsions d'entrées sont lentes par rapport aux opérations internes du circuit de masquage des aléas temporels et plus le nombre de transistors associé aux éléments de retard augmente, lorsque ces derniers sont réalisés à l'aide de chaînes d'inverseurs. Dans l'éventualité où l'implémentation des retards s'avère prohibitive, les protocoles de communication, rejetés précédemment en raison de leurs relatives lenteurs, pourraient être appropriés. Dans les faits, la solution proposée aborde le problème de synchronisation pour des signaux de hautes fréquences. Ainsi, cette solution n'est pas adaptée aux signaux de moyennes et basses fréquences. Le concepteur devra utiliser une solution appropriée à ses besoins.

Finalement, une dernière limite concerne la fréquence du signal d'entrée. Le circuit proposé est conçu pour un signal d'entrée d'une largeur d'impulsion supérieure à D_{DI} avec des aléas séparés par une durée d'au moins D_{DI}. Dans l'éventualité d'un signal d'entrée excédant cette limite (plusieurs aléas séparés de moins de D_{DI} ou pour un signal d'entrée périodique de fréquence plus élevée que $(1/(2 \times D_{DI}))$), le module d'isolation génère un signal intermédiaire d'une durée inférieure à D_{DI} le tout en contradiction avec sa fonction d'isolation. Dans ces conditions d'utilisation, le concepteur doit prévoir un filtre en amont du circuit de masquage en ajustant la fréquence de coupure de ce filtre à la même fréquence que celle du circuit de masquage (ou selon ses besoins) pour assurer le comportement souhaité.

Il convient de noter que plus le circuit d'emballement est amélioré et plus D_{DI} est petit. Dans ces circonstances, les impacts des limitations présentées ci-dessus s'amenuisent.

4.6 Conclusion

Le masquage des aléas temporels est une tâche complexe. Le problème fondamental des solutions proposées dans la littérature est de considérer une fréquence beaucoup plus élevée pour le bruit que pour le signal porteur d'informations utiles. Dans un environnement asynchrone, les signaux arbitraires aux entrées ne respectent pas cette contrainte et mènent à des erreurs en n'éliminant pas les aléas temporels à l'entrée ou en créant d'autres à la sortie, à partir de signaux viables en entrée.

Le circuit proposé possède, pratiquement, les caractéristiques d'un filtre idéal (figure 4.1a) où le réglage du retard D_{EI} fixe la « fréquence de coupure » du circuit. Ce circuit fonctionne de façon déterministe pour tous les signaux d'entrées. Ce circuit peut donc être utilisé dans un environnement asynchrone sans aucun problème. L'érosion et la dilatation successives du signal d'entrée permettent d'éliminer les signaux de période inférieure à la fréquence souhaitée et laisser passer ceux de fréquences supérieures. L'emballement (gain élevé) sert à corriger toutes erreurs que le circuit aurait pu engendrer dans la mesure où la fréquence du signal d'entrée tend vers la fréquence de coupure.

Le circuit proposé implémente physiquement les contraintes de masquage des aléas temporels. Ce circuit masque la métastabilité avant qu'elle ne se produise à sa sortie pour garantir que les circuits suivants n'aient pas de problèmes de synchronisation, et ce, de façon transparente pour le concepteur. En ce sens, le concepteur ajoute ce circuit sur chaque signal de synchronisation et oublie le concept même de la métastabilité. Étant donné que la métastabilité est masquée avant qu'elle ne se produise, le circuit proposé n'engendre pas de dégradations des performances. Il faut toutefois prévoir une latence d'environs d'un temps de préparation pour un élément de mémoire (ajusté à l'aide des retards D_{D1} et D_{E1}). Naturellement, cette latence est directement relative à la technologie utilisée et au type d'application à laquelle elle s'applique.

L'aspect intéressant de ce circuit est l'absence de contraintes sur la mise en forme du signal d'entrée. Le circuit est responsable d'analyser le signal d'entrée et de masquer ses aspects problématiques. Le circuit proposé produit donc un signal de sortie qui ne produira pas de métastabilité dans les circuits en aval dans la mesure où le circuit proposé est construit conformément aux caractéristiques desdits circuits.

Les résultats montrent que le circuit proposé passe de l'état passant à l'état bloquant par un incrément du signal d'entrée inférieure à 10fs. Il s'agit d'une innovation majeure puisqu'il s'agit d'une amélioration de 3 ordres de grandeur par rapport aux circuits de l'état de l'art. Nonobstant ce fait, les caractéristiques d'emballement ont été choisies pour que le circuit s'emballe toujours avant d'amorcer le signal de sortie, confirmant ainsi un masquage complet des aléas temporels.

CONCLUSION

Les conceptions asynchrones reflètent et s'adaptent beaucoup mieux à la réalité des circuits intégrés. En ce sens, la synchronisation sous-jacente aux méthodologies de conception synchrone est une astuce totalement artificielle ayant pour unique but de simplifier le processus de conception. Cette synchronisation, anciennement salvatrice, doit impérativement être remplacée par la logique GALS pour sustenter les besoins toujours croissants de l'industrie. Les défis d'une telle migration résident dans la réalisation de méthodes, de techniques et d'outil GALS favorisant l'exil des concepteurs de cette chasse gardée que représentent les méthodologies synchrones au profit de techniques plus efficaces et performantes tel que la logique GALS.

Conformément à cet objectif, un outil de segmentation synchrone/asynchrone est primordial. En ce sens, notre premier objectif, illustré à la figure 1, était de créer un outil de segmentation capable d'analyser une conception synchrone et de repérer automatiquement les liens de communication ayant avantage à être permutés en liens asynchrones. Cet outil basé sur une fonction de coût (une métrique à plusieurs paramètres) permet d'établir la liste des liens synchrones à remplacer par des liens asynchrones pour obtenir un gain de performance en fonction des paramètres de la métrique. La bonne nouvelle, une version préliminaire de cet outil existe, ce qui valide la faisabilité de l'outil. Cependant, de pair avec la logique GALS, le concept de la métastabilité n'est jamais très loin, il nous est donc apparu essentiel d'aborder ce problème avant toute chose. Par manque de temps, l'outil n'a pas été complété, mais les éléments de conception de cet outil sont présentés à l'annexe V.

En complément à cet outil, cette thèse a présenté les méthodes et techniques sous-jacentes à l'intégration d'un support minimal GALS sur un système de prototypage (FPGA) nativement synchrone. Cet objectif général s'inscrit dans l'effort de migration des systèmes synchrones de très grandes tailles ou de très grandes vitesses, rendus problématiques à cause de la synchronisation globale, vers des systèmes GALS où de petits îlots synchrones sont liés les uns aux autres par des communications asynchrones. Le choix du FPGA s'impose de lui-même dans la mesure où une large diffusion des méthodes et techniques proposées est

souhaitable. De plus, la nature exploratoire de cette recherche a permis de diversifier les objectifs et, du même coup, hétérogénéiser les contributions en adressant plusieurs aspects sur la logique GALS.

La principale contribution entourant l'objectif de création d'une méthodologie d'implémentation de composants GALS dans un FPGA synchrone consiste à la création d'un support viable pour la vérification de circuits GALS. Par cette méthode, il est possible d'utiliser les ressources disponibles des FPGA synchrones pour implémenter des composants asynchrones et des canaux de communication asynchrones. En ce sens, diverses implémentations d'ECM et d'arbitres à deux ports ont été comparées en fonction de leur plage fonctionnelle de façon à cibler la meilleure réalisation dans un contexte donné. Une telle méthode pourrait éventuellement servir à d'autres fins, la vérification de l'intégrité de l'alimentation d'un circuit en fonction de la densité de transitions synchrones n'étant qu'un exemple parmi tant d'autres.

