Ecole Polytechnique de Louvain

ICTEAM institute

Conception et optimisation de FPGA à très faible consommation pour l'intégration de logique reconfigurable dans des systèmes autonomes en énergie

Promoteur:
Pr. Jean-Didier LEGAT

Lecteurs:
Dr. David BOL
Pr. Denis FLANDRE

Travail de fin d'études présenté en vue de l'obtention du grade d'ingénieur civil électricien.

Guerric de Streel

Juin 2012

Abstract

Energy-autonomous systems (EAS) are emerging applications in the semiconductor industry characterized by special requirements in term of processing abilities and power consumption. An aggressive cut into processing speed contraints allows reduction of the total power consumption into a power budget limited by a small battery or by an energy harvesting system.

The developpement of such Systems on Chip (SoC) calls for flexibility integration in the design flow that is usually brought by microcontrollers or by multiplying ASICs peripherals. Reconfigurable logic circumvents the intrinsic low performance of software processing in microcontrollers. By bridging the gap between microcontrollers and ASICs, it has become a revolutionary tool for integrated system design. Unfortunately, FPGA fabrics to be embedded suffer a high power overhead compared to dedicated ASICs. The μW range power budget targeted by EAS prevents the integration of classic or even low-power FPGA fabrics.

This work aims at establishing guidelines for the power-constrained design of programmable logic core to be used as embedded IP in EAS or highly power constrained systems. A promissing way to obtain this reduction is through a substantial scaling of the supply core voltage applied to the fabric. Bringing down the supply voltage to Ultra-Low-Voltage (ULV) levels below 0.5V has shown dramatic potential to improve energy efficiency as long as the speed constraint can be relaxed. Operating MOSFETs in weak inversion mode with low drain current exacerbate sensitivity to process variation or leakage current. Maintainin sufficient lifetime while operating at an optimum for the power/performance trade-off implies the development of creative techniques and the adaptation of state of the art scientific research developped for Ultra-Low-Power (ULP) ASICs to the reconfigurable world.

This work shows that, by combining a power-oriented implementation using multi-V_T, a careful repartition of different MOSFET flavors and an aggressive scaling of the core voltage, the static power consumption of a tile can be brought down to $0.5\mu W$. A 16 bit multiplier synthesized on the designed FPGA features an energy per operation as low as $2.5pJ$ wich is $11\times$ more efficent compare to the state of the art ULP ULV microcontroller proving that reconfigurable logic can provide an intermediate solution between software reconfiguration and ASIC implementation in ULV domain. The supply voltage scaling and the precise control over the power/-performance trade-off through V_T partitioning is the subject of a conference paper presented at FTFC 2012.

Along with these solutions, a power gating system is studied to evaluate the possibility to shut down unused area in operation. At higher level, the architecture parameters of logic blocs and routing segments are studied to search for a power optimum. A framework for physical simulation and an interface between high-level place & route layer and low-level simulation level are developped. Finally, a 65nm test-chip is designed to validate the different simulation results.

Abstract

Les systèmes autonomes en énergie sont un secteur d'application émergeant dans l'industrie du semiconducteur. Ils sont carractérisés par des exigeances spécifiques en terme de capacité de calcul et de puissance consommée. Une large réduction de la vitesse de calcul permet d'amener la puissance totale consommée dans un budget de puissance contraint par une petite batterie ou par un système de récupération d'énergie.

Le développement de tels systemes intégrés (Systems on Chip SoC) nécéssite l'intégration de flexibilité dans le flot de conception. Cette flexibilité est généralement apportée par l'intégration de microcontrôleurs ou la multiplication des périphériques ASICs. La logique reconfigurable permet de contourner les faibles performances intrinsèques aux microcontrôleurs. En se plaçant en intermédiaire entre les microcontrôleurs et les ASICs, la logique reconfigurable s'est imposée comme un outil révolutionnaire pour la conception de systèmes intégrés. Malheureusement, les structures FPGA à embarquer souffrent d'une large pénalité en puissance en comparaison aux ASICs. Le budget de l'ordre du μW ciblé par les systèmes autonomes empêche l'intégration de structures reconfigurables classiques ou même, à faible consommation.

Ce travail a pour but d'étudier la conception de blocs de logique reconfigurable fortement contraints en puissance qui pourront être utilisés comme IP dans des systèmes autonomes. Une des pistes pour atteindre des consommations aussi faibles consiste à diminuer la tension d'alimentation appliquée à la structure. Il a été prouvé qu'amener cette tension d'alimentation dans le domaine de la très basse tension (Ultra-Low-Voltage ULV, en dessous de $0.5V$) permet de réduire la puissance totale consommée d'un ordre de grandeur tant que la contrainte en vitesse peut être relâchée. L'utilisation de MOSFET en faible inversion exacerbe la sensibilité aux variations de process ou au courant de fuite. Le maintien d'une durée de vie suffisante tout en imposant la tension optimale au niveau du compromis puissance/performance nécessite le développement de nouvelles techniques ainsi que l'adaptation de l'état de l'art développé pour les ASICs ULP.

Ce travail montre que, en combinant une implémentation ULV multi-V_T orientée vers la réduction de la puissance, la puissance statique d'une tuile de FPGA peut être réduite à $0.5\mu W$. Un multiplieur 16 bit synthétisé sur un tel FPGA présente une énergie par opération de $2.5pJ$, ce qui est $11\times$ plus efficace que l'implémentation de la même multiplication sur un microcontrôleur de l'état de l'art ULP ULV utilisant les mêmes techniques de réduction de la puissance. La logique reconfigurable se place donc en intermédiaire entre la reconfiguration software et l'implémentation ASIC dans le domaine ULV. La réduction de la tension d'alimentation et le contrôle précis du compromis puissance/performance au travers de la répartition du V_T a fait l'objet d'une publication présentée à la conférence FTFC 2012.

A côté de ces solutions, un système de power gating est étudié pour évaluer la possibilité d'éteindre les portions non-utilisées du FPGA. A haut-niveau, les paramètres de l'architecture des blocs de logique et des segments de routage sont étudiés pour chercher l'optimum en terme de puissance consommée. Un framework de simulation physique et une interface entre les logiciels de placement & routage haut-niveau et les simulations physiques bas-niveau ont été développés. Pour finir, un test-chip en technologie $65nm$ est conçu afin de valider les différents résultats de simulation.

Remerciements

J'aimerais remercier mon promoteur, Professeur Jean-Didier Legat, pour son soutien et son enthousiasme pour le concept des eFPGAs et sa foi en ce projet. Merci également de m'avoir proposé, tout au long du projet, différents défis telles que l'écriture d'un article et la réalisation d'un *test-chip* qui m'ont plongé dans le monde de la recherche.

Je tiens également à remercier le laboratoire d'électronique digitale pour l'accueil chaleureux qu'ils m'ont réservé. Merci à David Bol pour sa patience face à la montagne de questions que j'ai pu lui poser et auxquelles il n'a jamais manqué d'avoir réponse. Ses encouragements et son omniscience de la conception digitale ont été décisifs dans la réalisation de ce mémoire. Mes remerciements vont également à Julien De Vos et à François Botman pour leurs réponses à mes questions et pour l'intérêt qu'ils ont porté à ce projet. A l'année prochaine.

Mes remerciements vont aussi à tous mes amis pour leur bonne humeur. Merci à Sophie et Cyrille pour la rivalité amicale qui m'a permis de finir l'écriture dans des délais raisonnables.

Pour finir, merci à mes frères pour leur enthousiasme et à Maman pour son soutien logistique et moral. Enfin, mes pensées vont vers Papa dont j'ai senti le regard bienveillant tout au long de ce travail.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

BLE Basic Logic Element
CAD Computer Aided Design
CB Connection Block
CLB Clustered Logic Block
CLC Configurable Logic Cell
COTS Commercial Off The Shelf
CPLD Complex Programmable Logic Design
CVD Chemical Vapor Deposition
DAG Directed Acyclyc Graph
DFF D Flip Flop
DIBL Drain Induced Barrier Lowering
EAS Energy Autonomous System
EEPROM Electrically Ereasable Programmable Read-Only Memory
eFPGA embedded Field Programmable Gate Array
FBB Forward Body Biaising
FPGA Field Programmable Gate Array
FSM Finite State Machine
GP General Purpose
HVT High-V_T
LUT Look-Up Table
LP Low-power Purpose
LVT Low-V_T
RFID Radio Frequency IDendification
SB Switch Block
SNM Signal Noise Margin
SoC System on Chip
SVT Standard-V_T
TGL Transistor Gate Logic
ULP Ultra-Low-Power
ULV Ultra-Low-Voltage
VPR Versatile Place and Route

Chapitre 1

Introduction

Depuis un demi siècle, l'électronique intégrée a provoqué une révolution du monde technologique et du quotidien en couvrant des applications aussi diverses que l'énergie, les transports, la santé ou les loisirs. Lors de la dernière décennie, l'électronique a changé de visage pour se tourner vers un nouveau défi : la mobilité. L'évolution des technologies et la réduction des dimensions des transistors du micromètre vers le nanomètre ont permis d'intégrer sur des surfaces de plus en plus limitées une puissance de calcul considérable. Cette évolution a donné naissance à une vague d'électronique à faible consommation (*Low-Power* LP) qui a déferlé sur le grand public sous la forme de *notebook*, *smartphones*, *GPS*, tablettes tactiles ou lecteurs mp3.

Dans la continuité de cette tendance, l'électronique mobile moderne doit relever de nouveaux défis. Le domaine des soins de santé demande des systèmes implantables ou portables pour la surveillance ou le soin des patients. L'industrie demande l'intégration de senseurs réalisant des mesures de rendement pour un contrôle fin des lignes de production. Le développement du futur Web3.0 aussi appelé "internet des objets" passe par le déploiement de milliard de noeuds autonomes attachés à des objets.

Cette vision de l'électronique de demain se heurte à différents problèmes : comment contourner la densité énergétique trop faible des batteries actuelles, comment s'affranchir du problème du remplacement de ces batteries, comment équilibrer efficacité énergétique et puissance de calcul (ce qui revient à établir le compromis optimal entre transmission directe ou traitement local des données), comment limiter l'empreinte carbone liée à la fabrication d'un grand nombre de ces puces... ?

Face à ces défis, la recherche s'est penchée sur l'augmentation de l'autonomie des systèmes intégrés pour développer une nouvelle classe de systèmes autonomes en énergie (*Energy Autonomous Systems* EAS). Ces systèmes à haute autonomie sont particulièrement adaptés aux applications dont l'accès pour maintenance est limité tels que les systèmes médicaux implantés. Ils sont caractérisés par des exigences spécifiques en terme de vitesse, de traitement et de puissance consommée. En diminuant substantiellement les contraintes sur la fréquence de fonctionnement, il est possible de réduire la consommation totale d'énergie pour atteindre le budget limité de puissance qu'offrent de petites batteries ou un système de récupération de l'énergie présente dans l'environnement.

Pour réduire la quantité de données que les EAS doivent transmettre, il faut accroitre la puissance de calcul. L'augmentation de la complexité de ces systèmes provoque une augmentation de coûts de développement et de fabrication en technologie avancée. L'intégration de flexibilité permet de partager le coût de développement en couvrant différentes niches industrielles. Dans ces systèmes, la flexibilité est apportée par l'intégration de microcontrôleurs ou en multipliant le nombre de périphériques. La logique reconfigurable permet de contourner les faibles performances intrinsèques au traitement software effectué sur des microcontrôleurs. Elle permet également d'éviter le gaspillage de surface de silicium induit par l'ajout d'une multitude de coprocesseurs, périphériques ou entrées/sorties qui ne sont pas utilisés dans chaque niche. En se plaçant en intermédiaire entre ces deux extrémités, la logique reconfigurable s'est

imposée comme une révolution dans la conception de circuits digitaux et apporte un compromis entre flexibilité et performance. Les FPGA embarqués permettent d'apporter de la flexibilité à différents niveaux dans un *System On Chip* (SoC). Ils peuvent être utilisés afin d'ajouter un coprocesseur, des entrées/sorties ou des périphériques reprogrammables comme illustré dans la Figure 1.1.

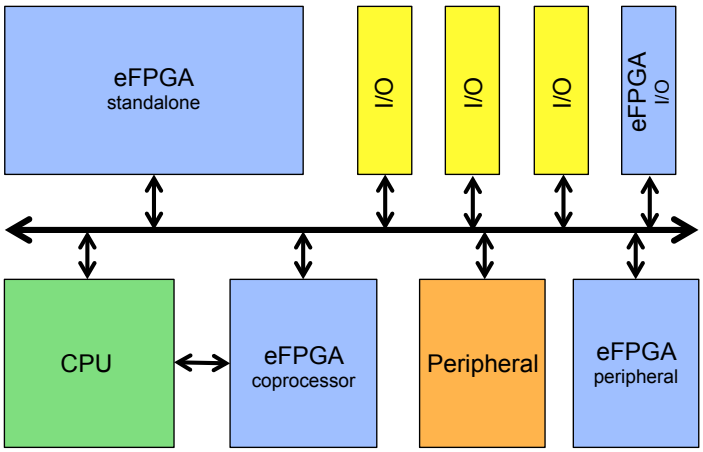


FIGURE 1.1: Illustration de l'insertion d'un eFPGA dans un SoC.

Objectifs, résultats, méthodologie et structure de l'étude

L'industrie et la recherche sont majoritairement centrées sur la réduction de l'écart séparant les FPGAs et les ASICs en terme de performance et de surface. Se faisant, la consommation des FPGAs est généralement sacrifiée. L'ordre de grandeur des consommations des systèmes autonomes en énergie est de l'ordre du μW, ce qui empêche l'intégration de FPGA classiques ou même de FPGA à faible consommation.

L'objectif de ce travail est d'étudier différentes techniques de réduction de la consommation d'un noyau de logique reconfigurable à intégrer dans un système dont la consommation est fortement contrainte.

Deux axes de recherche ont été développés pour atteindre cet objectif. Dans un premier temps, l'impact des paramètres haut-niveau de l'architecture du FPGA sur les performances en vitesse et en consommation est étudié. Dans un second temps, une implémentation *ultra-low-voltage* (ULV) est proposée en utilisant différentes techniques de réduction de consommation développées pour les ASICs *Ultra-Low-Power* (ULP). L'axe de recherche sélectionné pour réduire la consommation consiste à diminuer la tension d'alimentation jusqu'aux ultra faibles tensions (en dessous de $0.5V$). Cependant, les transistors en faible inversion ont un faible courant de drain qui augmente la sensibilité aux variations de fabrication ou aux courants de fuite. Le maintien d'un rendement de fabrication et d'une durée de vie raisonnable, tout en appliquant une tension optimale au point de vue énergétique, impose le développement de nouvelles techniques et l'adaptation de l'état de l'art développé pour les ASICs ULP à la logique reconfigurable.

Les résultats principaux de cette étude sont :

- La modélisation de l'impact de la taille des *Look-Up Table* et des grappes logiques sur la consommation statique et dynamique.
- L'optimisation de l'architecture des *Look-Up Table* et de leur implémentation en technologie $65nm$ utilisant un mélange LP-GP.
- L'optimisation de l'implémentation du *Cluster Logic Bloc* CLB et de la structure de routage en passant par l'étude de la répartition optimale des différents V_T et des différents types de transistors. Ces résultats ont fait l'objet d'un article accepté pour publication à la conférence FTFC 2012 ([19]).

- Le développement d'une structure de simulation physique d'un FPGA.
- Le développement d'une couche software réalisant l'interface entre le placement et routage haut-niveau et les simulations physiques bas-niveau.
- La réalisation d'un *test chip* en technologie $65nm$ pour valider les résultats obtenus par simulation.

Ce travail est divisé en 4 chapitres. Le premier reprend l'état de l'art lié à la conception de FPGA ainsi qu'à celle de circuits digitaux ULP ULV. Le second traite de l'influence des paramètres haut-niveau et de l'architecture du FPGA sur les performances en vitesse et en consommation. Le troisième décrit les détails de l'implémentation ULP du FPGA ainsi que les différentes techniques mises en oeuvre pour réduire la consommation du FPGA à bas niveau. Pour finir, le quatrième chapitre reprend les étapes de la conception du *test chip* ainsi que différentes remarques sur le chemin restant à parcourir pour atteindre une implémentation performante d'un bloc de logique reconfigurable pour systèmes autonomes.

Chapitre 2

Etat de l'art : architecture et optimisation des FPGAs et des circuits digitaux ULP

Ce chapitre donne un aperçu de l'évolution des différents blocs constitutifs des FPGAs depuis leur apparition dans le courant des années 1980 jusqu'aux tendances dominantes dans les structures modernes. Les différentes optimisations de l'architecture des blocs logiques et de la structure de routage pour minimiser la surface, le délai et la puissance consommée y sont abordés ainsi que les avantages et inconvénients des différentes méthodes d'implémentation de la configurabilité. Dans un second temps, un rapide aperçu de la conception de circuits digitaux ULP en technologie CMOS nanométrique est dressé. Un état de l'art de techniques utilisées pour limiter la consommation des FPGAs permet de rassembler les notions des deux sections précédentes pour former un point de départ à l'implémentation proposée dans la suite de ce travail.

L'intégration de logique programmable dans un circuit à embarquer soumis à des contraintes énergétiques strictes, nécessite une adaptation du processus de conception. En effet, la conception de FPGA *Commercial Off The Shelf* (COTS) ne tient généralement pas en compte la puissance consommée comme critère d'optimisation. L'industrie et la recherche dans le domaine de la logique reconfigurable sont majoritairement orientées vers la recherche de vitesse pour parvenir à approcher une réalisation ASIC d'un circuit tout en conservant la reconfigurabilité. La référence [1] indique que les FPGAs sont entre 3 et 4 fois plus lents en moyenne que les implémentations équivalentes en ASIC. Ils occupent entre 20 et 35 fois plus de surface et consomment jusqu'à 10 fois plus de puissance dynamique.

L'antagonisme entre performance et consommation a ralenti l'apparition de matrices reconfigurables dans le domaine de la faible consommation. Cependant, certains fabricants commencent à introduire des blocs de logique implantables ou à ajouter des périphériques reconfigurables dans des *Systems on Chip* SoC (cf. Section 2.4). De plus, certaines publications concernant l'adaptation de techniques de réduction de consommation utilisées dans la conception d'ASICs indiquent que la recherche pour la réduction de consommation évolue (cf. Section 2.3).

Malgré cette diversification, le domaine de la très faible consommation (ULP) reste hors d'atteinte. Il n'existe pas encore de publication concernant la conception de FPGA ULP.

2.1 Architecture des FPGAs

La majorité des FPGAs modernes est structurée en îlot. Il s'agit d'une matrice de blocs de logique liés par des canaux de routage, illustrée dans la Figure 2.3. Cette matrice est entourée d'un anneau de contact (I/O). L'architecture d'un FPGA comprend une description haut-niveau de la logique et des interconnexions.

L'importance relative du routage et des blocs logiques varie selon le facteur de mérite considéré : puissance, délais ou surface. L'architecture haut-niveau sélectionnée joue également un rôle important dans la répartition des performances dans le FPGA. En guise d'exemple, citons les performances en vitesse des circuits programmés sur le FPGA qui sont liées à la qualité du routage alors que la quantité de routage nécessaire sur le *critical path* est liée à l'architecture des blocs logiques ([13]).

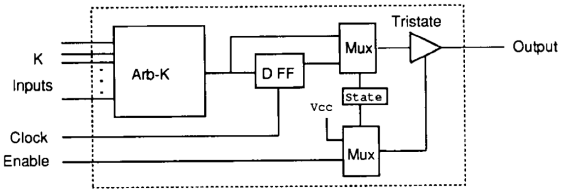
2.1.1 Structure des blocs logiques

Les blocs logiques définissent la finesse du grain du FPGA. Leur conception est liée à un compromis entre des blocs à grains fins qui sont rapides mais qui nécessitent une grande quantité de routage externe peu performant et des blocs à gros grains qui sont plus lents mais dont le routage externe est moins grand.

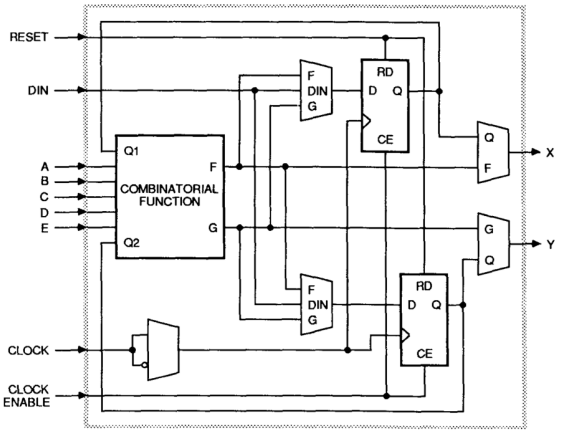
Historiquement, certains FPGAs utilisaient le transistor comme bloc élémentaire qui représente un extrême dans ce compromis. Cette architecture permet une utilisation optimale des ressources car chaque porte des circuits programmés sur le FPGA utilise le nombre minimal de blocs logiques. Cependant, les performances de ce type d'architecture sont lourdement handicapées par la grande quantité d'interconnexions et de cellules de configuration nécessaire. L'extrême opposé de ce compromis pourrait être l'intégration d'un processeur dans chaque bloc logique (*Network on Chip* NoC).

A la fin des années 1980, la référence [10] propose d'intégrer dans une interconnexion implémentée avec des *antifuses* (cf. Section 2.1.5) des blocs logiques composés de 3 multiplexeurs à 2 entrées. Au début des années 1990, Xillinx intègre dans la famille des XC3000 ([11], cf. Fig. 2.1 (b)) un bloc logique "moderne" composé d'une LUT à 5 entrées. La référence [12] explore le dimensionnement de ce type de blocs logiques (cf. Fig. 2.1 (a)), aussi appelé *Basic Logic Element* (BLE), composé d'une LUT et d'un Flip-Flop pour implémenter un comportement séquentiel.

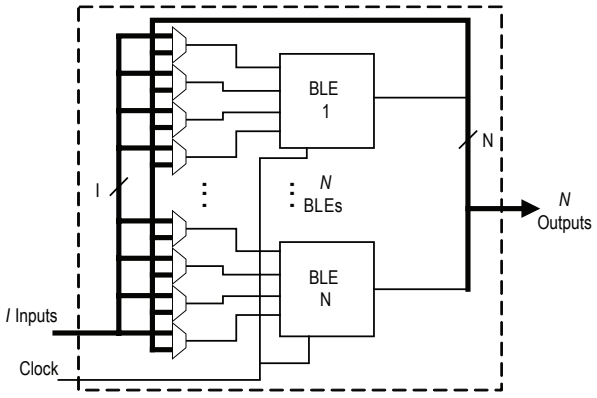
Les FPGAs modernes introduisent une hiérarchie entre le routage global et le routage local en regroupant des BLEs dans des grappes. La grappe ou *Clustered Logic Block* (CLB) fait également l'objet d'un compromis. En effet, si le nombre de BLEs qu'elle contient augmente, le routage externe va diminuer car elle pourra contenir davantage de logique mais le routage interne va ralentir à mesure que sa complexité augmente. Un CLB (cf. Fig. 2.1 (c)) est caractérisé principalement par 3 paramètres : la taille des LUTs (K), le nombre de LUTs (N) et le nombre d'entrées du CLB (I). L'optimisation de ces paramètres par rapport à la surface ou à la vitesse, a fait l'objet de nombreux travaux de recherche. Citons entre autre [13] [14] [15] [16] dont l'analyse détaillée figure dans les Sections 2.1.3, 2.1.4.1, 2.1.4.2.



(a) Forme générique d'un Basic Logic Element développé dans [12]. Le bloc Arb-K représente la LUT capable d'implémenter toute fonction logique de K entrées. Le BLE comporte également un registre et un *tristate buffer* à la sortie. Ce buffer n'est pas toujours nécessaire selon l'architecture du routage global (cf. Section 2.1.2)



(b) BLE implémenté dans la famille XC3000 (Xilinx [11]). On remarque la structure similaire au BLE générique illustré dans la Figure 2.1 (a)



(c) Architecture d'un CLB [13].

FIGURE 2.1: Description des blocs logiques

2.1.1.1 Hétérogénéité

De nombreux FPGAs modernes intègrent, en plus des blocs de logique reprogrammable (*general purpose logic*), des blocs de *specific purpose logic*. Ceux-ci permettent d'implémenter certaines fonctions avec des performances supérieures en vitesse, surface et consommation par rapport à une implémentation sur les blocs de *general purpose logic*. En revanche, si la fonction logique implémentable sur ces blocs spécifiques n'est pas utilisée, la surface et la consommation statique sont gaspillées [1]. Pour éviter ce gaspillage, les fabricants ont développé des sous-familles d'une même structure entre lesquelles le rapport *general purpose logic-specific purpose logic* varie.

La *specific purpose logic* présente dans les FPGAs modernes comprend généralement des structures arithmétiques de *carry*, multiplication ou addition ainsi que des éléments de mémoire. On distingue alors deux types d'hétérogénéité :

- hétérogénéité *soft fabric* dans laquelle les structures spécifiques sont intégrées dans les tuiles de logique au côté des structures de *general purpose logic*. L'intégration de DFF dans le BLE ou l'intégration de chaines de propagation du *carry*, d'additionneurs ou de soustracteurs en sont des exemples. Certaines structures vont plus loin en intégrant des *carry lookahead* ou *carry skip*.
- hétérogénéité *hard fabric* dans laquelle les structures spécifiques sont situées dans des tuiles dédiées et séparées des tuiles de *general purpose logic*. Les blocs de RAM et de multiplieurs présents dans de nombreuses structures modernes en sont des exemples ([29], [30]).

La Figure 2.2 illustre l'intégration d'hétérogénéité *hard fabric* dans la famille des Stratix d'Altera. On retrouve différents types de blocs mémoire ainsi que des blocs DSP regroupant un multiplieur 18×18, un additionneur, un accumulateur et un filtre *Finite Impulse Response* FIR.

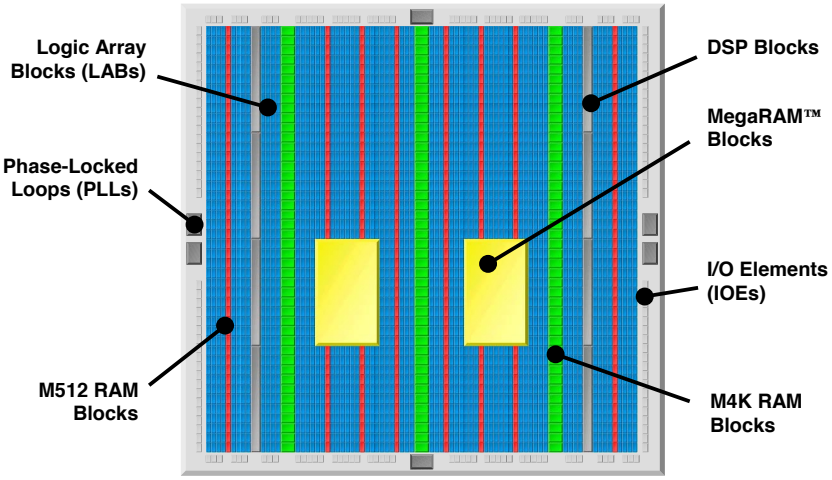


FIGURE 2.2: Illustration de l'hétérogénéité dans la famille des Stratix d'Altera [27].

Les blocs de mémoire qui sont intégrés, présentent généralement une fonctionnalité double-port et parfois un mode de configuration en FIFO. D'autres modes d'opération peuvent être supportés en implémentant de la logique de contrôle sur les blocs reconfigurables à proximité.

Certaines familles de FPGAs intègrent également des microprocesseurs au sein de la structure qui sont plus performants que l'implémentation sur la *general purpose logic*. La famille des Virtex 5 de Xilinx intègre ainsi jusqu'à 2 Power PC.

1. En guise d'exemple, les DFFs présents dans les BLEs font partie des structures spécifiques car ils pourraient être réalisés en utilisant les LUTs et les interconnections. Une implémentation des DFFs dans la *general purpose logic* souffre d'une telle pénalité en performances qu'il n'existe plus aucune structure commerciale qui n'intègre pas les DFFs dans les BLEs.

TABLE 2.1: Comparaison de l'hétérogénéité présente dans la famille des Virtex 5 de Xilinx et dans la famille des Stratix III d'Altera

	#LUT [K]	DSP	Total RAM [KB]	PLLs	*µprocessor* IP	Total I/O
Xilinx Virtex V	6-LUT 12.48 – 207.36	25 × 18 mult+ adder+accumulator 32 – 1056	936 – 18576	1 – 6	PowerPC hard *None* – 2	172 – 1200
Altera Stratix III	4-LUT 475 – 3375	18 × 18 mult+ adder+accumulator+FIR 216 – 768	2430 – 20490	4 – 12	Nios II soft -	296 – 1120

2.1.2 Structure de routage

Le routage d'un FPGA permet de relier les différents blocs logiques implémentant des séries de portes logiques entre eux ou aux I/Os. Cette portion de la structure est critique pour les performances du FPGA. En effet, elle doit permettre une bonne connectivité entre les différents blocs tout en conservant de bonnes performances en vitesse, surface et puissance consommée. La demande en routage peut varier fortement d'un circuit à l'autre, tout comme le ratio entre quantité de segments courts (locaux) et de segments longs (globaux). La structure de routage doit donc permettre le routage d'un maximum de circuits différents, tout en préservant les performances. En supplément du réseau de routage standard, il y a un certain nombre de signaux "logistiques" comme les *clocks* ou les *reset* (il peut y avoir jusqu'à 64 *clocks* distribuées dans les gros FPGAs modernes) qui doit être distribué en contrôlant précisément le *skew*.

2.1.2.1 Routage d'un FPGA en îlot

La Figure 2.3 montre la répartition des canaux de routage pour une structure en îlot. Lors de la conception, le nombre de segments de routage par canal W ainsi que la distribution des segments globaux ou locaux sont fixés.

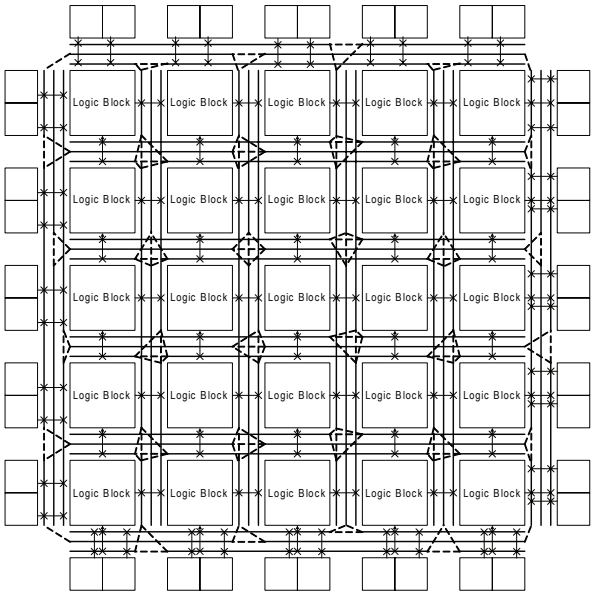


FIGURE 2.3: Structure de routage pour un FPGA en îlot [1]

La référence [21] définit l'architecture de routage à travers 5 points caractéristiques :

1. La longueur des segments de routage en terme de nombre de blocs logiques le long desquels le segment s'étend.
2. L'architecture des interrupteurs de routage.
3. La répartition de ces interrupteurs et la connectivité qu'ils assurent entre les *Connection Blocks* CB, *Switch Blocks* SB et CLBs.
4. Les dimensions des transistors utilisés pour implémenter les interrupteurs.
5. Les dimensions des pistes de métal qui implémentent physiquement les segments (largeur et espacement).

Les 3 premiers points font l'objet d'une discussion dans les paragraphes suivants tandis que les 2 derniers points, liés à l'implémentation bas-niveau, sont examinés dans les Chapitres 4 et 5.

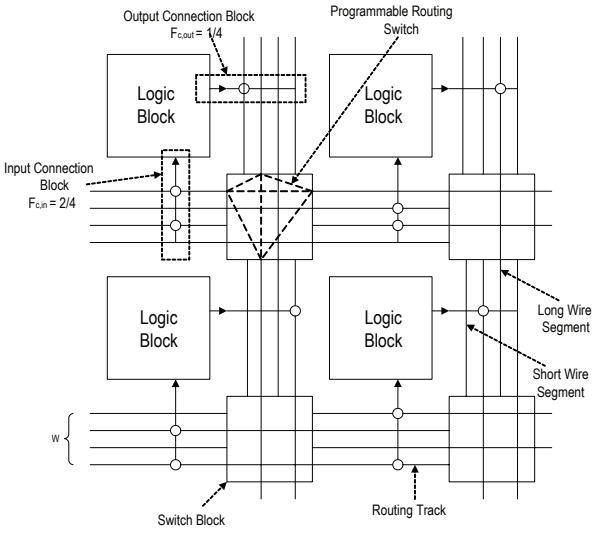
2.1.2.2 Structure détaillée et discussion de la connectivité

Chaque CLB possède I entrées et N sorties qui doivent être desservies par la structure de routage. Le nombre d'interconnexions qui relient les entrées des CLBs aux segments qui passent à proximité détermine la connectivité des entrées (aussi appelée flexibilité dans [23]). On définit cette connectivité $F_{c,in}$ comme le rapport entre ce nombre d'interconnexions possibles et la largeur des canaux de routage W. Plus $F_{c,in}$ est élevé, plus les canaux seront atteignables par les CLBs et donc plus le routage sera flexible. Cependant, cette connectivité est également soumise à un compromis car plus elle augmente, plus le nombre de *switches* nécessaires pour l'implémentation augmente et plus les performances en vitesse et en surface diminuent. A l'inverse, si le routage n'est pas suffisamment flexible, la synthèse de circuits demandera un W minimum élevé qui génèrera également une pénalité en vitesse et surface. On peut définir la connectivité des sorties $F_{c,out}$ de la même manière. Les connections entre les segments verticaux et horizontaux sont réalisées dans les *Switch Blocks* (SBs) qui sont également caractérisés par une flexibilité. Les différentes flexibilités du routage sont illustrés dans la Figure 2.4.

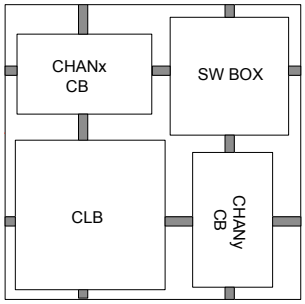
La référence [23] discute la connectivité à accorder à ces différents blocs. La flexibilité F_s d'un SB est définie comme le nombre de connections possibles qui peuvent être réalisées à partir d'un segment entrant comme illustré dans la Figure 2.5 (a). Le SB est également défini par sa topologie. La Figure 2.5 (b) montre deux SBs de même flexibilité ($F_s = 2$) mais présentant une topologie différente qui influence les performances des algorithmes de routage. La référence [23] illustre cela en indiquant que la connexion des points A et B de la Figure 2.5 (b) n'est réalisable que dans la seconde topologie.

La référence [23] tente de déterminer F_s, F_c (F_c est définit comme $F_{c,in} \times W$) et W pour parvenir au routage d'une série de circuits *benchmark*. La Figure 2.6 (a) présente l'évolution du nombre de connections routées avec succès sur le FPGA en fonction de F_c. On remarque que le routage complet n'est possible qu'à partir de $F_c \geq 0.5 \times W$ et ce pour la majorité des valeurs de F_s. La Figure 2.6 (b) précise cette tendance en montrant l'évolution du pourcentage de connections routées en fonction de F_s. On constate que, si F_c est élevé, le routage total des circuits de *benchmark* est atteint pour des F_s petits (jusqu'à trois). Descendre F_s en dessous de trois implique qu'un segment entrant ne peut pas se connecter aux trois directions sortantes, ce qui brise la symétrie du FPGA et il n'y a pas de raison de favoriser une direction par rapport aux autres. Les résultats de [23] permettent donc de conclure que la flexibilité du CB doit être élevée pour ne pas dégrader l'accessibilité des CLBs. Il est en effet inutile d'augmenter le nombre d'entrées des CLBs dans l'optique de garantir un bon taux de remplissage de ceux-ci si ils ne sont pas correctement desservis (cfr. Section 2.1.4.1). Cependant, les auteurs ont constaté que la contrainte sur la flexibilité du SB est moins importante.

Les auteurs utilisent un bloc logique composé uniquement d'une LUT et d'un DFF qui représente l'état de l'art des FPGAs du début des années 1990. Leurs résultats ne sont donc pas directement transposables aux FPGAs modernes composés de CLBs et dont la structure de routage est composée de segments de différentes longueurs. De plus, les auteurs ne peuvent pas tirer de conclusion sur les valeurs de F_s et F_c optimales vis-à-vis de la surface ou de la vitesse car ces facteurs de mérite dépendent fortement des blocs logiques (cfr. Section 2.1.4) et de l'implémentation bas-niveau. Une étude centrée uniquement sur le routage haut-niveau permet seulement d'évaluer la capacité de leur algorithme à

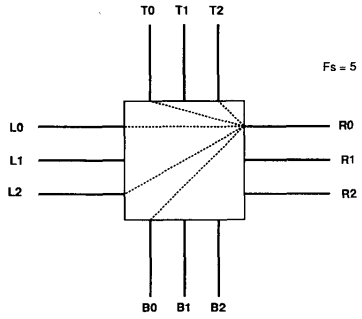


(a) Représentation haut-niveau de la flexibilité de la structure de routage [28].

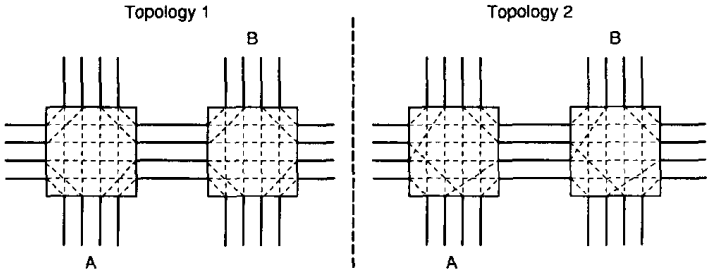


(b) Schéma d'une tuile de FPGA comportant le bloc logique et sa structure de routage proche [19].

FIGURE 2.4: Structure de routage pour un FPGA en îlot (suite).



(a) Définition de la flexibilité d'un SB. En partant de R_0, 5 connections peuvent être effectuées [23].



(b) Illustration de l'impact de la topologie du SB sur le routage d'une connection A vers B [23].

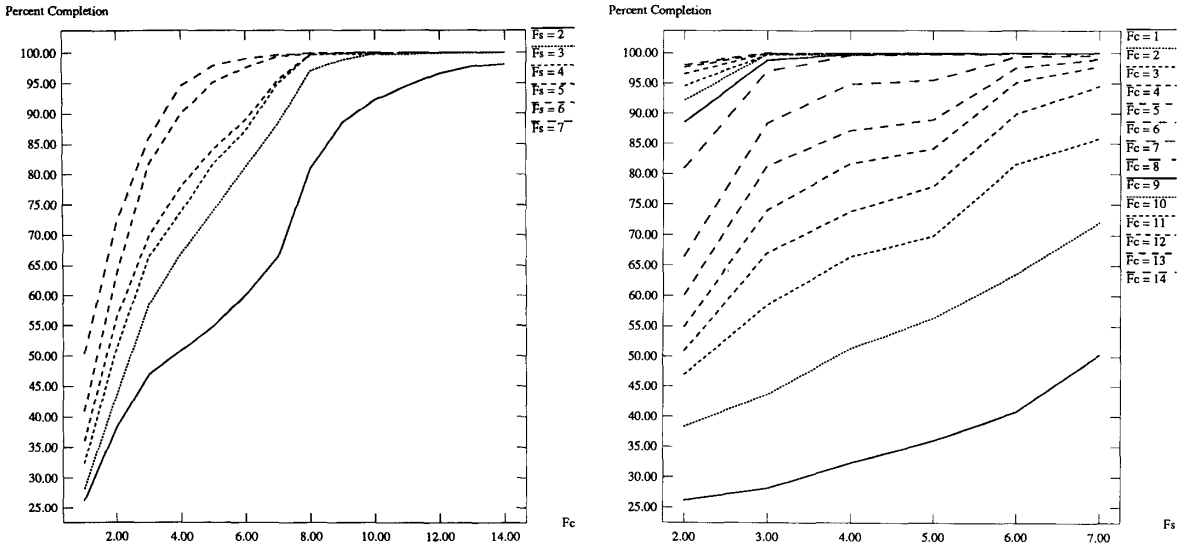
FIGURE 2.5: Structure et propriétés du *Switch Box*

synthétiser facilement mais ne donne pas d'informations sur les performances des résultats. Il n'existe aucune étude similaire à celles décrites dans la Section 2.1.4 qui détermine les valeurs optimales pour les flexibilités.

La topologie des SB fait l'objet d'un grand nombre de recherches. Les figures 2.7 (a) (b) (c) présentent trois architectures répandues. Les deux premières architectures ne permettent pas de connecter deux segments qui n'ont pas le même indice (position dans le canal). La Figure 2.7 (d) montre une situation pour laquelle les deux premiers blocs ne peuvent être utilisés car ils ne permettent pas de relier A et B tandis que le bloc (c) de Wilton permet de lier deux segments d'indices différents augmentant ainsi la flexibilité. Ce raisonnement doit être nuancé lorsque les segments s'étendent sur plusieurs CLBs car la topologie de Wilton pour les segments plus longs impose la multiplication du nombre de *switches*.