Dans le but d'étoffer la gamme d'applications et les performances générales des systèmes GALS sur circuits programmables, une voie alternative à l'approche d'une méthodologie d'implémentation de composants GALS est celle d'une nouvelle architecture supportant nativement la logique asynchrone. Le compromis choisi, dans ces circonstances, est celui d'un support GALS minimum de façon à ce que le FPGA puisse toujours être considéré comme synchrone, mais qu'il dispose, en plus, de ressources spécialisées pour les communications asynchrones sans être limitées par les ressources synchrones, comme pour l'approche précédente. Dans le but de modifier au minimum l'architecture synchrone à la base de notre architecture, nous avons également contribué à l'élaboration d'un nouveau circuit, implémentant un arbitre à deux ports. Un arbitre basé principalement sur deux ECM et possédant l'avantage de masquer certaines conditions de métastabilité, produisant ainsi un arbitre avec une très faible probabilité de métastabilité. Cette contribution a aiguillé nos recherches subséquentes vers le domaine du masquage de la métastabilité où il est préférable de procéder au masquage en amont plutôt qu'en aval, puisqu'une fois que la métastabilité est créée, la détection de la métastabilité est plus ardue et le temps pour le faire est plus grand.

Cette approche s'est avérée beaucoup plus efficace que la solution traditionnelle consistant à attendre la résolution de la métastabilité.

Ce constat a, par la suite, été généralisé par la création d'un circuit masquant intrinsèquement toutes les conditions de métastabilité. Ce circuit implémente les contraintes nécessaires permettant de masquer la métastabilité avant qu'elle ne se produise. Ce concept est une contribution majeure dans le domaine. À cet effet, les contributions ne sont pas, non plus, terminées étant donné la gamme d'applications pouvant exploiter cette innovation. En ce sens, nous travaillons sur la réalisation d'un ECM dont la sortie n'entrera jamais en métastabilité, et ce, indépendamment de la nature des signaux d'entrées. Bref, un objectif en continuité avec les travaux de cette thèse et avec un fort potentiel commercial.

Bref, à l'origine de mes travaux de recherche sur la logique asynchrone (2002-2003), le concept de logique GALS et de logique asynchrone était très marginalisé. Cependant, au fil du temps, les preuves de concepts se sont accumulées et aujourd'hui plusieurs réalisations sont parfaitement fonctionnelles. À titre d'exemple, j'ai pu participer, en qualité d'étudiant-doctorant, à un des projets de la compagnie OCTASIC pour lequel la logique asynchrone a permis l'atteinte de meilleures performances. En définitive, le futur de la logique GALS est probablement plus rapproché que ce que nous pourrions en penser. Il s'agit évidemment d'un avenir où les méthodes, les techniques et les outils GALS/asynchrones proposés dans cette thèse pourront être utiles.

ANNEXE I

Conditions de simulation SPICE

- Technologie 0.18µm.
- Modèle SPICE BSIM 3.2.
- Conception des transistors complètement personnalisée.
- Extraction des capacités parasites en fonction de la topologie.
- Taux de changement des transitions des entrées limités à 1ps.
- Tampon d'entrées (deux inverseurs en série) pour simuler des entrées réelles.
- Tampons de sorties (deux inverseurs en série) pour simuler une charge de sortie.

ANNEXE II

Méthodologie de simulation SPICE

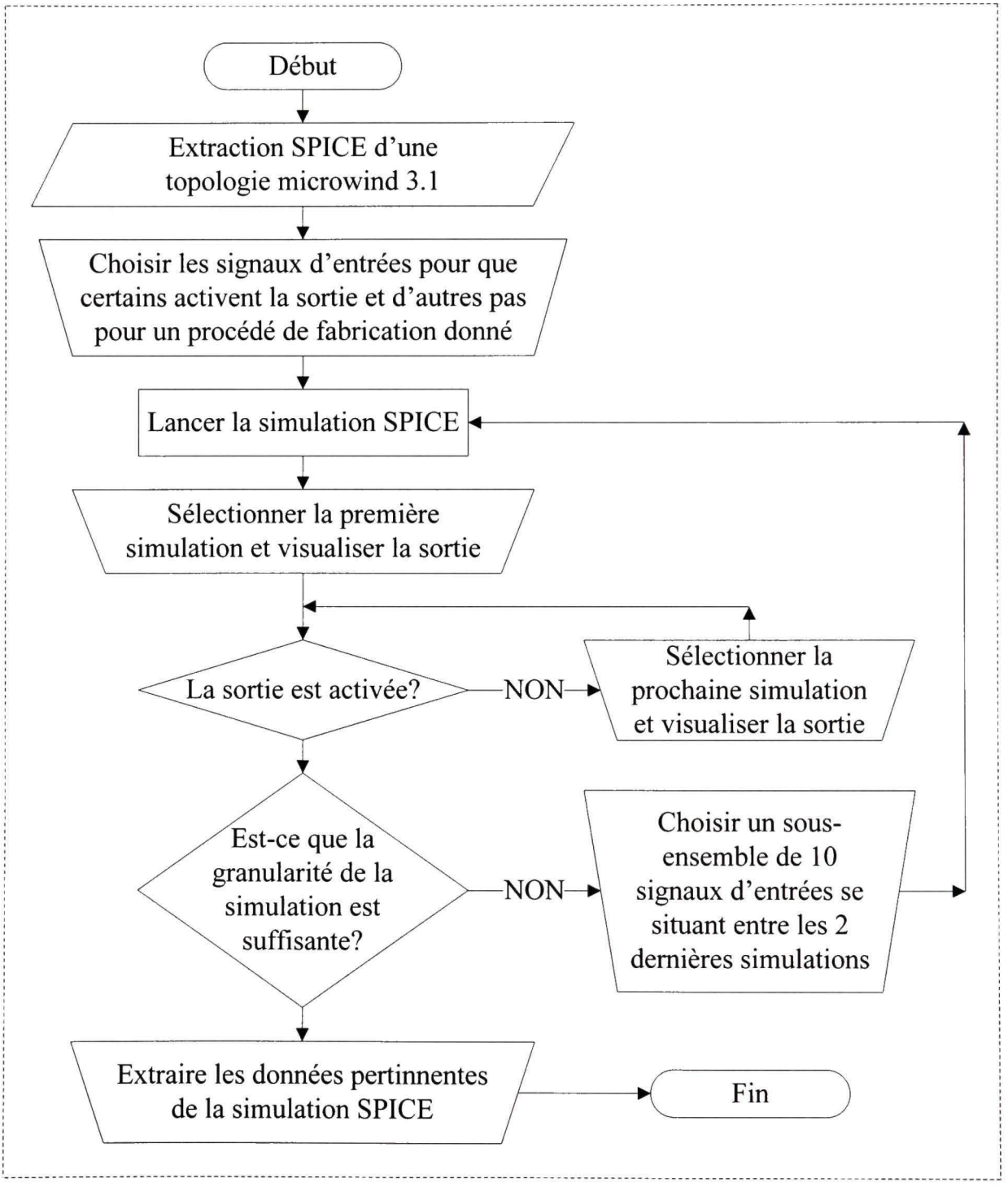


Figure II.1 Méthodologie de simulation SPICE.