2.1.2.3 Distribution de longueurs des segments

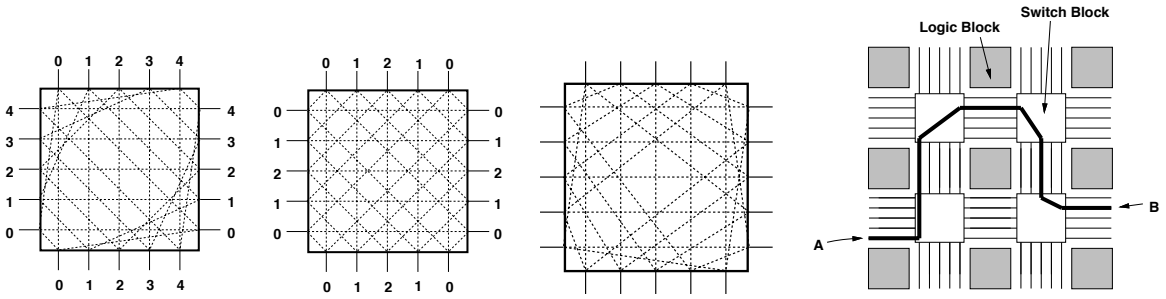
La majorité des FPGAs modernes possède une distribution de segments longs pour le routage global et de segments courts pour le routage local. La Figure 2.8 (a) illustre le concept de distribution



(a) Evolution du nombre de connections routées avec la flexibilité du CB. W est fixé à 14.[23]

(b) Evolution du nombre de connections routées avec la flexibilité du SB. W est fixé à 14.[23]

FIGURE 2.6: Impact de la flexibilité sur les performances de routage.



(a) SB disjoint ou *subset* (Xillinx) [24].

(b) SB universel [25].

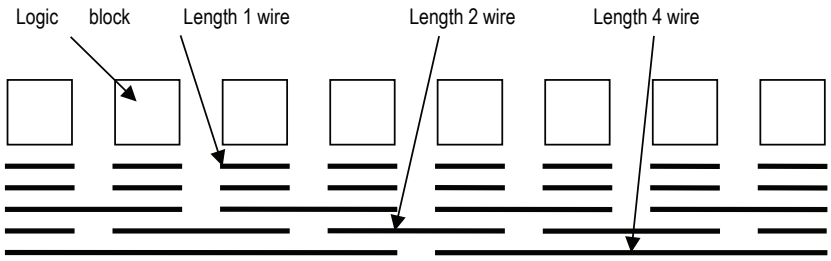
(c) SB de Wilton [24].

(d) Illustration de l'effet de la topologie [24].

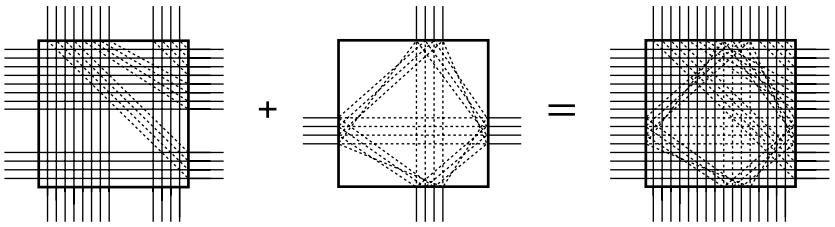
FIGURE 2.7: Topologie des *Switch Blocks*

de segments qui a fait l'objet de différentes recherches dont les travaux de Betz [21]. La référence [22] comprend une comparaison des différentes distributions et les auteurs concluent que les segments de longueur intermédiaire (s'étalant sur 4 à 8 CLBs) mènent aux meilleurs performances en vitesse. Cette conclusion est validée par l'architecture de la famille Stratix (FPGA haute performance [27]) d'Altera qui comprend un mélange de segments de longueur 4 et de longueur 8.

La présence de segments de longueur supérieure à 1 implique une modification des SBs. Ceux-ci doivent pouvoir laisser passer les segments qui ne s'arrêtent pas dans le SB. L'implémentation de ces structures est détaillée dans le Chapitre 4. Le SB doit être revu pour rester optimal. La référence [27] présente le SB de Iram qui est repris dans la Figure 2.8 (b). Ce SB comprend un SB de Wilton pour les segments qui s'arrêtent dans le SB et une série de connections du même type que celles des SB disjoints pour les segments qui traversent le SB. Ceci permet d'implémenter des connections entre le milieu d'un segment et le début d'un autre en minimisant la surface.



(a) Illustration de la distribution des longueurs de segments [28].



(b) SB Imran [26] pour W=16 et L=4. A gauche, connections entre les segments qui ne s'arrêtent pas au SB, au milieu, connections de type Wilton entre les segments qui s'arrêtent au SB, à droite, le SB total.

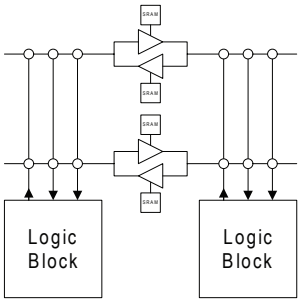
FIGURE 2.8: Distribution de longueurs des segments et impact sur la topologie du SB.

2.1.2.4 Architecture des *drivers*

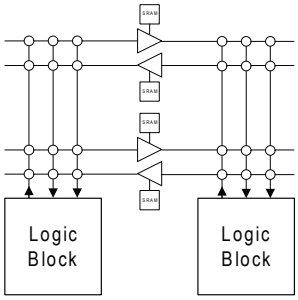
Les segments peuvent être de deux types : bidirectionnels ou unidirectionnels ([22]). Ce qualificatif décrit la possibilité d'un segment à implémenter une connexion qui va de gauche à droite (resp. bas en haut), de droite à gauche (resp. haut en bas) ou les deux en fonction de sa configuration. Les figures 2.9 (a) et (b) reprennent une illustration du concept de direction. Les segments bidirectionnels sont implémentés avec deux *tri-state buffer* pour permettre la désactivation du *driver* qui n'est pas dans la bonne direction. Après programmation, 50% des *drivers* sont donc inactifs pénalisant surface, consommation (courant de fuite) et délai (capacités en supplément). Un segment unidirectionnel ne nécessite qu'un seul *tri-state buffer* qui ne sera jamais gaspillé si le segment est actif.

La référence [22] distingue, au sein des implémentations unidirectionnelles, deux types de segments : *multi-driver* ou *single-driver* qui sont illustrés dans les figures 2.9 (c) et (d). Les segments *multi-driver* peuvent être contrôlés par un *driver* situé au milieu du segment mais la présence de plusieurs *drivers* impose l'utilisation de *tri-state buffer* pour mettre en haute impédance tous les drivers qui ne sont pas maître du segment. Un segment qui ne possède qu'un seul *driver* à son point d'origine peut utiliser un *driver* classique, ce qui permet une réduction de la surface ou, pour une même surface, une augmentation de la force de *driving*. Cependant, la sélection du signal à envoyer sur le segment doit alors être réalisée par un multiplexeur situé au point d'origine du segment imposant une proximité entre l'origine du signal à envoyer et le début du segment, ce qui constitue une contrainte supplémentaire sur le routage.

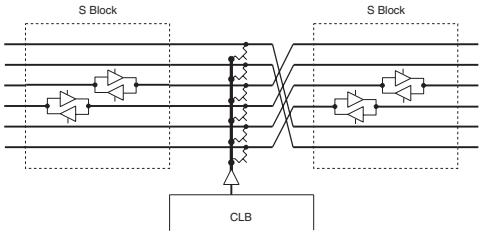
Les auteurs de [23] ont réalisé une comparaison entre ces trois architectures de segments (segments bidirectionnels *multi-drivers*, segments unidirectionnels *multi-drivers*, segments unidirectionnels *single-driver*). Ils ont observé que la surface totale du FPGA diminue de 20% lorsque l'on passe d'une architecture bidirectionnelle à une architecture unidirectionnelle *multi-driver* tandis que le délai moyen sur le chemin critique augmente légèrement (3%). L'étude de l'architecture unidirectionnelle *single-driver* montre une réduction de 18% de la surface et de 16% du délai par rapport à l'architecture bidirectionnelle. La réduction de la surface vient du passage aux segments unidirectionnels qui sont largement moins chargés (cfr. Chapitre 4). Pour finir, on peut remarquer que le passage au *single-driver* simplifie la topologie du SB qui ne doit plus implémenter de connections entre les milieux de deux segments qui se croisent.



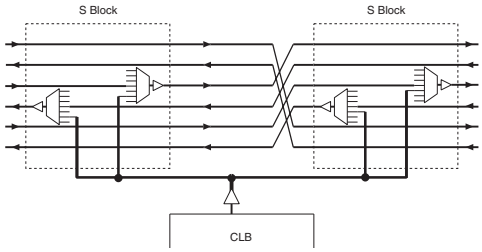
(a) Segments bidirectionnels [22].



(b) Segments unidirectionnels [22].



(c) Segments *multi-driver* [22].



(d) Segments *single-driver* [22].

FIGURE 2.9: Directionnalité et répartition des *drivers*.

2.1.3 *CAD Flow* d'optimisation

La versatilité des FPGAs impose le développement d'un flot de conception (*CAD Flow*) spécifique pour pouvoir tester les performances d'une architecture. L'idée reprise dans la plupart des travaux de recherche dont [13] [14] [15] [16] est de simuler la programmation d'une série de circuits de *benchmark* sur l'architecture testée et d'observer quelques facteurs de mérite tels que la surface totale ou le délai sur le chemin critique.

Pour placer et router un fichier verilog sur une architecture non-standard, une série d'opérations doit être effectuée :

1. Conversion du fichier verilog en fichier .blif contenant une description de la *netlist* manipulable par la plupart des outils de synthèse.
2. Une étape d'optimisation logique peut-être appliquée pour minimiser le nombre d'états des FSMs ou pour retirer les redondances. Cette étape est identique à l'étape d'optimisation logique que l'on trouve dans un *flow* ASIC.
3. La *netlist* est ensuite traduite en un graphe et décomposée en sous-graphes de taille K [2] qui seront implémentés dans les LUTs. La décomposition du graphe en un réseau de sous-graphes de taille K est un problème d'optimisation complexe. Après cette étape, la *netlist* ne contient plus que des K-LUTs et des DFFs.
4. Les K-LUTs et les DFFs sont groupés pour former des BLEs.
5. La *netlist* décomposée en BLE est ensuite confrontée à un fichier de description de l'architecture étudiée et les BLEs sont regroupés dans des CLBs qui sont répartis sur la grille du FPGA.
6. Les CLBs placés sont routés avec le nombre minimum de pistes possibles dans chaque canal.
7. Une estimation de la surface totale de FPGA nécessaire pour synthétiser le circuit ainsi que le délai sur son chemin critique sont calculés sur base du résultat du placement et du routage sur

2. Un sous-graphe de taille K, aussi appelé *K-feasible cone*, est un sous-graphe partant d'un noeud (porte logique) spécifique et comprenant le plus de noeuds possibles tout en ayant au maximum K entrées.

TABLE 2.2: *CAD Flow*

	Etape	Software	Informations sur l'architecture
1	Conversion .v vers .blif	ODIN[53]/QUARTUS [51]	-
2	Optimisation logique	SIS[52]/QUARTUS[51]/ABC[54]	-
3	Mapping technologique en K-LUTs & DFFs	FlowMap [50] /ABC[54]	K
4	Rassemblement et placement des LUTs & DFFs dans les CLBs	T-VPACK[17]	N, I
5	Placement des CLBs	VPR[17]	description détaillée de l'architecture
6	Routage	VPR[17]	description détaillée de l'architecture
7	Evaluation de surface & délai	VPR[17]	description détaillée de l'architecture & informations sur l'implémentation

l'architecture ainsi qu'avec des informations sur les performances de l'architecture (surface du transistor de taille minimale et délai à travers différents éléments).

Ces étapes et les logiciels permettant leurs réalisations sont repris dans la Table 2.2.

2.1.4 Optimisations

2.1.4.1 Optimisation de la surface

La surface du FPGA est estimée par le logiciel VPR ([17]) qui prend comme donnée la surface d'un CLB, la surface et la résistance du NMOS et PMOS de taille minimale et une description de l'architecture des *switches*. Si le nombre de transistors dans un *switch* est donné en terme de MOS de taille minimale équivalente, le logiciel somme les contributions des CLBs et du routage pour obtenir une estimation de la surface totale. Si la résistance du *switch* est donnée, le logiciel extrapole le nombre de MOS de taille minimale pour obtenir cette résistance donnant lieu à une estimation moins précise de la surface.

Ces estimations se basent sur l'hypothèse que la surface des canaux de routage est dominée par la surface des *switches* et non par la surface des connections métalliques. La référence [13] souligne que cette hypothèse est valable pour des technologies incluant un grand nombre de couches de métal disponibles pour le routage. La surface d'un MOS de taille minimale comprend, en supplément du transistor en lui même, l'espace minimum vertical et horizontal à respecter entre deux MOS (Fig. 2.10).

La surface du FPGA nécessaire à la synthèse d'un circuit donné est liée à la quantité de logique intégrable dans chaque bloc logique. Si ces blocs sont agrandis, leur nombre diminue pour une même fonctionnalité à synthétiser mais la surface de chaque bloc augmente. La référence [3] formalise ce compromis en posant N_j^i, le nombre de blocs logiques nécessaires pour implémenter un circuit i sur une architecture j et LR_j^i, la somme de la surface d'un bloc logique et du routage qui l'entoure dans l'architecture j. L'équation :

$$A_j^i = N_j^i \times LR_j^i \tag{2.1}$$

donne la surface totale de FPGA nécessaire pour implémenter le circuit en question. Pour l'optimiser, la fonctionnalité du bloc logique doit être accrue jusqu'à ce que la réduction de N_j^i s'équilibre avec l'augmentation de LR_j^i.

Impact de K et N : la référence [13] illustre le compromis entre surface des LUTs et nombre de LUTs par circuit dans un graphe repris sur la Figure 2.11 (a). On constate que la surface des CLBs (qui sont équivalents aux BLEs car les simulations sont faites pour N=1) croît exponentiellement avec K. Cette tendance s'explique par la croissance du nombre de bits de configuration (2^k par K-LUT) et par la hausse du nombre de multiplexeurs utilisés pour réaliser la LUT (K-1 par K-LUT). En contrepartie,

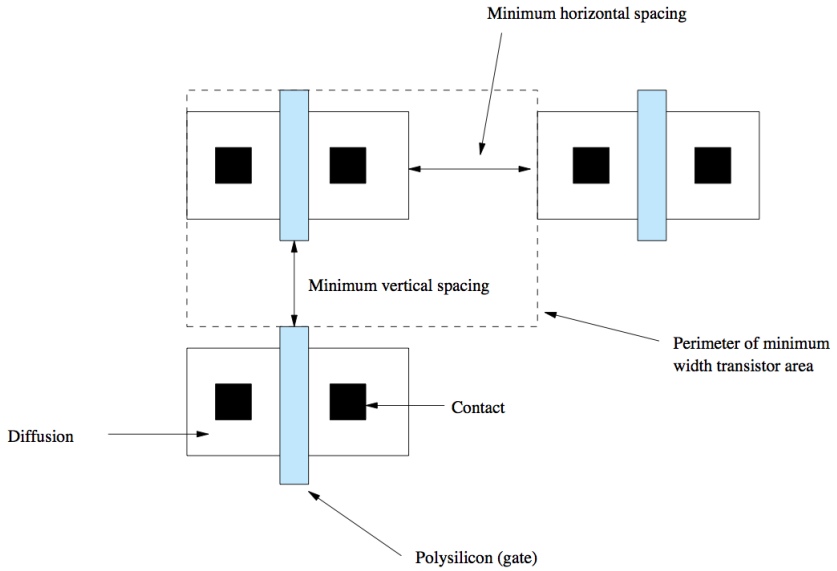
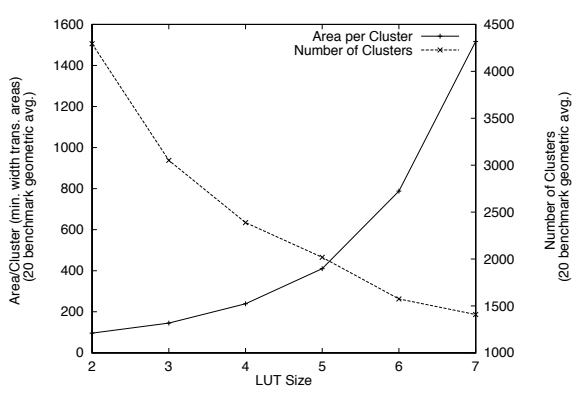
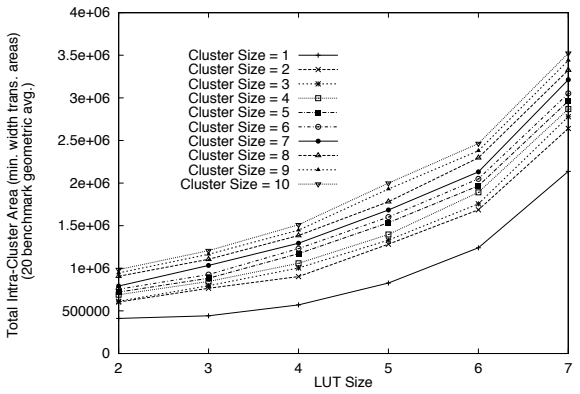


FIGURE 2.10: Description du concept de taille minimale de MOS dans le modèle de la surface utilisé par VPR [17].

l'augmentation de la capacité logique des LUTs permet une diminution du nombre moyen de CLB par circuit (Fig. 2.11 (a)). En supplément de la hausse de la taille du BLE, la surface du routage local composé de $K \times N$ multiplexeurs de $I + N$ entrées augmente aussi avec K. Pour finir, le nombre d'entrées I doit augmenter avec K pour maintenir un taux d'accessibilité raisonnable. Le produit des deux courbes de la Figure 2.11 (a) donne la surface totale comprise dans les CLBs et la Figure 2.11 (b) reprend l'évolution de la surface totale de tous les CLBs pour différents N.



(a) Evolution de la surface d'un CLB et du nombre moyen de CLB utilisés pour implémenter les circuits du *benchmark* (N=1).



(b) Evolution de la surface intra-CLB avec la taille des LUTs et des CLBs.

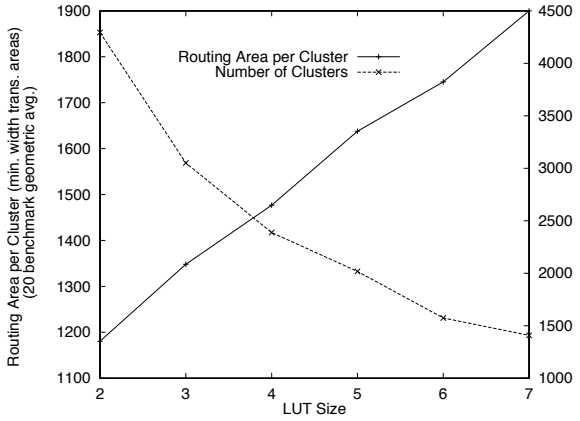
FIGURE 2.11: Etude de l'évolution de la surface des CLBs avec K et N [13].

La diminution du nombre de CLB à elle seule ne permet donc pas de contrebalancer la hausse de la surface des CLBs et n'incite pas à augmenter K. Cependant, elle permet de diminuer la quantité de ressources de routage global utilisée. L'évolution du routage global avec K est obtenue à l'aide du *CAD Flow* décrit dans la Section 2.1.3. La Figure 2.12 (b) issue de [13] montre une diminution de la surface de routage global due à la diminution de la demande en ressource de routage.

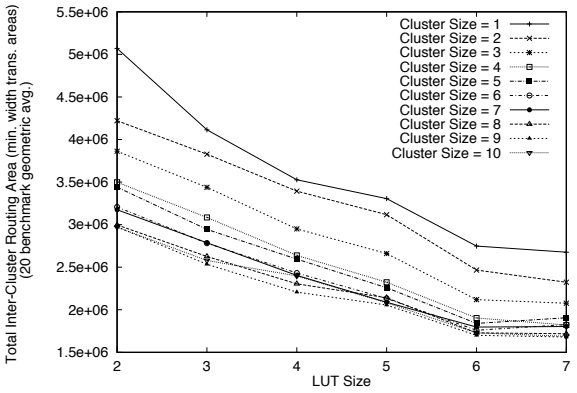
Les auteurs de [13] expliquent cette tendance en indiquant que le nombre de CLBs diminue plus rapidement que l'augmentation de la surface de routage global par tuile. Cette observation est confirmée par la Figure 2.12 (a) mais aucune analyse précise n'est proposée dans cet article.

Pour palier à cela, une courte analyse réalisée dans le cadre de ce mémoire est proposée. Deux phénomènes interviennent ici :

1. Le routage des circuits du *benchmark* exige de moins en moins de ressources en routage global à mesure que K augmente et donc le nombre de segments par canal de routage W, peut être diminué.
2. L'augmentation de K provoque l'augmentation de I pour maintenir l'accessibilité des CLBs. L'augmentation de la connectivité des CLBs complexifie la structure des CBs selon l'architecture de routage développée.



(a) Evolution de la surface de routage **global** par tuile et du nombre moyen de CLBs utilisés pour implémenter les circuits du *benchmark* pour $N = 1$ (s'applique aussi pour N plus grand).



(b) Evolution de la surface moyenne de routage global avec la taille des LUTs et des CLBs.

FIGURE 2.12: Etude de l'évolution de la surface de routage global avec K et N [13].

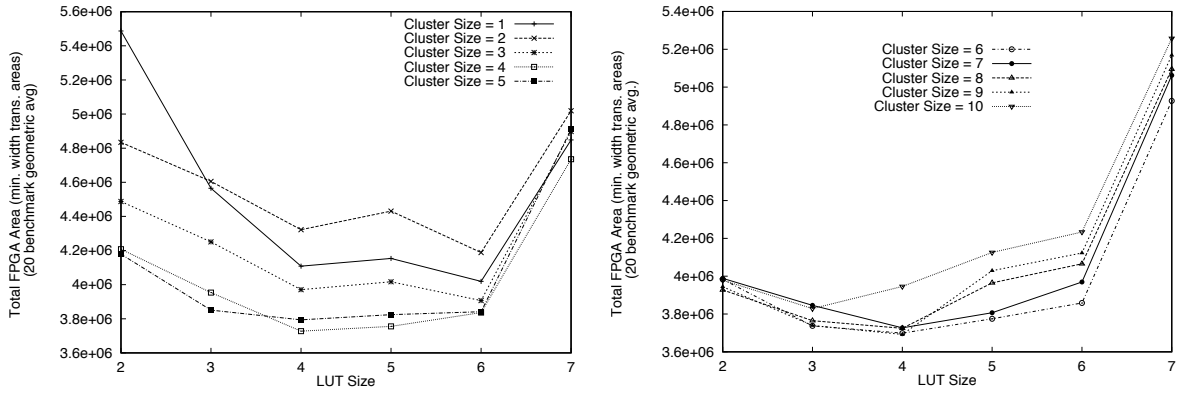
Le second effet, au travers de l'augmentation du nombre de multiplexeurs sélectionnant les signaux à imposer aux entrées des CLBs, cause l'augmentation de la surface de routage global observée sur la Figure 2.12 (a). Cette augmentation est linéaire avec K car le nombre d'entrées des CLBs évolue linéairement avec K (Section 3.2.2.1).

Si l'étude de l'évolution de la surface intra-CLB est peu dépendante de la qualité du *CAD Flow*, ce n'est pas le cas de la surface extra-CLB. En effet, un mauvais algorithme de routage peut multiplier le nombre de segments utilisés. Si le W minimal moyen sur l'ensemble du *benchmark* augmente, la surface moyenne de routage global présentée dans la Figure 2.12 (a) augmente aussi et le bénéfice que l'on retire à augmenter K, diminue. Ceci peut modifier la position de l'optimum simulé pour l'architecture. Il n'existe aucune étude sur ce point mais l'amélioration des performances des algorithmes de placement et routage tend à limiter leur impact sur ces optima.

En regroupant l'évolution croissante de la surface intra-CLB avec (K, N) et l'évolution décroissante de la surface extra-CLB avec (K, N), on peut observer un optimum. La Figure 2.13 permet de trouver la taille des LUT, optimale pour chaque N.

De ces résultats, la référence [13] extrait deux observations :

1. Si $N < 4$, K= 4, 5 ou 6 semble être un bon choix pour limiter la surface de la structure.
2. Si $N > 4$, la surface semble rester constante avec la taille des LUTs tant que $K < 6$.



(a) Evolution de la surface moyenne totale du FPGA nécéssaire pour implémenter les circuits du *benchmark* pour $N = 1 - 5$.

(b) Evolution de la surface moyenne totale du FPGA nécéssaire pour implémenter les circuits du *benchmark* pour $N = 6 - 10$.

FIGURE 2.13: Etude de l'évolution de la surface totale du FPGA avec K et N [13].

L'optimisation de la surface laisse donc une certaine latitude dans le choix de K et N ; cette latitude sera réduite en considérant d'autres facteurs de mérite comme la vitesse de la structure ou le produit entre vitesse et surface [3].

2.1.4.2 Optimisation de la vitesse

La vitesse d'un FPGA s'évalue en étudiant le délai moyen sur le chemin critique d'une série de circuits du *benchmark*. Le délai dans la structure est simulé par le logiciel VPR ([17]) qui implémente un modèle de délai comprenant les délais intra- et extra-CLB. Le détail du modèle de délai intra-CLB est repris dans la Figure 2.14. Une série de simulations physiques (SPICE) permet d'obtenir les délais entre les différents points du schéma.

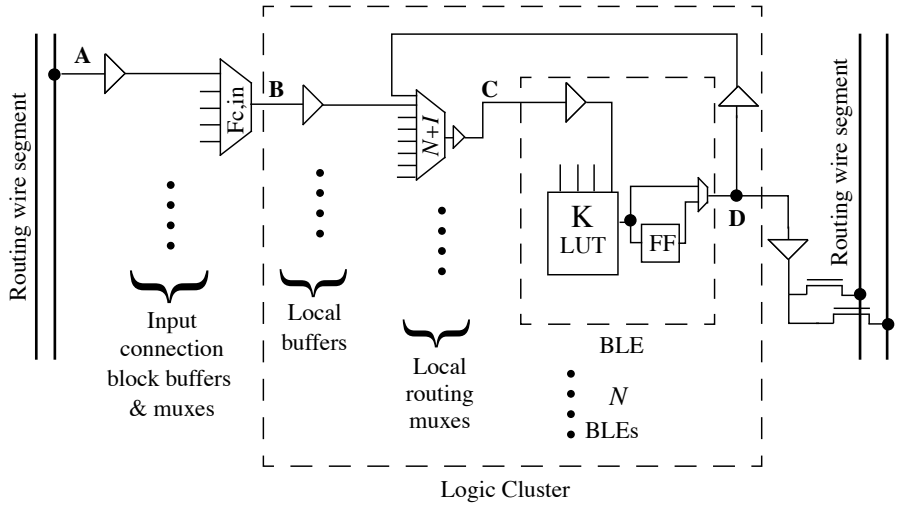


FIGURE 2.14: Détail du modèle de délai utilisé par VPR [17].

3. Les observations des optima sont liées aux détails d'implémentation comme la performance du routage global et la surface qu'il représente. Si le routage global est volumineux (par exemple à cause de l'utilisation de *tri-state buffers* ou parce que la connectivité des CB, SB est trop grande), la tendance à agrandir les grappes ou à augmenter la taille des LUTs sera plus forte.

La taille des LUTs influence le délai à travers le BLE. De plus, si le nombre d'entrées du CLB est adapté à la taille des LUTs pour maintenir l'accessibilité, K influence indirectement la taille et donc le délai des multiplexeurs de routage local. En revanche, le délai entre les points A et B (entrée du CLB) ne dépend pas de K mais seulement de la flexibilité du routage global. La taille de la grappe ne modifie pas non plus le délai entre les points A et B mais influence directement le délai à travers le routage local. Le délai dans le BLE ne varie pas avec N.

Le modèle du routage global est basé sur des simulations physiques des délais dans les *switches* et les *drivers* qui composent les parties actives des CB et SB. Si la connectivité de ces deux blocs augmente ou si le nombre d'entrées (I) ou de sorties (K) des CLBs augmente, le multiplexage du routage global se complexifie et les délais augmentent.

Le logiciel détermine le délai total sur chaque chemin entre une entrée et un registre, entre deux registres ou entre un registre et une sortie en additionnant les délais de tous les éléments présents sur le chemin. Le résultat est donc fortement lié au réalisme des simulations physiques réalisées. Il est important de reproduire le plus fidèlement possible les charges sur les segments par exemple. Le logiciel a été conçu pour le placement et le routage à haut niveau mais manque de précision à bas niveau. Ce comportement est lié à la difficulté que représente la simulation physique d'une structure composée de plusieurs dizaines ou centaines de tuiles. Pour y parvenir, comme cela est réalisé dans ce mémoire pour des petits circuits (cf. Chapitre 4), l'utilisateur doit apporter une grande quantité d'informations sur l'implémentation des différents blocs et sur la technologie utilisée.

L'optimisation du délai sur le chemin critique passe par le même type de compromis que l'optimisation de la surface totale du FPGA. L'augmentation de la taille des LUTs provoque une augmentation des délais à travers le CLB et, indirectement, à travers le routage global. Cependant, la diminution du nombre de BLEs sur le chemin critique diminue le nombre de ressources de routage global et local utilisé. La simulation de l'évolution du délai sur le chemin critique (Fig. 2.15) indique que l'augmentation de K et N provoque une amélioration des performances de la structure. La référence [13] propose une analyse de ce résultat en examinant l'évolution des délais intra-CLB, inter-CLB et des délais dans le BLE.

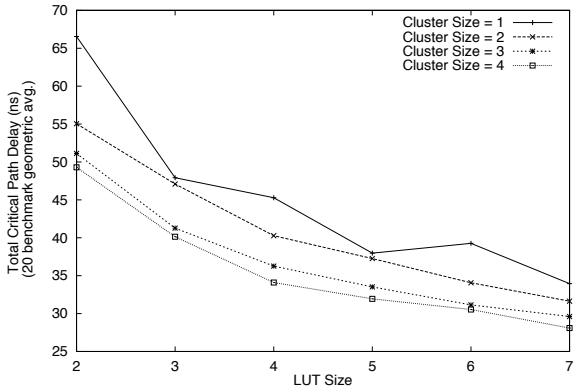
Tout comme pour l'optimisation de la surface du FPGA, les détails bas-niveau d'implémentation modifient la répartition du délai entre les différentes parties du chemin critique. Cependant, les modifications de proportion sont plus importantes dans le cas du délai. L'extension des conclusions de cette étape d'optimisation est donc soumise à réserve. Cette remarque est corroborée par la comparaison des résultats de [13] et de [19]. Ahmed et al ont réalisé leurs simulations physiques en technologie CMOS $0.35\mu m$ rendue obsolète par l'évolution des *process* modernes. Ils observent alors que le délai total à travers les BLEs ($4.8[ns]$ pour $K = 4$, $N = 4$) ne représente que 45% du délai total intra-CLB ($10.5[ns]$ pour $K = 4$, $N = 4$). Dans le cas de [19], sur une technologie moderne CMOS $65nm$, le rapport du délai des BLEs sur le délai total est proche de 70%.

Les conclusions de [13] indiquent que, étant donné la domination du délai de routage inter-CLB sur le délai intra-CLB, l'augmentation de la taille des LUTs et de la taille des grappes ne peut qu'améliorer les performances du dispositif. Cependant, l'écart considérable entre les deux types de délai observé en $0.35\mu m$ tend à se réduire lorsque la technologie s'approche de l'état de l'art. Cette conclusion n'est donc peut-être pas applicable pour un FPGA moderne dont la conception cible le domaine ULP comme c'est le cas dans ce mémoire.

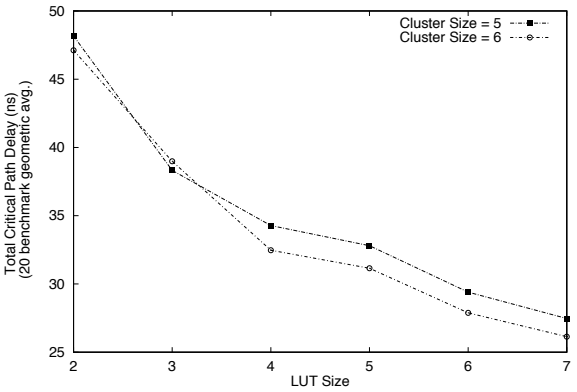
2.1.4.3 Optimisation de la puissance

La principale *driving force* qui pèse sur l'industrie du FPGA tend à rapprocher les performances en vitesse et en surface des FPGAs et des ASICs. En général, cette évolution se déroule au détriment de la puissance consommée qui atteint la dizaine de Watts pour les grosses structures d'Altera ou de Xillinx. Cette course à la performance combinée avec l'antagonisme entre vitesse et puissance consommée ont retardé le développement de FPGAs à faible consommation. A haut niveau, seul [20] étudie rigoureusement la répartition optimale de la logique et du routage vis à vis de la puissance.

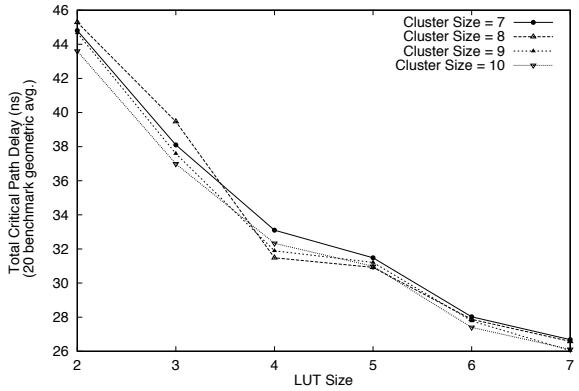
Pour réaliser cette optimisation, ils ont étendu le *CAD Flow* détaillé dans la Section 2.1.3 pour y insérer une mesure de la puissance consommée, évaluée par un modèle mixte. Ce modèle mélange des



(a) Evolution du délai moyen sur le chemin critique pour $N = 1 - 4$



(b) Evolution du délai moyen sur le chemin critique pour $N = 5 - 6$



(c) Evolution du délai moyen sur le chemin critique pour $N = 7 - 10$

FIGURE 2.15: Etude de l'évolution du délai moyen en fonction de la taille des LUTs [13].

estimations bas niveau (au niveau des transistors) pour les interconnexions et haut niveau (au niveau des LUTs) pour les BLEs.

Il y a 3 sources de consommation dans un FPGA comme pour tous les circuits CMOS digitaux :

1. Puissance de commutation : liée au chargement ou au déchargement des noeuds lors d'une transition.
2. Puissance de court-circuit : court-circuit entre GND et V_{DD} qui apparait lorsqu'il existe un chemin passant à travers le réseau PMOS et le réseau NMOS pendant une commutation. Ce type de consommation augmente si les transitions s'étendent sur une longue période car les grilles resteront longtemps à des tensions intermédiaires.
3. Puissance statique : consommée lorsqu'il n'y a aucune transition aux noeuds.

Les deux premiers types de consommation sont liés aux transitions qui peuvent être :

1. Fonctionnelles si elles sont nécessaires pour que la porte logique accomplisse sa fonction.
2. "Glitch" si elles sont provoquées par un déséquilibre des délais dans les chemins qui mènent aux entrées de la porte.

La caractérisation et l'estimation de ces consommations pour un circuit complexe peuvent être réalisées à différents niveaux. Si le calcul est réalisé au niveau du transistor, ce qui est utilisé par [20] pour modéliser la consommation dans les interconnexions, on modélise la puissance de commutation par :

$$P_{sw} = \frac{1}{2} f V_{DD}^2 \sum_{i=1}^{n} C_i \alpha_i \tag{2.2}$$

avec n, le nombre de noeuds du circuit, C_i, la capacité à chacun de ces noeuds, f, la fréquence de commutation et α_i, le facteur d'activité au noeud i. Les capacités peuvent être extraites après le *layout* pour annoter la *netlist* à postériori. Cette expression tient exclusivement compte des transitions dont l'amplitude correspond à la tension d'alimentation. Cependant, certains *glitches* n'auront pas une amplitude aussi grande et ne seront pas modélisés correctement. Pour corriger cette approximation, la référence [20] introduit un facteur d'activité effectif $\hat{\alpha}_i$.

La puissance de court-circuit, P_{sc}, est calculée sur base d'une simulation de la puissance dynamique totale ($P_{sw} + P_{sc}$) en fonction de la durée de la transition. La Figure 2.16 présente l'évolution de la puissance dynamique avec la durée de la transition pour un *buffer* de taille minimale dont la charge est variée. De cette facon, la référence [20] indique que la dépendance entre la durée de la transition et la puissance dynamique (qui ne peut venir que de la puissance de court-circuit par définition de celle-ci) est linéaire. Les auteurs la modélisent alors comme :

$$P_{sc} = \alpha_{sc} t_r P_{sw} \tag{2.3}$$

avec α_{sc}, obtenu en estimant la pente sur les courbes de la Figure 2.16.

Pour obtenir une estimation fiable de P_{sc}, il faut connaitre t_r, par exemple à 10-90%, qui est simulé pour différentes tailles de *buffers*. On mesure le délai dans le buffer, t_{buffer} (50-50%), et on mesure t_r (10-90%) pour obtenir α tel que : $t_r = \alpha t_{buffer}$. Ce modèle permet alors, sur base de α_{sc}, α et t_{buffer}, de calculer une estimation de P_{sc}.

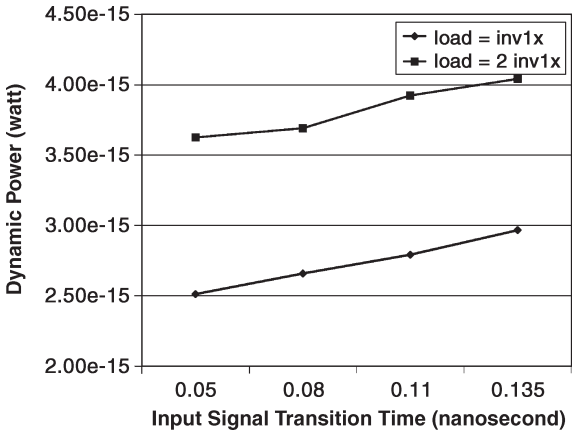


FIGURE 2.16: Evolution de la puissance dynamique consommée par un *buffer* dont la durée de la transition augmente et pour deux charges différentes. La dépendance de la puissance à la durée de transition est linéaire [20].

Le calcul de consommation au niveau des transistors nécessite donc beaucoup d'informations sur les capacités, les taux d'utilisation des noeuds ou encore sur la forme des transitions. Pour modéliser la consommation des LUTs, Li et al ont utilisé des *macromodels*. Il s'agit d'une table qui lie la puissance consommée obtenue par une série de simulations physiques en fonction des signaux d'entrée. Cette méthode de pré-calcul n'est réaliste que si le nombre de degrés de liberté n'est pas trop grand. Heureusement, les LUTs qui sont dans des CLBs sont toutes chargées de la même manière, ce qui limite le nombre d'entrées de la table qui contient toutes les mesures de puissance. Pour limiter la taille des tables de puissance, les simulations physiques sont réalisées en envoyant une série de vecteurs de tests

aléatoires et en calculant la puissance moyenne consommée par la LUT. L'hypothèse implicite utilisée par Li et al est que les probabilités d'occurrence des entrées sont égales.

Le modèle complet de la puissance comprenant le modèle bas-niveau pour les interconnexions et le *macromodel* pour les LUTs est alors inséré dans le *CAD Flow* comme illustré dans la Figure 2.17.

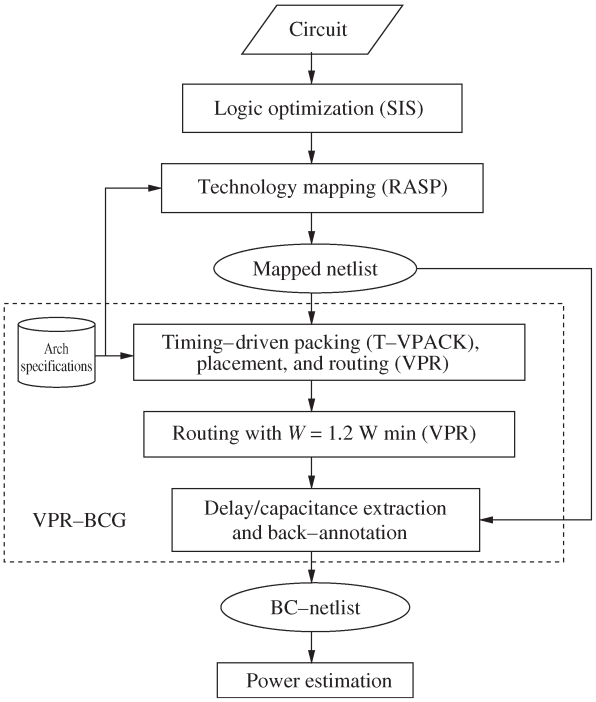


FIGURE 2.17: *CAD Flow* modifié pour intégrer l'évaluation de la puissance consommée par l'architecture. Deux étapes sont ajoutées par rapport au *CAD Flow* de la Section 2.1.3 : l'annotation à postériori des capacités en chaque noeud et l'évolution de la puissance totale suivant le modèle décrit dans les équations 2.2, 2.3 [20].