ANNEXE III

Résultat de l'extraction SPICE des retards

Le tableau III.1 donne tous les éléments de retard présentés à la figure 3.4 incluant la valeur de $Tecm_{MIN}$ pour chaque ECM. Il est à noter que tous les éléments de retard dans la table sont exprimés en picoseconde (ps). Chaque paramètre est présenté sur deux lignes : une ligne avec le préfixe « M » pour le retard associé à un front montant et une seconde ligne avec le préfixe « D » pour le retard associé à un front descendant. Ces différentes valeurs donnent les vraies valeurs de temps de propagation pour un signal d'entrée montant ou descendant. Cette distinction est nécessaire puisque les valeurs diffèrent légèrement. À titre d'exemple, un front montant passant par D_{BI} produira un front descendant passant par D_{AO}. Finalement, le tableau III.1 possède 5 colonnes des données, une colonne pour chaque procédé de fabrication considéré dont la vitesse des transistors « N » et « P » est : Typique-Typique (TT), Rapide-Rapide (FF), Lent-Lent (LL), Lent-Rapide (SF) et Rapide-Lent (FS).

Tableau III.1 Extraction des retards SPICE pour différentes variations de procédés de fabrication

		TT	FF	SS	SF	FS
D_{BI}	M	183,10	148,70	227,20	193,90	173,60
	D	180,60	148,50	221,20	176,90	185,80
D_{AA}	M	142,30	115,60	176,50	143,70	142,30
	D	140,60	115,20	173,00	146,50	136,20
D_{FO}	M	84,30	68,49	104,70	93,54	76,03
	D	84,52	69,26	103,80	78,24	91,58
D_{AO}	M	96,96	77,31	123,10	102,10	92,60
	D	96,34	77,05	121,50	95,71	97,39
D_{AC}	M	97,74	78,69	122,40	101,80	94,23
	D	97,70	78,93	121,40	98,23	97,68
D_{AI}	M	190,60	154,20	237,00	199,60	182,00
	D	189,60	154,50	234,50	185,90	194,00
D_{BA}	M	136,60	110,40	169,60	135,40	138,90
	D	129,40	106,60	158,20	133,90	125,80
D_{BO}	M	109,00	87,33	137,70	114,00	105,10
	D	106,50	86,48	132,00	108,20	105,40
D_{BC}	M	109,60	88,63	136,30	113,00	106,80
	D	109,60	89,13	135,40	111,30	108,70
D_{D1}	M	74,75	59,29	94,97	84,12	65,94
	D	75,04	61,93	91,09	69,45	80,87
D_{D2}	M	400,60	329,40	491,20	413,90	390,00
	D	400,40	331,10	487,50	402,20	401,10
D_{FA}	M	118,20	96,69	145,30	119,40	117,70
	D	126,30	102,10	157,60	130,70	123,00
$Tecm_{MIN\ A}$	M	40,84	32,63	51,83	47,24	34,96
	D	42,24	34,82	51,72	38,43	46,47
$Tecm_{MIN\ B}$	M	52,79	42,88	65,68	60,09	46,18
	D	54,51	45,05	66,36	50,11	59,38

ANNEXE IV

Modèle SPICE du circuit proposé de masquage d'aléas temporels

```
CIRCUIT GlitchRemover53.MSK dessiné avec uwind 3.1
* IC Technology: CMOS 0.18µm - 6 Metal
* Sélection de la librairie de simulation – Sélection du « ProcessCorner »
.lib '/CMC/kits/cmosp18/models/hspice/mm018.l' TT
*.lib '/CMC/kits/cmosp18/models/hspice/mm018.l' FF
*.lib '/CMC/kits/cmosp18/models/hspice/mm018.l' SS
*.lib '/CMC/kits/cmosp18/models/hspice/mm018.l' SF
*.lib '/CMC/kits/cmosp18/models/hspice/mm018.l' FS
*
VDD 1 0 DC 2.00
VIB 34 0 PWL(       0n 0V 1.999n 0V 2n 2V 3.999n 2V 4n 0V 4.999n 0V 5n 2V
+                   7.999n 2V 8n 0V 8.999n 0V 9n 2V)
*
* Liste des noeud
* "IA"          corresponds to n°34
* "MullerC"     corresponds to n°18
* "s11"         corresponds to n°11
* "s12"         corresponds to n°9
* "Z"           corresponds to n°7
*
* MOS devices – Début de l'extraction SPICE
MN1     0   6   4   0   nch   W= 0.44U   L= 0.18U
MN2     0   7   6   0   nch   W= 0.44U   L= 0.18U
MN3     0  10   7   0   nch   W= 0.44U   L= 0.18U
MN4     0  11   9   0   nch   W= 0.44U   L= 0.18U
MN5     0  13  10   0   nch   W= 0.44U   L= 0.18U
MN6    11  18   0   0   nch   W= 0.44U   L= 0.18U
MN7    30  18  13   0   nch   W= 0.44U   L= 0.18U
MN8     0  14  11   0   nch   W= 0.44U   L= 0.18U
MN9     0  14  30   0   nch   W= 0.44U   L= 0.18U
MN10    0  15  14   0   nch   W= 0.44U   L= 0.18U
MN11    0  16  15   0   nch   W= 0.44U   L= 0.18U
MN12    0  17  16   0   nch   W= 0.44U   L= 0.18U
MN13    0  18  17   0   nch   W= 0.44U   L= 0.18U
MN14    0  11  18   0   nch   W= 0.44U   L= 0.18U
MN15   31  13  18   0   nch   W= 0.44U   L= 0.18U
MN16    0   9  31   0   nch   W= 0.44U   L= 0.18U
MN17   32  28  18   0   nch   W= 0.44U   L= 0.18U
MN18   33  22  18   0   nch   W= 0.44U   L= 0.18U
MN19    0  22  32   0   nch   W= 0.44U   L= 0.18U
```

MN20	0	28	33	0	nch	W= 0.44U	L= 0.18U
MN21	0	23	22	0	nch	W= 0.44U	L= 0.18U
MN22	0	24	23	0	nch	W= 0.44U	L= 0.18U
MN23	0	25	24	0	nch	W= 0.44U	L= 0.72U
MN24	0	26	25	0	nch	W= 0.44U	L= 0.72U
MN25	0	27	26	0	nch	W= 0.44U	L= 0.72U
MN26	0	28	27	0	nch	W= 0.44U	L= 0.72U
MN27	0	29	28	0	nch	W= 0.44U	L= 0.18U
MN28	0	34	29	0	nch	W= 0.44U	L= 0.18U
MP1	1	6	4	1	pch	W= 1.20U	L= 0.18U
MP2	1	7	6	1	pch	W= 1.20U	L= 0.18U
MP3	1	9	7	1	pch	W= 1.20U	L= 0.18U
MP4	1	11	9	1	pch	W= 1.20U	L= 0.18U
MP5	1	13	10	1	pch	W= 1.20U	L= 0.18U
MP6	12	18	11	1	pch	W= 1.20U	L= 0.18U
MP7	13	18	1	1	pch	W= 1.20U	L= 0.18U
MP8	1	14	12	1	pch	W= 1.20U	L= 0.18U
MP9	1	14	13	1	pch	W= 1.20U	L= 0.18U
MP10	1	15	14	1	pch	W= 1.20U	L= 0.18U
MP11	1	16	15	1	pch	W= 1.20U	L= 0.18U
MP12	1	17	16	1	pch	W= 1.20U	L= 0.18U
MP13	1	18	17	1	pch	W= 1.20U	L= 0.18U
MP14	19	11	18	1	pch	W= 1.20U	L= 0.18U
MP15	1	13	18	1	pch	W= 1.20U	L= 0.18U
MP16	1	10	19	1	pch	W= 1.20U	L= 0.18U
MP17	20	28	18	1	pch	W= 1.20U	L= 0.18U
MP18	21	22	18	1	pch	W= 1.20U	L= 0.18U
MP19	1	22	20	1	pch	W= 1.20U	L= 0.18U
MP20	1	28	21	1	pch	W= 1.20U	L= 0.18U
MP21	1	23	22	1	pch	W= 1.20U	L= 0.18U
MP22	1	24	23	1	pch	W= 1.20U	L= 0.18U
MP23	1	25	24	1	pch	W= 1.20U	L= 0.72U
MP24	1	26	25	1	pch	W= 1.20U	L= 0.72U
MP25	1	27	26	1	pch	W= 1.20U	L= 0.72U
MP26	1	28	27	1	pch	W= 1.20U	L= 0.72U
MP27	1	29	28	1	pch	W= 1.20U	L= 0.18U
MP28	1	34	29	1	pch	W= 1.20U	L= 0.18U