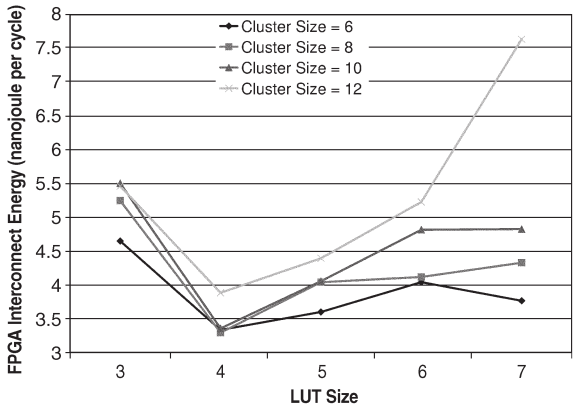
L'évolution de l'énergie moyenne consommée par les interconnexions est présentée sur la Figure 2.18 (a). Cette courbe présente un minimum qui est causé par deux phénomènes :

1. Si K augmente, les interconnexions augmentent pour maintenir l'accessibilité du CLB constante. Ce phénomène cause une augmentation de la consommation.
2. Si K augmente, il faut moins de LUTs et de CLBs pour implémenter les circuits du *benchmark*. Si le FPGA est plus petit, il y a moins de puissance dissipée dans les interconnexions.

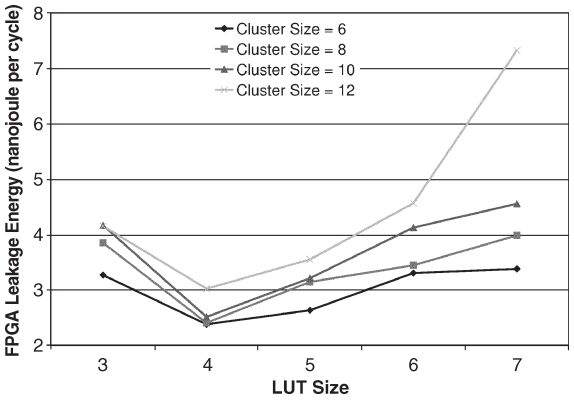
La référence [20] montre que, pour leur architecture de routage et pour leur technologie (CMOS $100nm$), l'optimum de consommation de l'interconnexion se situe à $K = 4$ et à $N < 12$.

La Figure 2.18 (b) montre que l'évolution de la puissance statique suit les mêmes tendances car elle est directement liée à la surface couverte par les interconnexions.

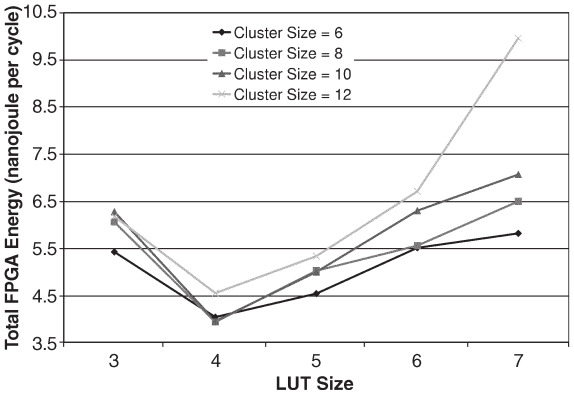
La Figure 2.18 (c) présente le même optimum pour la puissance totale consommée. Li et al ajoutent que, par hasard, l'optimisation de la surface conduit aux mêmes conclusions que l'optimisation de la puissance. Cependant, le modèle de la puissance est fortement lié aux détails d'implémentations et aux détails technologiques. Pour pouvoir comparer les résultats de [13] et de [20], il faut reproduire les simulations sur un même *process* et sur une même architecture, ce qui ne semble pas avoir été réalisé dans [20].



(a) Evolution de l'énergie dynamique consommée dans le routage global en fonction de K et N.



(b) Evolution de l'énergie statique consommée dans le FPGA en fonction de K et N.



(c) Evolution de l'énergie totale consommée dans le FPGA en fonction de K et N.

FIGURE 2.18: Evolution de l'énergie consommée en fonction de la taille des LUTs [20].

2.1.5 Implémentation de la configurabilité

La versatilité des FPGAs repose sur la grande quantité de bits de configuration disséminés dans la structure. Leur implémentation joue un rôle indirect dans l'évaluation des performances de la structure en ajoutant des contraintes sur les technologies utilisables pour le reste des blocs.

2.1.5.1 SRAM

La vaste majorité des FPGAs modernes implémentent chaque bit de configuration avec une cellule de SRAM.

Avantages :

1. Les cellules de SRAM permettent une reprogrammation simple et rapide à l'image d'une écriture dans une mémoire statique.
2. Leur implémentation en technologie CMOS ne nécessite aucune modification au *process* de fabrication et permet l'utilisation des noeuds technologiques nanométriques de l'état de l'art.

C'est essentiellement la possibilité d'utiliser les technologies les plus performantes qui pousse les industries à développer ce type de FPGA. Il est en effet nécessaire de suivre la descente des noeuds technologiques utilisés pour la conception d'ASICs pour éviter que l'écart entre la logique reprogrammable et dédicacée n'augmente pas.

Inconvénients :

1. Elles nécessitent 6 transistors (5 si on retire une des *bitline* qui n'est pas nécessaire en lecture continue) ou davantage si on veut garantir la stabilité lorsque la tension diminue (cf. cellules 7T, 8T, 12T présentées dans [2], [3], [48]).
2. Un FPGA qui contient sa configuration dans ce type de structure est volatile et nécessite une reprogrammation à chaque mise sous tension. Cette caractéristique est généralement contournée par l'ajout d'une mémoire non-volatile *off-chip* pour mémoriser et rapatrier la configuration à chaque démarrage.
3. Si la configuration est extérieure et doit être chargée à chaque démarrage, elle est vulnérable et peut être extraite ou clonée afin de servir dans un système concurrent. Certaines familles de FPGA intègrent un chiffrement de la configuration, parmi eux, le Virtex 5 de Xilinx ([29]) ou le Stratix III d'Altera ([30]) implémentent un chiffrement AES sur 256 bits.

2.1.5.2 Flash/EEPROM

Les transistors Flashs sont composés de deux grilles [8]. La grille supérieure permet la programmation en chargeant ou en déchargeant par effet tunnel la grille flottante inférieure (cf. Fig. 2.19 (a)).

Lorsque la grille flottante n'est pas chargée, ce qui correspond à l'état "non-programmé" de la cellule, le dispositif est caractérisé par une tension de seuil V_t faible. Selon l'implémentation dans laquelle elle se trouve, la cellule non-programmée peut contenir le '1' ou le '0'.

La programmation du transistor s'effectue en appliquant une tension élevée entre la source et le drain et sur la grille de sélection. Si la grille flottante n'est pas chargée, un canal est formé sous la grille et la tension V_{DS} accélère les électrons dont l'énergie cinétique augmente (électrons "chauds"). Le champ électrique dans l'oxyde provoqué par la tension entre le *bulk* et la grille de sélection attire les électrons chauds jusqu'à la grille flottante qui se charge.

La présence de charges négatives sur la grille flottante augmente la tension de seuil et décale la courbe $i_D - v_{GS}$ de la cellule programmée.

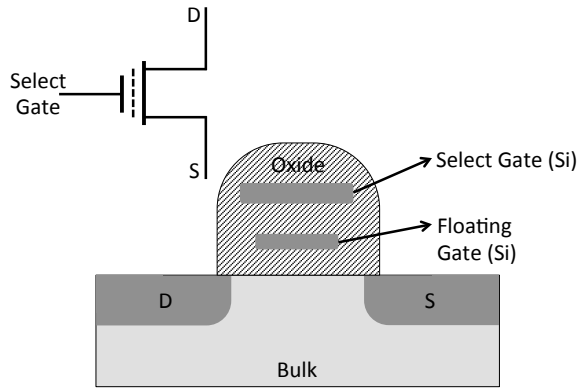
Dans un FPGA, les transistors à double grille sont utilisés en paire comme indiqué sur Fig. 2.19 (b). Les transistors de programmation sont accessibles par le circuit de configuration qui fixe les tensions de source, drain et de grille de sélection pour programmer la grille flottante. Comme cette grille est partagée avec le transistor de passage qui appartient aux LUTs ou aux multiplexeurs de routage, la programmation est propagée aux interrupteurs du FPGA. La cellule décrite à la Fig. 2.19 (b) a été développée dans [4] pour des FPGAs destinés aux applications spatiales pour sa bonne résistance aux impacts de particules ionisantes.

Avantages :

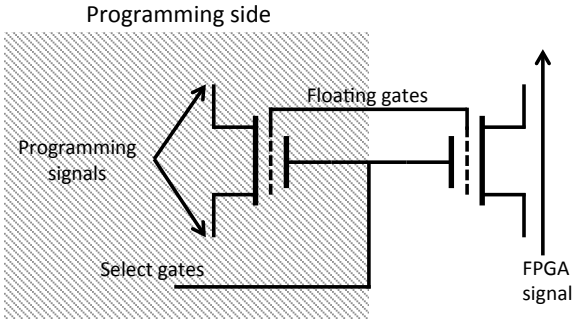
1. La technologie Flash contourne la volatilité des cellules de SRAM et réduit la consommation statique en diminuant le nombre de transistors par bit de configuration.
2. La problématique de la sécurité de la configuration disparait avec la volatilité.
3. La présence des données de configuration dans la structure pendant les périodes durant lesquelles l'alimentation est déconnectée permet de démarrer le circuit immédiatement après la mise sous tension sans devoir passer par la configuration.

La surface des cellules de Flash est inférieure à la surface des cellules de SRAM 6T mais la structure de programmation, plus complexe à cause des tensions à appliquer pour programmer les cellules Flash, diminue l'intérêt de ces cellules. Cependant, si le FPGA est suffisamment grand, la pénalité de surface imposée par le circuit de configuration Flash est amortie par le nombre important de cellules programmables.

Inconvénients :



(a) Schéma d'un transistor FLASH comprenant la grille de sélection et la grille flottante.



(b) Implémentation typique d'un bit de configuration dans un FPGA non-volatile basé sur une mémorisation FLASH de sa configuration [4].

FIGURE 2.19: Technologie FLASH et intégration pour FPGA.

1. La propriété de reprogrammation des cellules Flash est limitée en nombre de cycles à cause de l'accumulation de charges dans l'oxyde qui finit par empêcher la réinitialisation de la cellule. Cependant, dans certaines applications, le FPGA ne doit être initialisé qu'une seule fois.
2. La présence de cellules Flash sur la puce impose l'utilisation de *process* CMOS non-standard.

Certains circuits, comme le MAX II (Altera - CPLD à faible coût [31]), combinent des cellules Flash et des cellules SRAM sur la même puce. Les cellules SRAM contrôlent les éléments programmables alors que les cellules de Flash permettent d'enregistrer la configuration pour pouvoir charger rapidement le FPGA ou le CPLD au démarrage et sans risque de sécurité car les deux types de mémoire coexistent sur la même puce. L'architecture de ces types de FPGA/CPLD est identique aux architectures de FPGA basées sur des SRAM développés dans les Chapitres 3 et 4 mais l'introduction de cellules Flash impose une modification du *process* au dessus de la couche de CMOS traditionnelle et généralement l'utilisation de noeuds technologiques moins performants que ceux qui sont utilisés pour les circuits basés uniquement sur des cellules SRAM.

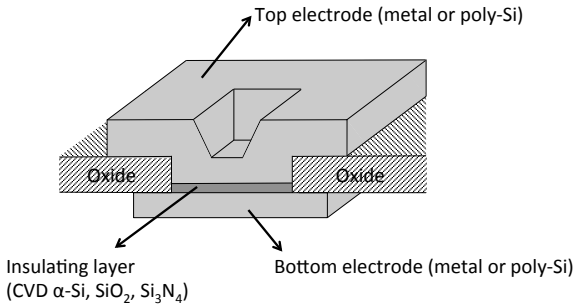
2.1.5.3 Technologie *anti-fuse*

La technologie *anti-fuse* est basée sur des fusibles programmables. Il s'agit de dispositifs qui présentent une haute impédance dans l'état initial et une basse impédance dans l'état programmé. La Figure 2.20 (a) présente la structure d'un fusible. Deux électrodes sont séparées par une couche d'isolation qui peut être rendue conductrice par l'application d'une tension élevée. Les électrodes sont historiquement constituées de polysilicium ou, dans les dispositifs modernes, de métal. L'ingénierie de la couche d'isolation a fait l'objet de recherche extensive entre 1970 et 1990 ([7], [6]). Dans les *antifuse* modernes, cette couche est réalisée en oxyde de silicium, en nitrure de silicium ou en silicium amorphe déposé par *chemical vapor deposition* CVD.

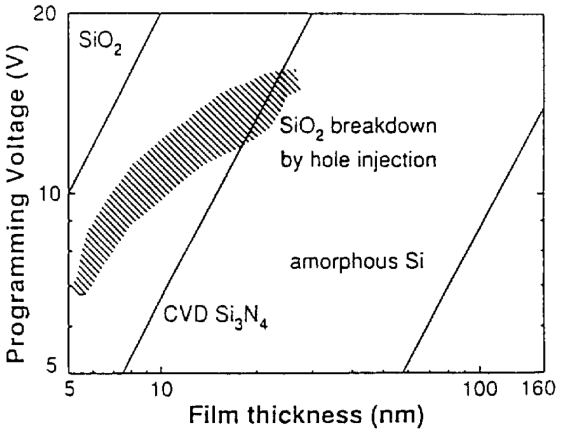
La Figure 2.20 (b) présente l'évolution de la tension de programmation de la jonction en fonction de l'épaisseur de la couche d'isolation ([5]). Le gain atteint par une diminution de l'épaisseur de la couche d'isolation (pour diminuer la tension de programmation) doit être contrebalancé par l'évolution de la durée de vie des *antifuses* non-programmés qui diminue avec l'épaisseur. La référence [6] préconise une tension de programmation $2,5$ fois supérieure à la tension d'alimentation du dispositif.

Dans un FPGA, ces structures permettent d'isoler toutes les connections entre blocs logiques et routage ou entre deux segments. L'utilisateur sélectionne alors les *antifuses* qu'il désire fermer. La Figure 2.20 (c) montre comment les *antifuses* s'insèrent entre les pistes de métal au niveau des *vias* entre les couches de Metal 3 et 4 pour les SX-A d'Actel. Les *antifuses* modernes d'Actel conçus dans un *process* CMOS $0.22\mu m$ dont la couche d'isolation composée de silicium amorphe et de diélectrique

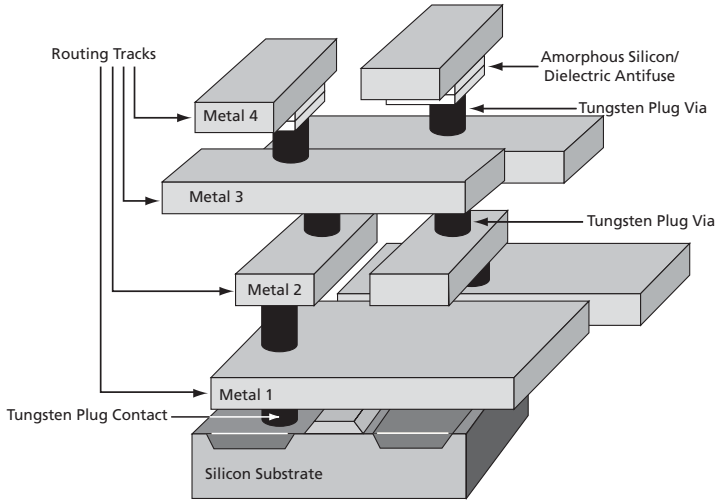
permettent d'atteindre une résistance en état programmé de 25 Ω et une capacité dans le même état de $1fF$ ([9]).



(a) Schéma d'un *antifuse*.



(b) Evolution de la tension de programmation d'un *antifuse* pour une impulsion de $1ms$ en fonction de la composition de la couche d'isolation et de son épaisseur [5].



(c) Schéma des niveaux d'interconnection dans un FPGA de la famille SX-A de Actel ([9]). Les *antifuses* sont situés entre les couches de métal 3 et 4 ([5]).

FIGURE 2.20: Technologie *antifuse* et intégration pour FPGA.

Avantages :

1. Les *antifuses* ont une très faible empreinte sur la surface du FPGA qui avoisine celle d'un *via*, ce qui permet de diminuer la pénalité en surface que les FPGAs concèdent sur les ASICs. Même en prenant en compte la surface des gros transistors qui génèrent les grands courants de programmation, ce type de dispositif est significativement plus économe en surface que les FPGAs basés sur des SRAMs ou sur des Flash.
2. Les *antifuses* ont une faible résistance et une faible capacité parasite lorsqu'ils sont passants.
3. Ils permettent d'obtenir la non-volatilité, ce qui évite l'étape de configuration au démarrage ainsi que les coûts liés aux circuits de programmation.

4. Il est impossible de faire du *reverse engineering* sur ce type de structure car un *antifuse* programmé n'est pas reconnaissable d'un *antifuse* non-programmé ([5]). De plus, comme la configuration n'est pas transférée à la puce, la sécurité est garantie au démarrage.

Inconvénients :

1. Les *antifuses* imposent la modification du *process*, ce qui contraint à l'utilisation d'un noeud technologique moins performant que les noeuds utilisés pour les FPGAs équipés de SRAMs.
2. La technique de programmation des *antifuses*, de part son mécanisme, ne se prête pas bien à une diminution des dimensions et est limité à des noeuds technologiques supérieurs aux technologies $0.15\mu m$. En effet, plus la largeur et l'épaisseur des *antifuses* diminuent, plus il est difficile de contrôler précisément la tension de programmation et le rendement de fabrication. De plus, la durée de vie des *antifuses* non-programmés diminue avec la largeur de la couche d'isolation. Actel n'a pas encore pu descendre en dessous de $0.15\mu m$ même pour ses plus récents FPGAs de la famille *Axcelerator* (2012).
3. Les *antifuses* ne peuvent être programmés qu'une seule fois. Ces FPGAs conservent cependant un court temps entre la conception et l'arrivée sur le marché comme avantage sur les ASICs. De plus, ils permettent d'amortir le coût des masques en produisant le même *die* pour différentes applications à faible volume de production pour lesquelles un ASIC aurait été trop coûteux.
4. Le rendement de fabrication des FPGAs implémentant ces technologies est significativement inférieur à celui des FPGAs basés sur des SRAMs ou sur des FLASHs. C'est l'impossibilité de tester le bon fonctionnement de tous les *antifuses* qui en est la cause.

2.2 Conception de circuits digitaux ULP

Les circuits digitaux ULP sont caractérisés par de faibles contraintes de vitesse mais de fortes contraintes sur :
- la puissance instantanée pour les systèmes dont la source de puissance est limitée dans le temps (ex. *tags* RFID)
- l'énergie par opération pour les systèmes à batterie (ex. implants biomédicaux ou réseaux de senseurs)

Différentes techniques sont disponibles pour diminuer la puissance consommée en agissant sur le compromis consommation/performance.

2.2.1 Analyse des sources de consommation

Comme énoncé dans la Section 2.1.4.3, il y a deux types de consommation dans un circuit CMOS : la consommation dynamique et la consommation statique. La puissance dynamique est elle même la somme d'une consommation de court-circuit et d'une consommation de transition que [32] regroupe dans l'expression suivante :

$$P_{dyn} = P_{sw} + P_{sc} = \frac{1}{2} N_{nodes} \alpha_F (1 + \beta_{sc}) C_L V_{DD}^2 f_{clk} \tag{2.4}$$

avec :
- α_F, le facteur d'activité qui modélise la densité de transition
- N_{nodes}, le nombre de noeuds du circuit
- C_L, la capacité moyenne des noeuds
- β_{sc}, un facteur d'échelle permettant de comparer P_{sc} et P_{dyn}

La référence [32] indique que, tant que le délai des portes du circuit est petit, la puissance de court-circuit sera limitée (5-10% de P_{sw}).

La puissance statique d'un circuit CMOS est due aux courants de fuite qui traversent les transistors. La modélisation des courants de fuite est complexe dans les dispositifs nanométriques et tient compte d'un grand nombre de phénomènes. La composante principale provient du courant traversant le canal

lorsque le dispositif est sous-seuil. Lorsque $V_G < V_T$, la concentration d'électrons dans le canal est inférieure à la concentration en porteur majoritaire dans le reste du *bulk*. La charge totale dans le *bulk* Q_{bulk} est composée d'une charge de déplétion Q_D et d'une charge d'électrons Q_N largement inférieure ([33]).

La charge de déplétion correspond au produit entre la densité de dopants et la surface de la zone de déplétion. Celle-ci, dans une structure MOS dépend de la chute de potentiel à travers le semiconducteur Φ_S :

$$Q_D = -qN_A x_D = -\sqrt{2q\epsilon_{Si}N_A\Phi_S} \tag{2.5}$$

avec :
- N_A, le dopage du *bulk*
- x_D, la largeur de la zone de déplétion
- Φ_S, le potentiel de surface qui dépend principalement de V_G et de la tension de *flatband*

La charge totale dans le *bulk* s'écrit, par la loi de Gauss :

$$Q_{bulk} = \epsilon_s E_0 = -\gamma C_{OX}\sqrt{\frac{kT}{q}(\frac{n_i}{N_A})^2 e^{\frac{q(\Phi_S - V)}{kT}} + \Phi_S} \tag{2.6}$$

avec :
- C_{OX}, la capacité d'oxyde
- n_i, la concentration intrinsèque du silicium
- V, la tension locale en un point du canal
- $\gamma = \frac{\sqrt{2q\epsilon_{Si}N_A}}{C_{OX}}$

Par définition, on a $Q_{bulk} = Q_D + Q_N$. Le courant sous-seuil s'écrit donc :

$$I_D = -\frac{W}{L}\mu_{eff}\int_{V_S}^{V_D} Q_N dV \tag{2.7}$$

$$= \frac{1}{2}(\frac{kT}{q})^2\frac{W}{L}\mu_{eff}\gamma C_{OX}(\frac{n_i}{N_A})^2\frac{exp(\frac{q\Phi_S}{kT})}{\sqrt{\Phi_S}}exp(-\frac{qV_S}{kT})[1 - exp(-\frac{qV_{DS}}{kT})] \tag{2.8}$$

Le courant sous-seuil dépend principalement de la tension de grille et est presque indépendant de la tension de drain car si $V_{DS} >> \frac{kT}{q}$, le dernier terme de l'Equation 2.8 est proche de 1 (cf. Fig. 2.21).

Le courant sous-seuil est couramment caractérisé par une pente sous-seuil $S = \frac{dV_G}{dlog(I_D)}$. L'application de cette définition sur l'Equation 2.8 donne :

$$S \cong \frac{kT}{q}ln(10)(1 + \frac{C_D + C_{it}}{C_{OX}}) \tag{2.9}$$

avec :
- C_{OX}, la capacité d'oxyde
- C_D, la capacité de la zone de déplétion
- C_{it}, la capacité modélisant les charges contenues dans les états de surface à l'interface entre le silicium et l'oxyde

Cette pente, illustrée dans la figure 2.21, détermine le rapport I_{ON}/I_{OFF} du dispositif. Une faible pente sous-seuil permet donc d'augmenter le rapport I_{ON}/I_{OFF}.

La réduction de la longueur du canal provoque deux effets néfastes à la tension de seuil. Lorsque le canal est long, la surface de la zone de déplétion contrôlée par la tension de grille est largement supérieure aux effets de bords provoqués par les tensions de source et drain. Etant donné que la tension de seuil dépend directement de la charge présente dans la zone de déplétion de grille, cet effet de partage de charge provoque une diminution de V_T. De la même façon, lorsque V_{DS} augmente, la largeur de la zone de déplétion contrôlée par la tension de drain augmente, réduisant la surface de la zone de déplétion

contrôlée par la tension de grille. Etant donné la dépendance de la tension de seuil envers la charge de déplétion contrôlée par la grille, ce contrôle de charge par le drain provoque une diminution de V_T (*charge sharing*). Ce phénomène porte le nom de DIBL pour *Drain Induced Barrier Lowering*. Plusieurs techniques permettent de remonter V_T face à ces effets canaux-courts ; il est possible d'augmenter le dopage du *bulk* pour rétracter les zones de déplétion mais la mobilité du canal diminue. Il est également possible de modifier la structure du MOS en implantant des HALOs ([33]). Un faible DIBL permet de réduire le courant de fuite I_{OFF}.

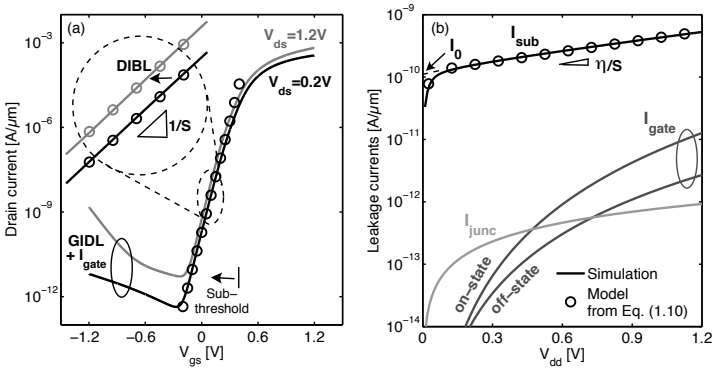


FIGURE 2.21: A gauche : évolution du courant de drain avec la tension de grille. La pente sous-seuil et le DIBL sont mis en évidence. A droite : évolution du courant de drain à $V_{GS} = 0[V]$ en fonction de V_{DD}. Le courant sous-seuil I_{sub} est dominant et présente une petite inflexion lorsque V_{DS} se rapproche de kT/q comme prédit par l'Equation 2.8. Les autres sources de fuite sont également présentées bien que leur analyse dépasse le cadre de ce travail [32].

2.2.2 ULV - alimentation sous-seuil

La réduction de la tension d'alimentation permet de limiter la puissance dynamique du circuit mais le courant de fuite reste approximativement constant avec V_{DD}. La figure 2.22 (a) montre l'évolution de la puissance dynamique simulée en fonction de V_{DD} comme démontré dans la Section 2.2.1. On constate sur la même figure que la puissance statique (produit de V_{DD} et du courant de fuite) décroit elle aussi avec V_{DD}. Une application ciblant une réduction de la puissance consommée devra donc cibler le plus petit V_{DD} possible pour rencontrer les contraintes de vitesse. Dans cette optique, les circuits opérants sous-seuil font une importante économie d'énergie (environ un ordre de grandeur).

La réduction du courant I_{ON} provoque une augmentation des délais dans les portes logiques car les capacités à charger restent constantes. La variation de la durée des opérations demande l'étude d'un second facteur de mérite : l'énergie par opération définie comme l'intégrale de la puissance totale consommée sur la période de *clock*. La combinaison de l'augmentation de la fréquence de *clock* minimum et de la diminution de la puissance totale consommée donne lieu à un optimum illustré dans la figure 2.22 (b). La référence [34] indique que ce point d'énergie minimum est typiquement situé sous le seuil.

La relative indépendance du courant de fuite avec V_{DD} provoque une réduction du rapport I_{ON}/I_{OFF} qui exacerbe la sensibilité du circuit aux variations de *process* ainsi qu'aux effets canaux-courts. En effet, l'impact des variations des V_T à travers le circuit est amplifié par la dépendance exponentielle du courant envers $V_{DD} - V_T$ ([34]). Ces phénomènes rendent difficile le bon fonctionnement du circuit au point d'énergie minimum.

2.2.3 FVS - *Frequency Voltage Scaling*

En étant conscient de la chute de performance induite par la réduction de V_{DD}, on peut précisément, sur base d'une contrainte de débit, déterminer le V_{DD} optimal. A l'inverse, si l'objectif est le point d'énergie minimal, on peut sélectionner le débit et le V_{DD} pour y arriver.

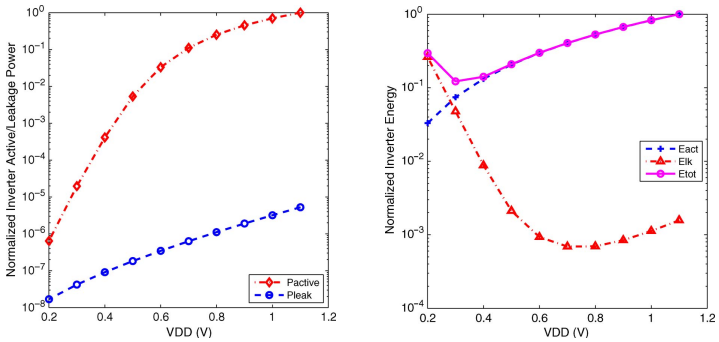


FIGURE 2.22: A gauche : évolution de la puissance dynamique et statique d'un inverseur en fonction de V_{DD}. A droite : évolution de l'énergie en fonction de V_{DD} [34].

La figure 2.23 reprend cette idée de sélection de la fréquence et de V_{DD}. La diminution de la fréquence d'opération permet de diminuer la tension minimale à appliquer au circuit. Deux limites sont observées : à haute fréquence, il est évident que la saturation du courant avec V_{DD} limitera la fréquence maximale et si V_{DD} devient trop faible, ce sont les contraintes de robustesse qui ne sont plus satisfaites [4] (cfr. Section 2.2.4).

Au niveau de la puissance, lorsque f_{clk} et V_{DD} sont élevés, P_{dyn} domine P_{stat} ; la puissance consommée est utilisée pour charger les noeuds et est donc utilisée efficacement. En revanche, plus f_{clk} et V_{DD} diminuent, plus P_{dyn} diminue et c'est la puissance de fuite qui domine.

L'observation de l'énergie indique que le point d'énergie minimum se situe dans la zone où P_{dyn} domine P_{stat}.

2.2.4 Contrainte de robustesse fonctionnelle

La marge de bruit (*Signal to Noise Margin* ou SNM) d'une porte logique est définie comme le niveau de bruit maximum que l'on peut ajouter aux niveaux logiques des entrées de la porte sans que son/ses sorties(s) ne change(nt). L'opération proche du seuil provoque une diminution des marges de bruit ([32]), ce qui exacerbe la sensibilité du circuit envers le *crosstalk*, les impacts de particules ionisantes, la variabilité ou tous les phénomènes ajoutant du bruit sur les niveaux logiques.

L'estimation rapide et précise des SNMs sur un circuit sous-seuil complexe représente un challenge technologique important. Bien que certains groupes de recherche ([35]) obtiennent de bons résultats, il n'existe pas encore d'outils CAD qui évaluent rapidement les SNMs d'une *netlist* quelconque. La référence [36] propose d'étudier les SNMs en bouclant une porte NAND sur une porte NOR, ce qui représente le pire cas possible (cf. Fig. 2.24).

L'évolution des marges de bruit et l'augmentation de la variabilité constatée proche ou sous le seuil dégradent donc le rendement de fabrication de ces circuits. Les simulations réalisées sur base du *testbench* de la figure 2.24 permettent de définir et simuler un rendement limité par la préservation de la fonctionnalité des portes ([32]). Ce rendement est défini, pour chaque V_{DD}, comme le rapport du nombre de *dies* exhibant une SNM positive par rapport au nombre de *dies* total. Une SNM négative est mesurée si le niveau logique bas qui sort de la porte A est supérieur au plus petit niveau logique bas reconnu par la porte B. La sortie de la porte A sera donc un niveau haut et le circuit ne fonctionnera plus correctement.

Cette contrainte de robustesse est plus contraignante lorsque la technologie s'approche des noeuds de l'état de l'art. La figure 2.23 montre que, pour une technologie $0.13\mu m$, cette limite est atteinte pour $V_{DD} \cong 0.2V$.

4. Le V_{DD} minimal ou le f_{clk} maximal doivent être ajustés pour garder une sécurité face à la variabilité.

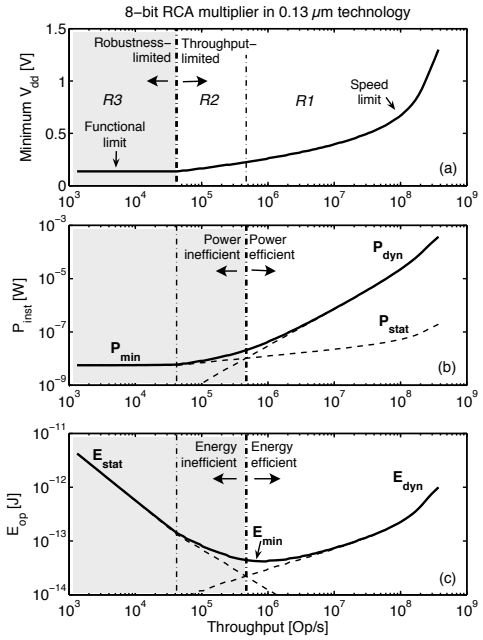


FIGURE 2.23: En haut : évolution du V_{DD} minimum pour atteindre une contrainte de performance. Au milieu : évolution de la puissance consommée en fonction de la contrainte de performance en adaptant le V_{DD} selon la courbe du haut. En bas : évolution de l'énergie par opérations en fonction de la contrainte de performance en adaptant le V_{DD} selon la courbe du haut. Les simulations sont réalisées sur un multiplieur 8 bit en technologie $0.13\mu m$ [32].

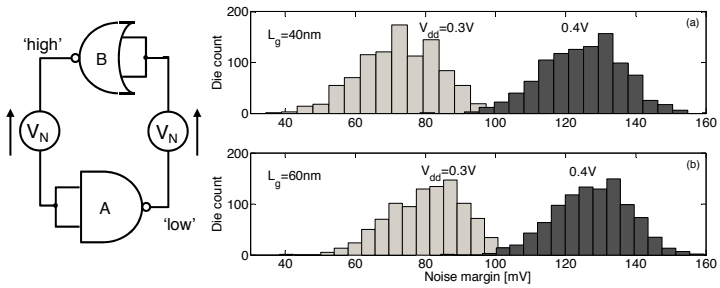


FIGURE 2.24: A gauche : schéma utilisé pour l'évaluation des SNMs pour la technologie $65nm$. A droite : histogramme de répartition des SNMs évaluées sur 1000 simulations Monte-Carlo pour deux longueurs de grille. On constate que l'allongement des grilles ne limite que peu la variance de la distribution des SNMs tout en imposant une pénalité en vitesse car les capacités de grille augmentent [36].

2.2.5 *Power Gating* de circuits sous-seuil

Certains circuits ULP passent régulièrement de périodes d'activité à des périodes de repos. Durant ces périodes de repos, la puissance dynamique est nulle mais la consommation statique continue d'amputer l'autonomie de ces circuits si ils opèrent sur batterie. L'insertion de transistors de *power gating* entre l'alimentation et le circuit permet de réduire les courants de fuite en imposant un V_{DD} fictif très faible au circuit (cf. Fig. 2.25 (a)).

L'insertion de ce transistor n'est pas anodine. En effet, lorsque le circuit est actif, le transistor ajoute une résistance en série avec le rail d'alimentation. Cette résistance induit deux effets :

– Elle réduit le courant I_{ON} qui sert à la charge des noeuds de sortie des portes logiques handicapant la vitesse du circuit.

– Elle réduit les marges de bruit des portes logiques car elle induit un bruit sur le rail d'alimentation dépendant du courant consommé. Cette réduction des SNMs peut poser un problème de rendement de fabrication par rapport à la contrainte de robustesse fonctionnelle (Section 2.2.4).

La figure 2.25 (b) illustre la dégradation des SNMs et des délais, causée par l'insertion du transistor de *power gating*. On constate également que plus la largeur du transistor est faible, plus son impact sera important car sa résistance parasite sera importante. En dessous d'une largeur normalisée de 0.1, les marges de bruits sont trop dégradées pour assurer le bon fonctionnement du circuit (technologie $45nm$).

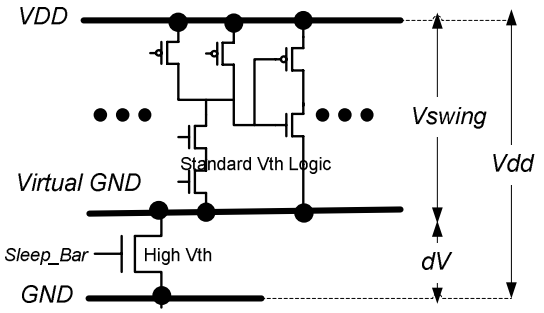
La référence [37] introduit deux facteurs de mérite qui mesurent les performances du *power gating* :

$$\frac{1}{K_{delay}} = \frac{t_{delay\_w\_PGS}}{t_{delay\_w/o\_PGS}} \tag{2.10}$$

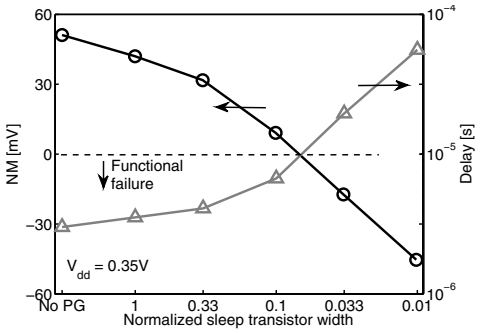
$$K_{leak} = \frac{I_{leak\_w\_PGS}}{I_{leak\_w/o\_PGS}} \tag{2.11}$$

La référence [38] prend en plus en compte la dégradation des marges de bruit au travers de K_{NM}, le rapport entre la marge de bruit (évaluée comme dans la figure 2.24) simulée sans *power gating* et la marge de bruit avec *power gating*.

Le dimensionnement du transistor de *power gating* est soumis à un compromis entre une bonne réduction du courant de fuite lorsque le circuit est coupé et une faible pénalité sur les délais et les marges de bruits lorsque le circuit est alimenté.



(a) Schéma de l'implantation d'un transistor de *power gating* ([37]).



(b) Impact de la largeur du transistor de *power gating* sur les délais et les marges de bruits ([36]). La largeur du transistor est normalisée par rapport à la somme des largeurs des NMOS présents dans le circuit.

FIGURE 2.25: *power gating*.

Au delà de la largeur du transistor de *power gating*, d'autres paramètres jouent un rôle important dans le dimensionnement. Citons les principaux : le V_T, la longueur de grille ou la tension de grille ([36]). Un transistor à haut V_T offrira un meilleur K_{leak} mais dégradera davantage K_{delay} et K_{NM} par rapport à un transistor de V_T standard.

Un transistor dont la longueur de grille a été agrandie possèdera une pente sous-seuil plus raide (car il est moins sensible au *charge sharing*) et donc un rapport I_{ON}/I_{OFF} plus élevé. Il coupera mieux le courant de fuite qu'un transistor plus court. De la même façon, le DIBL diminue avec l'augmentation de la longueur de grille car la proportion de zone de déplétion contrôlée par la grille augmente. La diminution du DIBL joue un rôle important lorsque le circuit est éteint car le transistor est alors soumis à un V_{DS} important. Un faible DIBL limite l'augmentation du I_{OFF} causée par ce V_{DS}.

Pour finir, l'augmentation de la tension de grille du transistor permet de diminuer la résistance en mode actif, ce qui améliore K_{delay} et K_{NM}. Ces trois effets sont illustrés dans la figure 2.26.

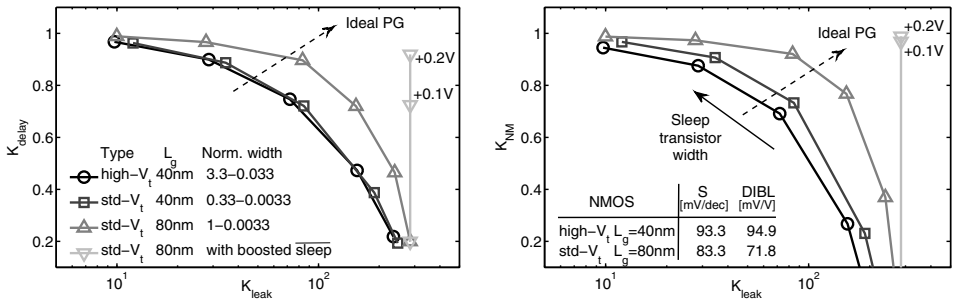


FIGURE 2.26: A gauche : évolution de K_{delay} pour différentes largeurs de *power gating*. A droite : évolution de K_{NM} pour différentes largeurs de *power gating*. On constate que le transistor de V_T standard dont la grille est allongée, est le plus performant. De plus, on constate que l'augmentation de la tension de grille permet de ramener K_{delay} et K_{NM} à des valeurs raisonnables tout en profitant d'un K_{leak} bas ([36]).

2.2.6 Multi-Vt, Multi-*process flavors*

Certaines technologies avancées offrent la possibilité de faire varier certains paramètres des MOS. En plus de la longueur de grille, on peut alors modifier la tension de seuil ou l'épaisseur d'oxyde. La variation de ces paramètres permet au concepteur d'ajuster le plus précisément possible le compromis vitesse/consommation pour les différents blocs du système ; ce qui peut se révéler crucial lors de la conception de circuits ULP. Pour savoir comment répartir les différentes versions des MOS sur le *design*, il faut savoir comment ces paramètres influencent les performances des transistors.