* Capacités extraites par le logiciel de capture (uwind 3.1)

C2	1	0	23.039fF
C3	1	0	56.065fF
C4	4	0	1.151fF

```
C6          6     0          1.340fF
C7          7     0          2.590fF
C9          9     0          3.478fF
C10         10    0          3.294fF
C11         11    0          3.044fF
C12         12    0          0.395fF
C13         13    0          3.000fF
C14         14    0          2.667fF
C15         15    0          1.340fF
C16         16    0          1.340fF
C17         17    0          1.340fF
C18         18    0          9.440fF
C19         19    0          0.395fF
C20         20    0          0.395fF
C21         21    0          0.395fF
C22         22    0          2.533fF
C23         23    0          1.340fF
C24         24    0          1.340fF
C25         25    0          1.340fF
C26         26    0          1.340fF
C27         27    0          1.340fF
C28         28    0          4.084fF
C29         29    0          1.342fF
C30         30    0          0.563fF
C31         31    0          0.563fF
C32         32    0          0.563fF
C33         33    0          0.563fF
C34         34    0          0.210fF
C35         35    0          0.079fF
* Fin de l'extraction
*
*.OPTION POST              →    Génère les fichiers externes
*.OPTION POST PROBE        →    Inclue uniquement les probes dans les fichiers externes
*
* Sélectionnez la bonne température pour la simulation de robustesse en température.
.TEMP 27.0
*.TEMP -40
*.TEMP 85

.OPTION POST=BINARY PROBE MEASOUT
.TRAN 0.5p 20.00N
.PROBE V(34)        V(28)        V(22)
.PROBE V(18)
.PROBE V(17)        V(16)        V(15)        V(14)
.PROBE V(11)        V(13)
```

```
.PROBE V(9)          V(10)
.PROBE V(7)          V(6)          V(4)
*
* Entrées pour simulation selon la méthodologie de l'ANNEXE II.
* C'est cette méthodologie qui est utilisée pour trouver le point de métastabilité.
*
*.ALTER 'VA 2.049'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.049n 2V 2.050n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.049n 0V 7.050n 2V)
*.ALTER 'VA 2.099'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.099n 2V 2.100n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.099n 0V 7.100n 2V)
*.ALTER 'VA 2.149'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.149n 2V 2.150n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.149n 0V 7.150n 2V)
*.ALTER 'VA 2.199'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.199n 2V 2.200n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.199n 0V 7.200n 2V)
*.ALTER 'VA 2.249'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.249n 2V 2.250n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.249n 0V 7.250n 2V)
*.ALTER 'VA 2.299'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.299n 2V 2.300n 0V 4.999n 0V 5n 2V
+                    5.999n 2V 6n 0V 7.299n 0V 7.300n 2V)
*.ALTER 'VA 2.349'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.349n 2V 2.350n 0V 4.999n 0V 5n 2V
                     5.999n 2V 6n 0V 7.349n 0V 7.350n 2V)
*.ALTER 'VA 2.399'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.399n 2V 2.400n 0V 4.999n 0V 5n 2V
                     5.999n 2V 6n 0V 7.399n 0V 7.400n 2V)
*.ALTER 'VA 2.449'
*VIA 34 0 PWL(       0n 0V 0.999n 0V 1n 2V 2.449n 2V 2.450n 0V 4.999n 0V 5n 2V
                     5.999n 2V 6n 0V 7.449n 0V 7.450n 2V)

.END
```

ANNEXE V

Outil automatisé de segmentation asynchrone sur FPGA synchrone

À l'origine, la thèse a été amorcée tel qu'illustré à la figure 1. C'est-à-dire en ciblant la conception d'un outil de segmentation synchrone/asynchrone. Cependant, lors de l'examen approfondi de la connectivité pour le remplacement des liens synchrones en liens asynchrones (après l'examen doctoral), la notion de synchronisation a été soulevée et a fait diverger la recherche sur la création de composants robuste à la métastabilité avant de considérer la segmentation elle-même, tout en gardant à l'esprit l'aspect performance.

Dans ces circonstances, les liens de communications synchrones limitent les performances globales d'une conception synchrone (Chapiro, 1985; Friedman, 2001). La technique d'amélioration ciblée consiste à remplacer ces liens de communications synchrones par des liens asynchrones (Chakraborty et Greenstreet, 2003), à la manière d'un système GALS (Chapiro, 1985; ITRS, 2007; Sutherland et Ebergen, 2002). En d'autres mots, ce problème nécessite l'élaboration d'outils de segmentation synchrone/asynchrone complémentant les méthodes et techniques présentées dans cette thèse.

Cette tâche de segmentation synchrone/asynchrone, apparemment simple, cache un problème NP complet puisque les décisions se prennent sur un système non déterministe en un temps polynomial. Malgré cette complexité insoluble, ce problème peut être simplifié par une méthode d'optimisation basée sur une fonction de coût empirique à plusieurs paramètres. L'outil est en cours d'élaboration et laisse entrevoir la faisabilité et la viabilité d'un tel outil pour la segmentation synchrone/asynchrone.

Choix de la métrique

La sélection de la métrique est l'étape de base de la formulation d'une règle de segmentation déterminant la limite entre les logiques synchrones et asynchrones. Dans l'optique d'établir une preuve de concept pour cette recherche doctorale, une métrique dédiée est nettement

suffisante. De plus, le langage de description matériel VHDL (Vachoux, 2002) servira d'entrée aux outils de l'environnement de logiciels intégrés (ISE) pour cibler des circuits de logique programmable de la compagnie Xilinx. Chaque module source est représenté par un modèle simplifié ou la logique interne est remplacée par des paramètres représentatifs tenant compte de la hiérarchie.

Parmi les paramètres clés, notons le nom des entités, la surface et le nombre de cellules de base que chaque module contient. Du côté des entrées, seuls les noms et les points spécifiques de connexion aux modules sont importants. Alors que pour les sorties, il faut ajouter à cela le temps de génération de la sortie et la liste des dépendances aux entrées. Finalement, les liens entre les modules sont considérés unidirectionnels et uniques, ainsi une connexion bidirectionnelle ou en fourche génère deux liens de communication avec des paramètres identiques, mais avec des sources et destinations potentiellement différentes. Donc en plus des noms de signaux, les paramètres considérés pour les liens de communication sont la fréquence et le retard de routage associé. Évidemment, les signaux de sortie peuvent être regroupés, mais dans ce cas, il faut conserver l'information du nombre de signaux regroupés.

Un tel modèle permet de faire une représentation fidèle d'une conception. Néanmoins, un niveau de détails aussi important n'est pas nécessaire, une simple revue de la littérature permet de voir quels sont les endroits de segmentation populaire (Chakraborty et Greenstreet, 2003; Chapiro, 1985; Hemani et al., 1999; Matzke, 1997; Svantesson, Kumar et Hemani, 1998) :

- les changements de domaines d'horloge naturels (communications globales);
- les ports de communication d'entrées;
- les frontières naturelles entre les modules;
- la taille des modules;
- les modules de haute fréquence;
- les modules de faible connectivité et longueur des fils de communication.