L'influence de L_g, V_T et t_{OX} est caractérisée, entre autre, par trois facteurs de mérites importants : la pente sous-seuil S, le DIBL et le produit $C_L S^2$. Ce dernier facteur tient en compte l'évolution de S en même temps que celle de la capacité de charge C_L. Il s'agit là d'une expression du compromis vitesse/consommation. L'étude de l'évolution de ce facteur de mérite et de son influence sur l'énergie consommée dépasse le cadre de ce travail et peut être trouvée dans [32].

L_g : Si L_g augmente, le phénomène de *charge sharing* est moins important, la capacité de déplétion diminue car la charge de déplétion contrôlée par la grille augmente, ce qui provoque une diminution de la pente sous-seuil. Si S diminue, le rapport I_{ON}/I_{OFF} augmente (cf. Fig. 2.21). L'augmentation de L_g a donc un effet **positif** sur S. Si L_g augmente, la capacité C_L augmente. L'augmentation de L_g a donc un effet **négatif** sur C_L. Si L_g augmente, le DIBL diminue car le pourcentage de la zone de déplétion contrôlée par la tension V_{DS} diminue. L'augmentation de L_g a donc un effet **positif** sur le $DIBL$.

V_T : La variation de V_T se fait au travers d'une variation du dopage du canal N_A. Si V_T diminue, N_A diminue, donc la profondeur de la zone de déplétion x_d augmente. La capacité de déplétion diminue car $C_D = \epsilon_{SI}/x_d$, ce qui provoque une baisse de S (cf. Equation 2.9). La diminution de V_T a donc un effet **positif** sur S. Et comme C_L ne varie pas avec V_T, la vitesse ne varie pas. La hausse de x_d causée par la baisse du dopage augmente la proportion de charge de déplétions contrôlées par la tension V_{DS}, ce qui augmente de DIBL. La baisse de V_T a donc un effet **négatif** sur le $DIBL$.

t_{OX} : La diminution de t_{OX} provoque une hausse de la capacité de grille $C_{OX} = \epsilon_{OX}/t_{OX}$, ce qui provoque une amélioration de la pente sous-seuil. La diminution de t_{OX} a donc un effet **positif** sur S. En contrepartie, la hausse de C_{OX} entraine une hausse de C_L et donc une pénalité en vitesse. La diminution de t_{OX} entraine un meilleur contrôle de la grille sur la zone de déplétion, ce qui limite le contrôle de V_{DS} sur la charge de déplétion. L'augmentation de t_{OX} a donc un effet **positif** sur le $DIBL$. Il faut également remarquer que la baisse de t_{OX} entraine une baisse de V_T si le dopage du canal est maintenu constant. Les effets décrits pour une variation de V_T interviennent donc également même si c'est la variation de t_{OX} qui domine.

La technologie $65nm$ de STMicroelectronics utilisée dans ce travail propose trois catégories de V_T (*Low, Standard, High*) et deux catégories de t_{OX} (LP pour un oxyde épais et GP pour un oxyde fin).

Un transistor LP possède un V_T plus élevée qu'un GP. Le délai d'un circuit LP sera donc supérieur à celui d'un circuit GP pour un même V_{DD}. La référence [32] illustre cette différence (cf. Fig. 2.27) sur un multiplieur. On constate que la séparation entre la zone limitée par la contrainte de robustesse (R_3, similaire pour GP et LP) et la zone limitée par la contrainte de vitesse (R_2) se déplace. L'impact des V_T est également illustré. Les points d'énergie minimum pour les deux configurations ne sont donc pas situés aux mêmes fréquences. La technologie LP semble ainsi destinée aux applications à faible débit pour lesquelles elle permet d'atteindre des performances énergétiques largement supérieures par rapport à la situation GP.

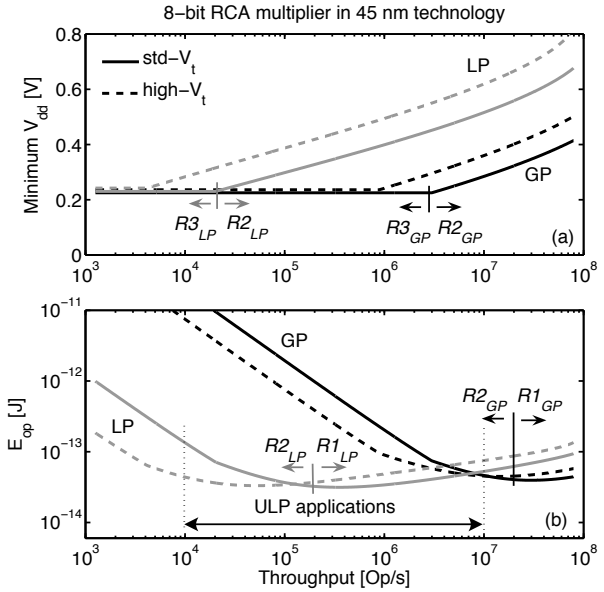


FIGURE 2.27: En haut : comparaison de l'évolution du V_{DD} minimum en fonction de f_{clk} en LP et GP. En bas : comparaison de l'évolution de l'énergie par opération en fonction de f_{clk} en LP et GP [32].

2.3 FPGA à faible consommation

La réduction de la puissance consommée par un FPGA demande une attention particulière lors de chaque étape de la conception et de l'utilisation. Différentes recherches ont étudié ces différentes étapes. Citons :

- La référence [20] étudie comment choisir les paramètres haut-niveau de l'architecture du FPGA pour limiter la puissance dynamique et la puissance statique (cf. Section 3.2.2.3).
- Les références [40] et [39] proposent de répartir deux V_{DD} différents sur la structure. Chaque tuile peut alors être alimentée avec un V_{DD} élevé, bas ou peut être éteinte suivant la configuration. Cette technique est associée à une modification des algorithmes de placement et routage pour répartir de manière optimale le nombre de blocs à haut V_{DD} et donc à haute vitesse.
- La référence [39] proposent une architecture comprenant deux V_{DD} et deux V_T. Les parties du FPGA qui ne doivent pas effectuer de transitions rapides (blocs de configurations) sont implémentées avec un V_T élevé pour réduire les courants de fuite tandis que les parties contraintes en vitesse sont implémentées avec un V_T faible. La répartition des blocs de V_{DD} élevé et faible est fixée à travers la structure et c'est à l'algorithme de placement à répartir les portions de circuit demandant de hautes performances sur les blocs rapides. Cette technique impose une pression supplémentaire sur l'algorithme de placement et de routage.

Pour bénéficier de la réduction de consommation atteinte par les circuits ULV, la référence [41] explore la possibilité de diminuer la tension d'alimentation d'un FPGA COTS tout en contrôlant la chute de ses performances. Les auteurs déterminent ainsi que la réduction de la tension d'alimentation

de $1.2V$ à $\cong 500mV$ permet une réduction d'un facteur 10 de la puissance consommée (cf. Fig. 2.28). Les auteurs signalent également que les tensions, auxquelles ils ont soumis le FPGA, sont limitées par les mécanismes internes au FPGA utilisé qui efface la configuration en dessous d'une tension minimale en effectuant un *reset* de la structure. Ceci laisse à penser que la tension d'alimentation d'une structure *custom* pourrait être encore abaissée.

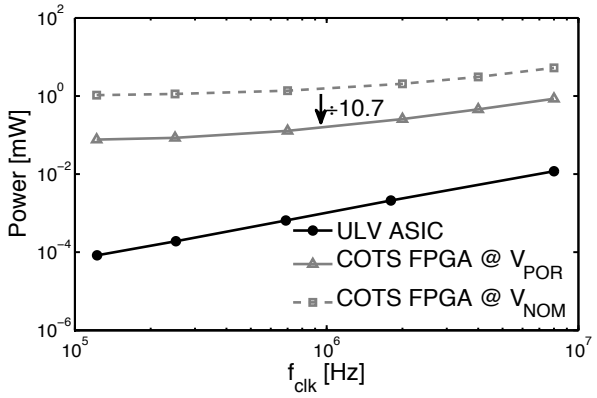


FIGURE 2.28: Comparaison de la puissance consommée par un coprocesseur AES sur 8 bits implémentés sur un Spartan3 de Xilinx opéré à sa tension nominale V_{NOM} ou à sa tension de *Power-On-Reset* V_{POR} ainsi qu'un ASIC ULV [41].

La réduction de la tension d'alimentation du FPGA exacerbe la sensibilité aux variabilités de *process*. Lorsque des technologies avancées sont utilisées, l'éloignement des *corners* combinés à une utilisation dans le domaine ULV imposera au concepteur du FPGA un *sizing* pessimiste et une tension V_{DD} relevée par rapport aux simulations typiques. A plus haut niveau, comme il n'y a pas de moyen de savoir si le chemin critique d'un circuit synthétisé ne contiendra aucun transistors lents, le concepteur devra accepter une fréquence d'opération supérieure par sécurité. Contrairement aux ASICs, la reconfigurabilité des FPGAs permet de compenser ces effets en évitant l'utilisation de ressources lentes pour l'implémentation des chemins critiques. Un placement & routage attentif au délai des différents éléments permettrait également d'éviter les sections défectueuses du FPGA. La référence [49] indique cependant que 3 questions subsistent :

- Comment extraire les informations de délai pour chaque élément de la structure ?
- Comment utiliser ces informations de délai pour optimiser le routage ?
- Quel est le bénéfice potentiel de ce type de routage ?

Les 2 premières questions restent des questions ouvertes bien que certains résultats partiels existent. Ainsi, il existe des méthodes qui permettent de mesurer le délai combiné de 5 à 7 LUTs chainées en synthétisant des *ring oscillators* sur le FPGA. Il n'existe cependant pas de méthode permettant de mesurer rapidement, précisément et séparément le délai de chaque LUT. La référence [49] répond partiellement à la troisième question en évaluant le gain potentiel d'une connaissance parfaite des délais dans la structure. Les auteurs indiquent qu'un tel routage permet une réduction de $1.5\times$ du délai moyen sur le *benchmark* MCNC.

Dans l'industrie, Actel propose des familles de FPGAs à faible consommation (IGLOO [42] et ProASIC 3L [43]) qui implémentent différents modes de fonctionnement. Ces FPGAs peuvent ainsi être opérés en 4 modes différents : *ON*, *Static*, *Idle* et *Flash*Freeze*.

Ces techniques permettent de réduire avec plus ou moins de succès la consommation des FPGAs. Cependant, la réduction de puissance ne permet pas de ramener la consommation dans le domaine des circuits ULP intégrables dans le budget de puissance d'un EAS. Toutes les structures opérant à des tensions d'alimentation supérieure ou égale à $1V$ présenteront systématiquement une consommation dynamique largement supérieure aux exigences des EAS.

2.4 eFPGA

La conception de SoCs complexes peut généralement bénéficier de l'intégration d'un certain degré de flexibilité. Cette flexibilité peut servir à couvrir plusieurs petites niches industrielles avec un seul *design* amortissant ainsi les coûts de développement. La possibilité de modifier le SoC après fabrication permet également de réduire le temps d'arrivée du système sur le marché (*"time-to-market"*). Elle est généralement apportée sous forme de reconfigurabilité software par l'addition de microprocesseurs. La généralité de ceux-ci implique un handicap considérable en termes de surface, de vitesse et de puissance par rapport à l'ajout d'un coprocesseur ASIC. L'embarquement de FPGAs se situe entre l'ASIC et le CPU offrant un compromis intéressant.

L'architecture des blocs FPGAs à embarquer (eFPGAs) est soumise aux mêmes compromis que celle des FPGAs standards (cf. Section 3).

Les eFPGAs à embarquer dans des SoCs doivent pouvoir être synthétisé par le concepteur du SoC pour qu'il puisse adapter la taille et d'autres paramètres, si possible, à ses besoins. Cette souplesse lors de la création *layout* est un défi technologique.

La référence [44] propose une méthode de conception de eFPGAs basée sur l'utilisation d'un *Flow* ASIC traditionnel. Le but étant de laisser la possibilité au connecteur du SoC de choisir la taille de la structure à embarquer. Il peut alors synthétiser le eFPGA le plus adapté aux besoins du projet sans avoir à insérer une IP de taille fixe qui serait sur- ou sous-dimensionnée. Cette flexibilité handicape cependant la surface car la structure est moins dense qu'un eFPGA conçu sur un *Flow custom*. Les autres facteurs de mérite (vitesse et consommation) suivent eux aussi cette tendance agrandissant ainsi l'écart entre l'implémentation d'un coprocesseur sur l'eFPGA et l'implémentation ASIC. Pour éviter une grande pénalité lors du passage au *Flow* ASIC traditionnel, les auteurs implémentent séparément des *tactical cells* en *full custom*. Il s'agit de blocs dont dont le délai représente une grande portion du délai sur le chemin critique moyen tels que les LUT, les Flip-Flop de configurations ou certains multiplexeurs de routage. C'est donc sur ces blocs que le gain d'une implémentation *full custom* sera le plus grand.

La référence [45] propose également un *Flow* pour automatiser le *layout* de ces structures sans utiliser un *Flow ASIC semi-custom*. Ils utilisent des macro-blocs paramétrisables auxquels le concepteur fournit les paramètres des CLBs et des SB (I, N, K, topologie du SB, W).

Au niveau de l'industrie, Menta propose deux types d'IP intégrable dans un SoC. L'eFPGA Core-S est synthétisable par leur environnement de conception eFPGA Creator [46]. Le concepteur a la possibilité de paramétriser ses tuiles et d'ajouter des *macrocells* tels que des multiplieurs ou de la mémoire selon les perspectives d'utilisation du SoC. L'eFPGA Core-H est une *macrocell* paramétrisable par les services de conception de Menta. Il s'agit de la seule entreprise proposant des outils de création et de conception de logique programmable à ses clients. Pour atteindre une plus grande part du marché de la conception de SoC, la seconde version des eFPGA Core ne se base plus sur un *design flow custom* mais sur une librairie de cellules standards et sur un *Flow* automatisé sacrifiant ainsi une partie des performances atteintes.

Microchip a ajouté dans certains microcontroleurs des *Configurable Logic Cell* (CLC, [47]). Il s'agit de petites cellules de 16 entrées qui contiennent 8 blocs configurables de 4 entrées. Ces blocs peuvent être programmés pour effectuer un nombre restreint de fonctions combinatoires (AND, NAND, AND-OR, AND-OR-INVERT, OR-XOR et OR-XNOR) ainsi que 4 sortes de *latches* (S-R, D-clocked avec Set et Reset, D-transparent avec Set et Reset et J-K-clocked avec reset). Ces blocs ont donc une capacité de synthèse limitée. Le nombre de modules CLC dépend du dispositif. On peut, par exemple, en trouver 4 pour un petit microcontrolleur de 8 bits.

Chapitre 3

Optimisation des paramètres haut-niveau et choix de l'architecture

Ce chapitre décrit l'analyse de l'impact des différents paramètres haut-niveau de l'architecture ainsi que de la topologie du FPGA sur la puissance consommée et sur la vitesse. Dans un premier temps, chaque étape du CAD Flow utilisé tout au long de l'optimisation est décrite et analysée. Le problème d'optimisation est ensuite énoncé et l'impact de chaque paramètre du CLB est modélisé. Une démarche semblable est ensuite réalisée pour la topologie de la structure de routage et ses paramètres.

3.1 *CAD Flow* : analyse

La structure du *CAD Flow* utilisé dans ce travail lors de l'optimisation de l'architecture est identique à ce qui a été présenté dans la Section 2.1.3. Cette section comprend une description des différents *software* utilisés.

3.1.1 Synthèse et optimisation logique

La première étape en partant d'un fichier verilog est de le convertir en une *netlist* composée d'un réseau (séquentiel ou combinatoire) de fonctions logiques. Le format couramment utilisé pour l'importation/exportation de *netlist* est le *BLIF* (*Berkley Logic Interchange Format* [55]). Les fonctions logiques sont décrites par des LUTs de taille variable et par des registres.

Les *netlists* passent par une étape d'optimisation logique. Différents *softwares* permettent de réaliser cette optimisation (*SIS* et *ABC* développés à l'université de Berkley par exemple [52] [54]). Ces algorithmes transcrivent la *netlist* en graphe orienté acyclique (DAG). Les noeuds de ce réseau booléen représentent les signaux intermédiaires du circuit ainsi que la fonction booléenne qui les caractérisent. Les arêtes représentent les entrées de la fonction booléenne vers laquelle elles pointent. La Figure 3.1 (a) présente un exemple simple de traduction de *netlist*.

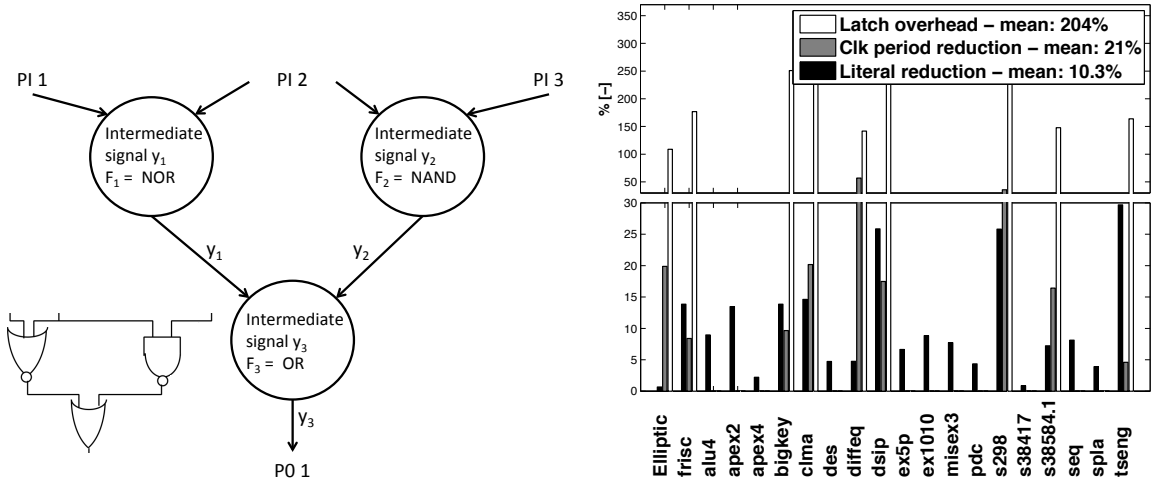
Une série d'opérations permett alors d'optimiser le circuit. Citons :

- Simplification du nombre de noeuds : le but est d'obtenir une représentation logique comprenant le minimum d'arêtes (les arêtes sont appelées *literals*) et de noeuds intermédiaires tout en préservant la fonctionnalité.
- *Technology mapping :* le but est d'utiliser les portes logiques définies dans une librairie pour représenter les fonctions de tailles arbitraires issues du graphe booléen. Différentes optimisations permettent de limiter la surface et le délai par exemple, en évaluant le *fanout* sur les différents chemins.
- Restructuration : le but est de réduire le délai du circuit une fois que le réseau est décomposé en portes simples (NAND à 2 entrées et inverseurs) en réduisant la profondeur logique du chemin critique. Pour réaliser cela, l'algorithme groupe certaines sections du chemin critique et tente de les resynthétiser en utilisant moins de portes de la librairie. Cette étape peut également être réalisée indépendamment des librairies.
- *Retiming :* Le but est de déplacer les registres à travers le réseau. Il est possible de minimiser le nombre de registres mais généralement, cette étape sert à réduire le délai sur le chemin critique. Cette étape est souvent couplée à une étape de restructuration. De cette facon, combiner le *retiming* qui répartit la logique combinatoire entre les registres et la restructuration qui optimise chaque sous-réseau combinatoire permet d'exploiter les interactions entre les réseaux combinatoires séparés par les registres.

La description détaillée de ces algorithmes dépasse le cadre de ce travail. Cependant, les performances de l'étape d'optimisation logique ont été évaluées pour justifier la pertinence de l'intégration dans le *CAD Flow*.

Pour se faire, les 20 plus grands circuits du *benchmark* MCNC ont été traités par SIS (restructuration et *retiming*) [52]. La Figure 3.1 (b) montre le pourcentage de réduction de la période d'horloge ainsi que le pourcentage de réduction de *literals*. La réduction du nombre de *literals* est cruciale. En effet, ils sont définis comme les variables qui apparaissent dans les fonctions logiques à chaque noeud du réseau. Leur nombre est donc fortement corrélé au nombre de paires de transistors qui apparaitront dans les portes logiques implémentées en CMOS. Dans un FPGA, l'interprétation n'est pas aussi directe mais il est évident que la réduction du nombre de *literals* implique une réduction du nombre de BLEs nécessaires.

La Figure 3.1 (b) montre que l'étape d'optimisation permet de réduire en moyenne de 10% le nombre de *literals* du réseau mais les disparités entre les circuits sont grandes. La réduction de la période d'horloge moyenne atteint 21% au prix d'une augmentation conséquente du nombre de registres.