Naturellement, deux modules fonctionnant à des fréquences différentes sont propices à des communications asynchrones. Les ports d'entrées et la modularité intrinsèque d'une conception représentent également des endroits propices à la segmentation sur les liens de communication entre les modules. S'il s'avérait qu'aucune frontière naturelle n'était propice à la segmentation, les informations reliées à la taille, à la fréquence d'opération et à la hiérarchie permettraient de créer des frontières artificielles. Par exemple, un module trop volumineux pour sa fréquence d'opération donnée doit être segmenté, en se basant sur sa hiérarchie, où chaque toute la surface de chaque module synchrone est atteinte à l'intérieur d'une même période d'horloge. Inversement, des modules trop petits auront un coût de communication très élevé. Finalement, le faible niveau de connectivité est également un endroit de prédilection pour la segmentation, puisque ces endroits requirent moins de changement pour les mêmes résultats.

Compte tenu des difficultés d'extraction, il devient alors évident que certains paramètres, comme le cycle actif des signaux, ne seront pas extraits. Les sorties sont considérées comme synchrones, les conceptions purement combinatoires ne sont donc pas considérées. Les références géométriques, intéressantes pour le placement, ne seront pas, non plus, considérées à cause des difficultés d'extraction et de la notion du cas par cas qu'elles engendrent. Finalement, les retards liés au routage ne pourront qu'être estimés, pour les mêmes raisons que celles invoquées pour l'extraction des références géométriques.

Ces approximations supposent inévitablement une certaine marge erreurs. Cependant, l'idée n'est pas de créer une méthode infaillible, mais d'établir une preuve de concept. Ainsi, le lecteur intéressé saura vers quoi orienter ses efforts pour améliorer les performances de la méthode. Une première version de métrique peut donc être aussi simple qu'un coût non nul s'il y a changement de domaine d'horloge, une connexion unique ou une taille de module plus grande qu'un certain seuil.

Le coût total non nul d'un lien de communication synchrone signifie que le lien est un bon candidat à être transformé en lien asynchrone. Évidemment, cette métrique très simple ne tient pas compte de tous les éléments contenus dans un modèle complet, mais ces derniers

pourront être utilisés, si plusieurs liens sont sujets à la désynchronisation. Dans ce cas, une pondération pourra être appliquée aux différents éléments de façon à établir lesquels, des liens, doivent être changés en priorité. À titre d'exemple, une pondération de 75 % sur un changement de domaine d'horloge donnerait beaucoup d'importance à ce paramètre par rapport aux autres. Il est à noter que la métrique ne tient pas compte du coût de transformation d'un lien synchrone en lien asynchrone, car il s'agit d'une étude exploratoire sur les outils de segmentation.

Finalement, le type de conception est, lui aussi, important. Le type de conception linéaire avec une forte proportion de circuits en pipeline n'est pas représentatif des circuits couramment rencontrés. Cependant, ce type de lien est suffisamment pertinent pour se retrouver dans une conception complexe telle qu'une unité centrale de traitement ou autres. Bref, la méthode développée ne cible pas un type de circuit particulier, mais s'adapte à ses particularités.

Extraction des paramètres

L'extraction de ces paramètres à partir des fichiers sources est faite à l'aide d'outils similaires à Lex (Lesk et Schmidt) et Yacc (Johnson), soit les programmes Bison (Donnelly et Stallman, 1995) et Flex (Paxson, 1995), sous la licence GNU GPL (Stallman, 1983). Il s'agit d'outils de génération de code en C. L'utilisation combinée de Flex et de Bison permet de créer une bibliothèque lexicale particulière. Le lexique est défini à l'aide de Flex alors que Bison permet de définir le champ lexical et les tâches à accomplir pour chacune des expressions reconnues. L'outil émergeant de ce processus de définition d'un nouveau dictionnaire est un analyseur syntaxique. Même si les logiciels Flex et Bison sont peu conviviaux et relativement complexes, ils sont très souples et permettent d'obtenir des analyseurs syntaxiques complets, et ce, sans limitation commerciale.

Dans les faits, seuls des analyseurs syntaxiques sont nécessaires pour procéder à l'extraction des paramètres. Cependant, pour gagner du temps de développement, plusieurs résultats intermédiaires peuvent être obtenus par d'autres logiciels non libres (la synthèse et la

simulation). Le logiciel « ISE Project navigator » (Xilinx, 2003), de la compagnie Xilinx est un outil de conception pour FPGA générant des rapports intermédiaires (résultats). Ce logiciel regroupe plusieurs fonctionnalités, mais il est à noter que nous avons utilisé la version 5 qui contient tous les programmes nécessaires à la méthode présentée. Les versions suivantes intègrent d'autres outils, ce qui suppose que certains résultats devraient être trouvés de façon différente, mais n'invalide en rien les travaux en cours. De la même façon, même si les résultats sont générés à l'aide d'outils pour FPGA, les résultats de cette recherche s'appliqueront aussi bien aux FPGA qu'aux ASIC. Le tableau suivant illustre les différents outils nécessaires à l'extraction de différents paramètres.

Tableau V.1 Outils d'extraction des différents paramètres

	Extraction	**Project Navigator**	**Résultat intermédiaire**	**Analyseur syntaxique**
1	Hiérarchie	JHDParse.exe	Rapport JHD	scanJHD →SJ
2	Port	XST.exe	Syntaxe VHDL	scanPort →SP
3	Taille	XST.exe	Rapport SYR	scanCell →SC
4	Horloge	XST.exe	Syntaxe VHDL	scanGCLK →SG
5	Connectivité	Ngc2edif.exe	EDIF 2.0.0	scanEdif →SE

En fonction des objectifs d'extraction, un logiciel d'ISE permettant d'obtenir des résultats intermédiaires pertinents peut être déterminé. L'outil JHD donne le nom du module et des sous-modules du fichier source analysé, il servira à recréer la hiérarchie. XST est l'outil de synthèse de Xilinx. Cet outil est utilisé pour vérifier la syntaxe du code source VHDL (extraction des ports de communication), fournir le rapport de synthèse (extraction de la taille des modules) et fournir la base de données de la connectivité (permettant la génération du fichier de connectivité EDIF). Le format EDIF fut privilégié aux autres formats disponibles (prolog, VHDL, etc.) pour représenter la connectivité, car il existe une bibliothèque JAVA développée par BYUCC qui est libre et qui supporte le format EDIF 2.0.0 (Brigham Young University, 2006).

Les étapes précédentes simplifient l'analyse des données, la tâche résultante est d'extraire l'information contenue de ces divers rapports ou fichiers de synthèse par des analyseurs

syntaxiques créés à l'aide des logiciels Flex et Bison. Pour regrouper plus d'un analyseur dans un même logiciel, chaque analyseur syntaxique doit utiliser des noms de variables uniques. Par exemple, les variables propres au logiciel scanPort utilisent le préfixe « SP », et ce, aussi bien pour le nom des structures que les variables locales et globales. Cette nomenclature est nécessaire, car la communication entre les logiciels créés avec le duo Flex/Bison se fait par des variables globales. Par conséquent, chaque nom de variable doit être unique pour former une seule et même application.

Malgré cette complexité, il existe une macro dans Bison qui permet de tolérer les erreurs, ce qui simplifie le développement des analyseurs syntaxiques. Normalement, un analyseur syntaxique doit traiter le fichier dans son ensemble, sans erreur, sinon il arrêtera sa tâche et retournera une erreur. Cependant, en ajoutant cette macro à toutes les expressions qui ne sont pas nécessaires à l'extraction des paramètres, il est possible de concentrer les efforts d'analyse sur une partie du fichier. Cette macro s'écrit de la façon suivante :

- expression : error { yyclearin; }

Pour une expression donnée, si aucune corrélation n'est faite avec une des différentes définitions de l'expression, la macro yyclearin permet d'éliminer les erreurs (error) latentes dans la base de données. Ainsi, de cette façon, les expressions non reconnues ne font pas échouer l'analyse lorsqu'une analyse exhaustive n'est pas requise.

Application globale

L'application globale consiste à extraire des données à partir des différents analyseurs syntaxiques et à la conservation de ces informations dans une base de données adaptée. Aux fins de l'application, nous avons fait une base de données en C/C++ afin de faciliter l'interfaçage avec les analyseurs syntaxiques. L'application sert donc à ordonnancer les différentes tâches suivantes :

- création de la liste des fichiers VHDL;

- création des fichiers JHD;
- création de la liste des fichiers JHD;
- création d'un seul fichier contenant tous les fichiers JHD;
- extraction de la hiérarchie dans un arbre NR;
- extraction des ports VHDL pour chaque entité;
- création d'une table de correspondance entre les fichiers et les entités;
- synthèse des fichiers VHDL par XST;
- extraction des informations de taille des fichiers SYR;
- extraction des ports VHDL qui sont des ports d'horloge;
- extraction de la connectivité des ports;
- identification des liens de communication synchrones à changer en liens asynchrones.

Les quatre premières étapes de création et de la synthèse du projet sont conditionnelles à la date de modifications des fichiers sources. Une date postérieure à celle des fichiers générés enclenche un nouveau processus de création. Les autres étapes consistent à l'extraction des paramètres. Les travaux ne sont pas terminés, il reste à terminer l'analyse de la connectivité et l'identification du lien à remplacer. Ainsi, nous n'avons pas de résultats, cependant, nous jugions essentiel de présenter les travaux engagés concernant les outils de conception qui peuvent être relativement simples et qui pourront faire l'objet de travaux futurs.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Alfke, Peter. 2002. « Metastability Delay and Mean Time Between Failure in Virtex-II Pro FFs ». *TechXclusives, Xilinx*, (10/28/2002).

Ali, Shahid, et Shivraj G. Dharne. 2005. *Glitch Removal circuit, Patent 6,894,540.* 11 p.

Aspacio, Reuben A., Rajesh Bharadwaj et Hieu Xuan Nguyen. 2001. *Enhencad glitch removal circuitry, Patent 6,249,141.* 8 p.

Bainbridge, W.J., W.B. Toms, D.A. Edwards et S.B. Furber. 2003. « Delay-insensitive, point-to-point interconnect using m-of-n codes ». In *Asynchronous Circuits and Systems, Ninth International Symposium on.* p. 132-140.

Bandapati, S.K., S.C. Smith et M. Choi. 2003. « Design and characterization of convention self-timed multipliers ». *Design & Test of Computers, IEEE,* vol. 20, n° 6, p. 26-36.

Boyer, F.-R., H.G. Epassa, B. Pontikakis, Y. Savaria et Wei Ling. 2004. « A variable period clock synthesis (VPCS) architecture for next-generation power-aware SoC applications ». In *Circuits and Systems, The 2nd Annual IEEE Northeast Workshop on.* p. 145-148.

Breuninger, Robert K. 1990. *Metastable resistant Flip-Flop, Patent 4,929,850.* 9 p.

Brigham Young University. 2006. « BYU EDIF Tools Home Page ». <http://reliability.ee.byu.edu/edif/edifDownload.html>.

Chakraborty, A., et M.R. Greenstreet. 2003. « Efficient self-timed interfaces for crossing clock domains ». In *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on.* p. 78-88.

Chakraborty, S., J. Mekie et D. K. Sharma. 2003. « Reasoning about synchronization techniques in GALS systems: A unified approach ». *in Proc. Workshop Formal Methods GALS Arch. (FMGALS)*, (invited paper,).

Chapiro, Daniel Marcos. 1985. « Globally-Asynchronous Locally-Synchronous Systems (Performance, Reliability, Digital) ». PhD, Stanford University, 136 p.

Chattopadhyay, A., et Z. Zilic. 2005. « GALDS: a complete framework for designing multiclock ASICs and SoCs ». *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 13, n° 6, p. 641-654.

Chen, Deming, et Martin D.F. Wong. 2008. « Latest advances and future opportunities on CAD for FPGAs ». In *13th ASPDAC.* Republic of Korea.

Chen, Wai-Kai. 2003. *The circuits and filters handbook*, 2nd. Coll. « Electrical engineering handbook series ». Boca Raton, Fla.: CRC Press. <http://www.engnetbase.com/ >.

Chipworks. 2008. *Xilinx XC5VLX50 FPGA UMC 65 nm Process*. [Online]. Available: www.chipworks.com, 1-3 p.

Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A. . 2002. *Logic Synthesis of Asynchronous Controllers and Interfaces*. 274 p.

Dally, W. J., et C. L. Seitz. 1987. « Deadlock-Free Message Routing in Multiprocessor Interconnection Networks ». *Computers, IEEE Transactions on,* vol. C-36, n° 5, p. 547-553.

Donnelly, Charles, et Richard Stallman. 1995. « Bison, The YACC-compatible Parser Generator ». <http://dinosaur.compilertools.net/bison/index.html>.

Educause, et CASC. 2009. « Developing a Coherent Cyberinfrastructure from Local Campus to National Facilities: Challenges and Strategies ».

Erstad, David Owen. 2002. *Glitch removal circuitry, Patent 6,356,101*. 14 p.

Fant, Karl M., et Scott A. Brandt. 1996. « NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis ». In *Application Specific Systems, Architectures and Processors, International Conference on* (August 1996). p. 261-273. St. Paul, MN.

Felicijan, T., J. Bainbridge et S. Furber. 2003. « An asynchronous low latency arbiter for Quality of Service (QoS) applications ». In *Microelectronics, 2003, ICM 2003, Proceedings of the 15th International Conference on*. p. 123-126.

Friedman, E.G. 2001. « Clock distribution networks in synchronous digital integrated circuits ». *Proceedings of the IEEE,* vol. 89, n° 5, p. 665-692.

Furber, S.B. 1993. « Breaking step: the return of asynchronous logic ». *IEE Review,* vol. 39, n° 4, p. 159-162.

Furber, S.B. 1996. « Breaking step-the return of asynchronous logic ». In *Design and Test of Asynchronous Systems, IEE Colloquium on*. p. 1/1-1/4.

Gagné, R., J. Belzile et C. Thibeault. 2007. « Architecture for efficient GALS support in commercial FPGAs ». In *Circuits and Systems, 2007. NEWCAS 2007. IEEE Northeast Workshop on*. p. 638-641.

Gagné, R., J. Belzile et C. Thibeault. 2008. « Jadis synchrone, désormais GALS : l'hétérogénéité des architectures de FPGA ». In *FETCH-08* (7-9 janvier 2009). Montebello, Canada.

Gagné, R., J. Belzile et C. Thibeault. 2009a. « Architecture nativement GALS pour FPGA ». In *77e congrès de l'ACFAS* (11-15 mai 2009). Ottawa, Canada.

Gagné, R., J. Belzile et C. Thibeault. 2009b. « Asynchronous Component Implementation Methodology for GALS Design in FPGAs ». In *NEWCAS-TAISA-2009* (june 28 - july 1). p. 1-4. Toulouse, France.

Gagné, R., J. Belzile et C. Thibeault. 2009c. « Composants GALS implémentés sur FPGA synchrone ». In *77e congrès de l'ACFAS* (11-15 mai 2009). Ottawa, Canada.

Gagné, René, Jean Belzile et Claude Thibeault. 2009d. « Émulation de composants asynchrones sur FPGA pour conception GALS ». In *FETCH-09* (12-14 janvier 2009). Chexbres, Suisse.

Gagné, René, Jean Belzile et Claude Thibeault. 2009e. « From synchronous to GALS: A new architecture for FPGAs ». *Microelectronics Journal,* vol. In Press, Corrected Proof.

Gargour, Christian S., Venkat Ramachandran et D. Bensoussan. 1993. *Théorie et conception des filtres analogiques*. Sainte-Foy: Presses de l'Université du Québec, xx, 652 p.

Ginosar, Ran. 2003. « Fourteen Ways to Fool Your Synchronizer ». In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*. IEEE Computer Society.

Global semiconductor forum, . 2003. « FPGAs to be first 90nm ICs ».

Hassoun, S., Yong-Bin Kim et F. Lombardi. 2003. « Guest editors' introduction: clockless VLSI systems ». *Design & Test of Computers, IEEE,* vol. 20, n° 6, p. 5-8.

Hauck, S. 1995. « Asynchronous design methodologies: an overview ». *Proceedings of the IEEE,* vol. 83, n° 1, p. 69-93.

Hauck, S., S. Burns, G. Borriello et C. Ebeling. 1994. « An FPGA for implementing asynchronous circuits ». *Design & Test of Computers, IEEE,* vol. 11, n° 3, p. 60.

Hazewindus, Pieter Johannes. 1992. « Testing Delay-Insensitive Circuits ». Technical Report, California Institute of Technology, 156 p. <http://resolver.caltech.edu/CaltechCSTR:1992.cs-tr-92-14>.

Hemani, A., T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee et D. Lundqvist. 1999. « Lowering power consumption in clock by using globally

asynchronous locally synchronous design style ». In *Design Automation Conference, 1999. Proceedings. 36th.* p. 873-878.

Huffman, D. A. 1964. « The synthesis of sequential switching circuits ». In *Sequential Machines*. p. 3.

ITRS. 2007. « Home 2007 ». [Online]. Available: http://www.itrs.net.

Johnson, Howard, et Martin Graham. 1993. *High Speed Digital Design: A Handbook of Black Magic*, First. Prentice Hall, 384 p.

Johnson, Stephen C. « Yacc: Yet Another Compiler-Compiler ». <http://dinosaur.compilertools.net/yacc/index.html>.

Keech, Eugene E. 1991. *Matastable-proof flip-flop, Patent 4,999,528.* 18 p.

Kinniment, D. J. 2007. « Synchronization and arbitration in digital systems ». In, sous la dir. de Hoboken, NJ J. Wiley & Sons p. 280.

LaPerdus, Marc. June 11, 2007. « IBM aims to revive ASIC with next-gen spin ». *EE Times (www.eetimes.com).*

Laplante, Phillip A. 1997. *Real-Time Systems Design and Analysis: An engineer's handbook*, 2nd., 361 p.

Leach, Jerald G. 1993. *Glitch reduction in integrated circuits, systems and methods, Patent 5,184,032.* 16 p.

Lesk, M. E., et E. Schmidt. « Lex - A Lexical Analyzer Generator ». <http://dinosaur.compilertools.net/lex/index.html>.

Li, N., et B. Nowrouzian. 2002. « A recursive approach to the design of linear-phase half-band digital filters having very sharp transition bands ». In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on.* Vol. 2, p. 242.

Ligthart, M., K. Fant, R. Smith, A. Taubin et A. Kondratyev. 2000. « Asynchronous design using commercial HDL synthesis tools ». In *Advanced Research in Asynchronous Circuits and Systems, Sixth International Symposium on.* p. 114-125.

Liljeberg, P., J. Plosila et J. Isoaho. 2003. « Self-timed ring architecture for SOC applications ». In *SOC Conference, IEEE International [Systems-on-Chip].* p. 359-362.

Lines, A. 2004. « Asynchronous interconnect for synchronous SoC design ». *Micro, IEEE,* vol. 24, n° 1, p. 32-41.

Männer, Reinhard. 1988. « Metastable states in asynchronous digital systems: avoidable or unavoidable? ». *Microelectronics and reliability,* vol. 28, p. 295-307.

Martin, A.J., M. Nystrom et C.G. Wong. 2003. « Three generations of asynchronous microprocessors ». *Design & Test of Computers, IEEE,* vol. 20, n° 6, p. 9-17.

Martin, Alain J. 1990. « The limitations to delay-insensitive in asynchronous circuits ». In, sous la dir. de Dally, William J. p. 263-278. MIT Press.

Martin, Alain J. 1993. « Synthesis of asynchronous VLSI circuits ». In, sous la dir. de Addison-Weslay. p. 1-64.

Masteller, S., et L. Sorenson. 2003. « Cycle decomposition in NCL ». *Design & Test of Computers, IEEE,* vol. 20, n° 6, p. 38-43.

Matzke, D. 1997. « Will physical scalability sabotage performance gains? ». *Computer,* vol. 30, n° 9, p. 37-39.

Mead, Carver, et Lynn Conway. 1980. *Introduction to VLSI systems*. Addison-Wesley, 396 p.

Moore, Gordon E. 1965. « Cramming more components onto integrated circuits ». vol. 38, (April 19, 1965).

Moore, S., G. Taylor, R. Mullins et P. Robinson. 2002. « Point to point GALS interconnect ». In *Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on*, sous la dir. de Taylor, G. p. 69-75. <new>.

Moore, S.W., G.S. Taylor, P.A. Cunningham, R.D. Mullins et P. Robinson. 2000. « Self calibrating clocks for globally asynchronous locally synchronous systems ». In *Computer Design, International Conference on*. p. 73-78.

Morgensen, John N. 1976. *CMOS Schmitt trigger, Patent 3,984,703*. 7 p.

Morris, Kevin. December 6, 2005. « More and Moore - Xilinx Looks Ahead to 65nm ». *FPGA and Structured ASIC Journal*. p. 4.

Muller, David E., et W. S. Bartky. 1959. « A theory of asynchronous circuits ». In *Proceedings of an International Symposium on the Theory of Switching*. p. 204-243. Harvard University Press, April.

Najibi, M., K. Saleh, M. Naderi, H. Pedram et M. Sedighi. 2005. « Prototyping globally asynchronous locally synchronous circuits on commercial synchronous FPGAs ». In *Rapid System Prototyping, 2005, (RSP 2005), The 16th IEEE International Workshop on*. p. 63-69.

Nguyen, Hoang P, et Richard T. Schultz. 1999. *Meta-hardened flip-flop, Patent 5,999,029.* 11 p.

Nilsson, P., et M. Torkelson. 1996. « A monolithic digital clock-generator for on-chip clocking of custom DSP's ». *Solid-State Circuits, IEEE Journal of,* vol. 31, n° 5, p. 700-706.

Olsson, T., P. Nilsson, T. Meincke, A. Hemam et M. Torkelson. 2000. « A digitally controlled low-power clock multiplier for globally asynchronous locally synchronous designs ». In *Circuits and Systems, Geneva Switzerland, IEEE International Symposium on.* Vol. 3, p. 13-16.

Paxson, Vern. 1995. « FLEX, A fast scanner generator ». <http://dinosaur.compilertools.net/flex/index.html>.

Payne, R. 1996. « Asynchronous FPGA architectures ». *Computers and Digital Techniques, IEE Proceedings,* vol. 143, n° 5, p. 282-286.

Quoc Thai, Ho, J. B. Rigaud, L. Fesquet, M. Renaudin et R. Rolland. 2002. « Implementing asynchronous circuits on LUT based FPGAs ». In *Field-Programmable Logic and Applications, FPL 2002, Proceedings,12th International Conference on.* p. 36-46. Coll. « Field-Programmable Logic and Applications. Reconfigurable Computing Is Going Mainstream. 12th International Conference, FPL 2002. Proceedings (Lecture Notes in Computer Science Vol.2438) ». Montpellier, France: Springer-Verlag.

Rabaey, Jan M., Anantha P. Chandrakasan et Borivoje Nikolic. 2003. *Digital Integrated Circuits: A Design Perpertive*, 2nd. Coll. « Prentice Hall electronics and VLSI series ». Upper Saddle River, N.J.: Pearson Education, xxii, 761 p.

Ramirez, Sergio R. 1998. *Inertial delay circuit for eliminating glitches on a signal line, Patent 5,760,612.* 15 p.

Rapaka, V.S.P., et D. Marculescu. 2003. « A mixed-clock issue queue design for globally asynchronous, locally synchronous processor cores ». In *Low Power Electronics and Design, Proceedings of the 2003 International Symposium on.* p. 372-377.

Rigaud, Jean-Bastiste. 2002. « Spécification de bibliothèque pour la synthèse de circuits asynchrones ». Institut National Polytechnique de Grenoble - INPG, 200 p.

Savaria, Yvon. 1988. *Conception et vérification des circuits VLSI*, 1, École Polytechnique de Montréal. Montréal: Coopérative étudiante de Polytechnique, 398 p.

Schlarmann, M. E., S. Q. Malik et R. L. Geiger. 2002. « Positive feedback gain-enhancement techniques for amplifier design ». In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on.* Vol. 2, p. II-37-II-40 vol.2.

Seitz, Charles L. 1980. « System timing ». In *Introduction to VLSI systems by Carver Mead and Lynn Conway, Addison-Wesley.*

Semeraro, G., G. Magklis, R. Balasubramonian, D.H. Albonesi, S. Dwarkadas et M.L. Scott. 2002. « Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling ». In *High-Performance Computer Architecture, Eighth International Symposium on.* p. 29-40.

Semiat, Y., et R. Ginosar. 2003. « Timing measurements of synchronization circuits ». In *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on*, sous la dir. de Ginosar, R. p. 68-77.

Shams, M., J. C. Ebergen et M. I. Elmasry. 1998. « Modeling and comparing CMOS implementations of the C-element ». *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 6, n° 4, p. 563-567.

Shengxian, Zhuang, Li Weidong, Jonas Carlsson, Kent Palmkvist et Lars Wanhammar. 2002. « Asynchronous data communication with low power for GALS systems ». In *Electronics, Circuits and Systems, 9th International Conference on.* Vol. 2, p. 753-756.

Smith, R., et M. Ligthart. 2001. « High-level design for asynchronous logic ». In *Design Automation Conference.* p. 431-436. Asia and South Pacific.

Smith, S. C. 2007. « Design of an FPGA Logic Element for Implementing Asynchronous NULL Convention Logic Circuits ». *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 15, n° 6, p. 672-683.

Sobelman, G.E., et K. Fant. 1998. « CMOS circuit design of threshold gates with hysteresis ». In *Circuits and Systems, IEEE International Symposium on.* Vol. 2, p. 61-64.

Sparsø, Jens, et Steve Furber. 2001. « Chapters 1-8 ». In *Principles of asynchronous circuit design - A systems Perspective.* p. 1-152. Kluwer Academic Publishers.

Sproull, R. F., I. E. Sutherland et C. E. Molnar. 1994. « The counterflow pipeline processor architecture ». *IEEE Design & Test of Computers,* vol. 11, n° 3, p. 48.

Stallings, William. 1999. *Computer organization and architecture: designing for performance*, Fifth. Alan Apt, 748 p.

Stallman, Richard. 1983. « GNU General Public Licence ». <http://www.gnu.org/copyleft/gpl.html>.

Sutherland, I. E. 1989. « Micropipelines ». *Commun. ACM,* vol. 32, Issue 6 (June 1989), n° 6, p. 720-738.

Sutherland, Ivan E., et Jo Ebergen. 2002. « Computers without Clocks ». *Scientific American,* vol. August, (August 2002), p. 8.

Svantesson, B., S. Kumar et A. Hemani. 1998. « A methodology and algorithms for efficient interprocess communication synthesis from system description in SDL ». In *VLSI Design, 1998. Proceedings., 1998 Eleventh International Conference on.* p. 78-84.

Taubin, A., K. Fant et J. McCardle. 2002. « Design of delay-insensitive three dimension pipeline array multiplier for image processing ». In *Computer Design: VLSI in Computers and Processors, IEEE International Conference on.* p. 104-111.

Taylor, G., S. Moore, S. Wilcox et P. Robinson. 2000. « An on-chip dynamically recalibrated delay line for embedded self-timed systems ». In *Advanced Research in Asynchronous Circuits and Systems, 2000. (ASYNC 2000) Proceedings. Sixth International Symposium on.* p. 45-51.

Teifel, J., et R. Manohar. 2004. « An asynchronous dataflow FPGA architecture ». *IEEE Transactions on Computers,* vol. 53, n° 11, p. 1376-1392.

Teymouri, Sassan. 1992. *Glitch remover circuit for transmission links, Patent 5,113,098.* 6 p.

Traver, C., R. B. Reese et M. A. Thornton. 2001. « Cell designs for self-timed FPGAs ». In *ASIC/SOC Conference, 2001, Proceedings, 14th Annual IEEE International.* p. 175-179.

Unger, S. H. 1969. *Asynchronous Sequential Switching Circuits*. New York: Wiley-Interscience.

Vachoux, Alain. 2002. « VHDL Essentiel ». <http://lsmwww.epfl.ch/Education/former/2002-2003/modelnum03/documents/VHDL_instant.pdf>.

Van Berkel, K. 1992. « Beware the isochronic fork ». *Integration, The VLSI Journal,* vol. 13, n° 2, p. 103-128.

Van Berkel, K., F. Huberts et A. Peeters. 1995. « Stretching quasi delay insensitivity by means of extended isochronic forks ». In *Asynchronous Design Methodologies, 1995. Proceedings., Second Working Conference on*, sous la dir. de Huberts, F. p. 99-106.

Varshavsky, Victor, et Vyacheslav Marakhovsky. 2002. « GALA (Globally Asynchronous - Locally Arbitrary) Design ». In *Concurrency and Hardware Design, Advances in Petri Nets*. Springer-Verlag.

Vivet, P. 2001. « Une méthodologie de conception de circuits intégrés quasi-insensibles aux délais : application à l'étude d'un processeur RISC 16-bit asynchrones ». Grenoble, France.

Weaver, Nicholas, John Hauser et John Wawrzynek. 2004. « The SFRA: A corner-turn FPGA architecture ». In *ACM/SIGDA International Symposium on FPGA*. Vol. 12, p. 3-12.

Wong, C.G., A.J. Martin et P. Thomas. 2003. « An architecture for asynchronous FPGAs ». In *Field-Programmable Technology (FPT), IEEE International Conference on*. p. 170-177.

Xilinx. 2003. *ISE project manager 5.2.03i*. <http://www.xilinx.com/>.

Xilinx. 2005. *Constraints guide 8.1i*. [Online]. Available: http://toolbox.xilinx.com/docsan/xilinx8/books/docs/cgd/cgd.pdf, 361 p.

Xin, Wang, Ahonen Tapani et Nurmi Jari. 2006. « Prototyping a Globally Asynchronous Locally Synchronous Network-On-Chip on a Conventional FPGA Device Using Synchronous Design Tools ». In *Field Programmable Logic and Applications, 2006, FPL '06, International Conference on*. p. 1-6.

Yun, K.Y., et A.E. Dooply. 1999. « Pausible clocking-based heterogeneous systems ». *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 7, n° 4, p. 482-488.