(a) Exemple de traduction d'une *netlist* en réseau booléen. Les PI et PO indiquent les entrées et sorties du circuit.

(b) Résultats d'optimisation d'une série de circuits de MCNC.

FIGURE 3.1: SIS : principe de fonctionnement et résultats.

3.1.2 *Mapping* technologique

L'étape suivante du *CAD Flow* consiste, à traduire la *netlist* comprenant des fonctions booléennes de taille variable en une *netlist* composée de LUTs de taille K ou inférieure. Il existe différentes manières de réaliser cette répartition suivant l'objectif. Il est possible de cibler la minimisation du nombre de LUTs nécéssaires, la minimisation du délai sur le chemin critique ou encore la facilité qu'aura la suite du *CAD* à router la *netlist* [50].

Différents *softwares* permettent de réaliser ce *mapping*. Citons ABC [54], FlowMap [50], Quartus [51]. Ils opèrent, comme les algorithmes d'optimisation, en transformant la *netlist* en DAG. Ils définissent alors un cône de taille K sur un noeud x (*K-feasible cone*) comme un sous-graphe constitué du noeud x et de tous les noeuds qui sont reliés par un chemin direct aboutissant à x tels que :

- Le nombre d'entrées du sous-graphe est inférieur ou égal à K.
- Tous les chemins reliant un noeud du sous-graphe à x sont entièrement compris dans le sous-graphe.

Il est évident qu'un tel sous-graphe peut être implémenté dans une LUT de taille K par définition de celle-ci. Le *mapping* consiste donc à réduire le réseau booléen en un réseau de cônes de taille K.

La Figure 3.2 illustre le résultat du *mapping* technologique sur de petites portions de code *.blif*. On y constate la séparation des portes complexes en LUT de 4 entrées.

3.1.3 Groupement des LUTs et des DFF en CLB

Cette étape convertit la *netlist* de LUTs de taille K et de registres en une *netlist* de blocs logiques. Pour cela, une série d'informations concernant l'architecture des CLBs doit être fournie :

- Taille des LUTs : K
- Nombre de BLEs par grappe : N
- Nombre d'entrées par grappe : I
- Nombres de *clocks* différentes par grappe
- Possibilité de contrôler chaque pin de sortie par n'importe quelle sortie de BLE (multiplexage des sorties)

Les trois premières informations sont indispensables.

Before Mapping	After Mapping
.latch m D0 re clk 0 .latch o4 E0 re clk 0	.latch n53 D0 0 .latch n57 E0 0
.subckt mult_M1 a0=a a1=b a2=c a3=d b0=e b1=f b2=g b3=h c0=m0 c1=m1 c2=m2 c3=m3 c4=m4 c5=m5 c6=m6 c7=m7	.subckt mult_M1 a0=a a1=b a2=c a3=d b0=e b1=f b2=g b3=h c0=m0 c1=m1 c2=m2 c3=m3 c4=m4 c5=m5 c6=m6 c7=m7
.names o3 o4 m22 m23 m24 m25 m26 m27 B0 11010101 1 01010001 1 11010001 1 11010100 1 11000101 1 01000101 1 1---0101 1	.names m22 n59 n57 n68 n58_1 001- 0 ---1 0 .names m23 m25 m27 n60 n59 0111 0 101- 0 1100 0 .names n61 n62 n63 n64 n60 110- 0 -0-1 0

FIGURE 3.2: Exemple de *mapping* technologique. L'algorithme ne modifie ni les registres ni les *blackboxes* (ici, un multiplieur) et sépare les fonctions booléennes en LUT de taille 4.

Le logiciel T-VPACK, intégré dans le logiciel de placement et routage VPR [17], qui réalise cette étape permet également de grouper les LUTs en optimisant le délai dans le circuit. Pour cela, l'utilisateur doit fournir une estimation du rapport des délais inter-CLB sur intra-CLB. Il est également possible d'optimiser le groupement par rapport au nombre de connexions inter-CLB au lieu du délai. Le logiciel maximise alors le partage de pins d'entrée des grappes et tente de grouper les BLEs chainés dans une même grappe.

3.1.4 Placement et routage

Le placement et routage d'une *netlist* groupée sur une architecture de FPGA est l'étape clef du *CAD Flow*. Si l'architecture est non-standard comme l'architecture développée dans les Sections 3.2 et 2.1.2, le logiciel libre VPR [17] est le logiciel de placement et routage le plus répandu.

L'architecture est convertie en un graphe regroupant les ressources de routage qu'elle offre. Les différentes pins des CLBs et les segments de routage inter-CLBs forment les noeuds du graphe et les arêtes représentent une connexion possible entre ces éléments.

La Figure 3.3 illustre ce graphe de ressources de routage pour une tuile de FPGA simplifiée.

Le placement est réalisé par un algorithme de recuit simulé dont la fonction de coût est définie par [56] :

$$Cost = \sum_{n=1}^{N_{nets}} q(n) \left\{ \frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)} \right\} \tag{3.1}$$

avec :

- q(n), un facteur de compensation pour tenir empiriquement en compte le fait que les segments transportant un même signal peuvent être groupés, partageant ainsi le coût dû à l'éloignement.
- $bb_x(n)$ et $bb_y(n)$, la distance entre les deux extrémités de la connexion n.
- $C_{av,x}(n)$ et $C_{av,y}(n)$, la capacité moyenne des canaux de routage (en terme de segment).

VPR offre la possibilité de cibler le placement qui minimise la longueur totale de segment utilisée ou bien le placement qui minimise conjointement la longueur de segment et une évaluation grossière du délai sur le chemin critique (avant routage).

La Figure 3.4 illustre le résultat de cette étape de placement. Il s'agit d'illustrations réalisées par **l'interface graphique du complément au logiciel VPR (VPR2ELDO) développé dans la Section 5.2** qui permet d'interpréter les formats de sortie de VPR.

Après cette étape de placement, VPR doit implémenter chaque connexion de la *netlist* sur le graphe des ressources de routage. Pour cela, le logiciel implémente un algorithme de recherche de chemin basé

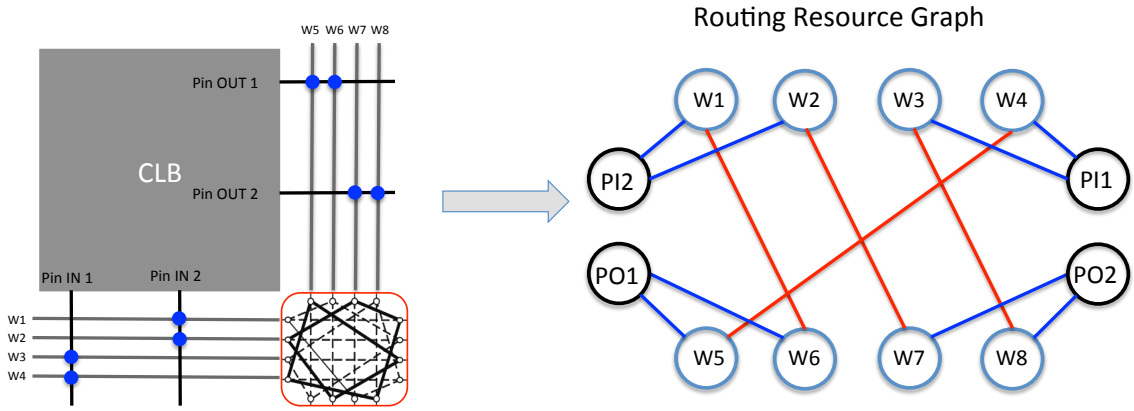
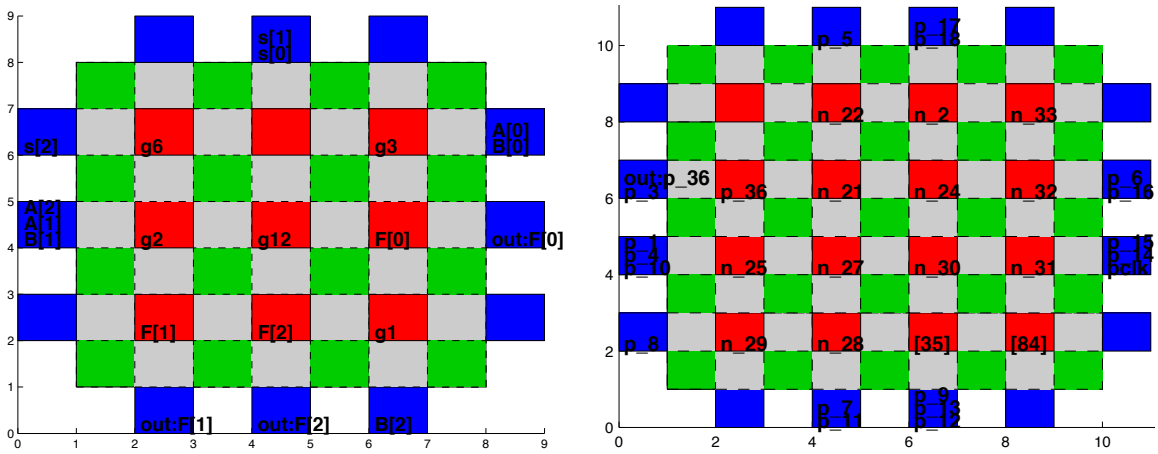


FIGURE 3.3: Exemple de conversion de l'architecture du FPGA en graphe de ressource de routage. Les noeuds notés P correspondent aux pins du CLB et les noeuds W correspondent aux segments de routage. Les arêtes reliant les noeuds W correspondent aux connections potentielles entre segments dans le *Switch Block* alors que les arêtes reliant les noeuds P aux noeuds W correspondent aux connections potentielles entre segments et pins dans les *Connection Blocks*.



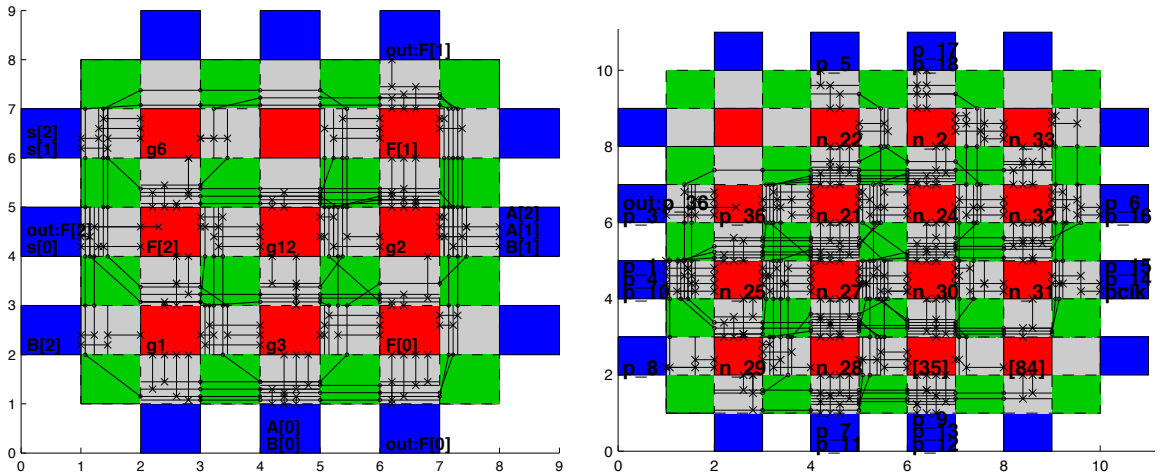
(a) Placement d'une ALU sur 3 bits.

(b) Placement d'un multiplieur sur 16 bits.

FIGURE 3.4: Exemple de placement des CLBs sur une structure. En bleu, les *pads* d'entrée/sortie, en vert, les SBs, en gris les CBs et en rouge les CLBs.

sur un algorithme de Dijkstra. Le routeur commence par router toutes les connexions sur le graphe sans se préoccuper du nombre fini de segments par canal de routage. Il suffit pour cela d'augmenter le nombre de noeuds W de la Figure 3.3. Chaque itération consiste ensuite à retirer les connexions qui passent par les canaux surchargés pour les re-router en minimisant une fonction de coût. Le coût de routage d'une connexion correspond à la somme des coûts d'utilisation de chaque ressource de routage. Le coût d'utilisation d'une ressource de routage dépend de sa sur-utilisation à l'itération en question et aux itérations précédentes. En augmentant à chaque itération le coût lié à la sur-utilisation des canaux, on peut forcer les connexions qui possèdent des routages alternatifs à être re-router. Les connexions qui restent sont donc les connexions critiques qui n'ont pas de routage alternatif (ou qui ont un routage alternatif très long donc très peu performant) [56].

Pour réaliser un placement et routage d'une *netlist*, il faut décrire avec précision l'architecture de FPGA utilisée et ses performances en densité et en vitesse. VPR utilise un format spécifique décrivant cette architecture. Il faut ainsi fournir tous les éléments décrits dans la Table 3.1 [17].



(a) Routage d'une ALU sur 3 bits.

(b) Routage d'un multiplieur sur 16 bits.

FIGURE 3.5: Exemple de routage des connexions sur une structure.

TABLE 3.1: Liste et description des éléments caractérisant une architecture.

Surface	Architecture	délais
surface approximative d'un CLB	distribution de longueur des segments	capacité d'entrée d'un CLB vue d'un segment de routage
surface des MOS constituant les MUX	$F_{c,in}$	délai d'un segment vers une pin d'entrée d'un CLB
surface d'un *buffer*/multiplexeur unité : surface de MOS de taille minimale	$F_{c,out}$	délai de sortie BLE vers sortie CLB
	agencement des pins du CLB sur son pourtour	délai d'entrée CLB vers entrée BLE
	description des segments (longueur, résistance, capacité, directionnalité)	délai de sortie BLE vers entrée BLE dans le même CLB
	répartition des SBs et CBs	délai combinatoire dans un BLE
	type de *switch* de routage (MUX ou *buffer*)	délai de l'entrée d'un BLE au DFF + setup
		délai *Clok-to-Q* du DFF vers sortie BLE
		délai des *pads* I/O
		délai dans les *switches* de routage
		C_{in}, C_{out} des *switches*

La Figure 3.5 illustre graphiquement le résultat de routage par VPR de deux circuits. Ces illustrations ont été réalisées par VPR2ELDO développé dans le cadre de ce mémoire.

3.2 Choix des paramètres : K, N, I

Les performances d'un FPGA sont déterminées par un grand nombre de paramètres tant à haut niveau qu'à bas niveau. Les interactions entre tous les paramètres de la structure rendent le problème d'optimisation très complexe. Cette section présente une étude de ce problème ainsi que les solutions proposées et mises en oeuvre dans ce travail.

3.2.1 Enoncé du problème d'optimisation

Objectifs : Les facteurs de mérite considérés pour ce projet sont la puissance consommée et la vitesse. La minimisation de la surface ne fait pas partie des objectifs principaux pour laisser un degré de liberté non contraint lors de la conception. De plus, l'objectif visé d'embarquer une structure reconfigurable

compétitive avec les performances des ASICs dans un EAS impose des contraintes plus strictes sur la vitesse et la puissance.

Degrés de liberté : La structure reconfigurable peut être divisée en deux secteurs qui interagissent : la logique programmable (CLB, BLE) et le routage programmable (SB, CB, segments). A haut niveau, les blocs de logique sont caractérisés par les paramètres N et K qui ont été introduits dans le Chapitre 2. Le routage est caractérisé par $F_{c,in}$, $F_{c,out}$, F_s ainsi que par la topologie des SBs et CBs. Le nombre d'entrées des CLBs, I, est un paramètre influençant à la fois les performances du routage et des blocs logiques.

Contraintes : Le nombre d'entrées des CLBs est contraint pour maintenir une bonne accessibilité des LUTs. Les paramètres du routage et la topologie des SBs/CBs sont également contraints pour laisser une bonne flexibilité et permettre un routage raisonnablement facile aux algorithmes décrits dans la Section 3.1.

Problème : Le but de l'optimisation est de trouver le ou les couples (K,N) qui minimis(ent) une fonction objectif $f(\cdot)$. Cette minimisation passe par l'étude de l'évolution du délai sur le chemin critique $D(K,N)$ et de la puissance statique et dynamique $P_{tot}(K,N)$ avec K et N. La fonction objectif est fixée par les exigences de l'application visée en termes de puissance et de vitesse qui indiquent l'importance relative d'un facteur de mérite par rapport à l'autre : $f(D,P_{tot})$.

Hypothèses simplificatrices : Pour simplifier l'étude, les modèles $D(K,N)$ et $P_{tot}(K,N)$ seront construits indépendamment des performances de l'implémentation bas-niveau. Ceci suppose que le délai et la puissance évoluent de façon similaire avec K et N pour toutes les implémentations.

3.2.2 Solution proposée

Pour réduire le nombre de degrés de liberté du problème, les paramètres haut-niveau du routage ($F_{c,in}$, $F_{c,out}$, F_s et topologie) ont été fixés à des valeurs raisonnables qui, selon la littérature (Section 2.1.2), permettent d'atteindre une bonne flexibilité. F_s est fixé à 3, $F_{c,in}$ et $F_{c,out}$ sont fixés à 1 et les SBs sont en topologie disjointe. Ces paramètres garantissent une excellente flexibilité comparé aux FPGAs commerciaux.

Etant donné que I est contraint par l'accessibilité des CLBs, la première étape est d'examiner l'évolution de l'accessibilité avec K,N et I. Il faut ensuite établir un modèle de l'évolution du délai et de la puissance consommée avec K et N.

3.2.2.1 Impact de K, N, I sur l'accessibilité

Comme la plupart des paramètres architecturaux des FPGAs, I est soumis à un compromis : si I est trop petit, le routage local est simplifié mais certaines LUTs ne pourront pas être entièrement utilisées. Si I est trop grand, le routage local augmente en complexité et l'accessibilité augmente puis sature. On impose un taux d'occupation :

$$I \;\; t.q. \;\; \#LUT \; par \; CLB \geq 0.98 \times N \tag{3.2}$$

En fonction de la connectivité du routage local, le nombre d'entrées nécessaires pour atteindre 98% d'occupation est inférieur à $K \times N$ ([13]). En effet, 3 phénomènes réduisent le nombre d'entrées nécessaires :

1. Si l'algorithme de placement dispose les LUTs en série dans un seul CLB, la sortie d'une LUT peut être bouclée sur l'entrée de la suivante par le routage local si celui-ci est suffisamment dense (cf. Fig. 2.1 (c)).
2. Si deux LUTs partageant un ou plusieurs signaux d'entrée sont placées dans un CLB, elles peuvent partager une même entrée.
3. Si une LUT n'est utilisée que partiellement, la demande en signaux d'entrée est réduite.

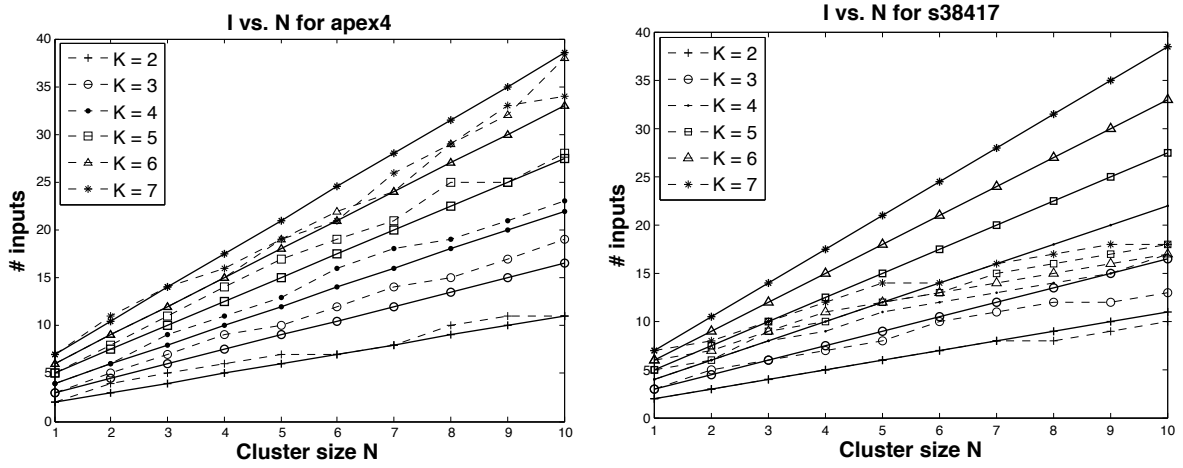
Les recherches de [14] montrent que si la taille des LUTs vaut 4, un nombre d'entrées valant $2N + 2$ permet d'atteindre en moyenne 98% d'occupation des grappes. La référence [13] étend ce résultat en montrant que la relation :

$$I = \frac{K}{2} \times (N + 1) \tag{3.3}$$

permet d'estimer le besoin en entrées selon K et N.

Dans le cadre de ce mémoire, ces résultats ont été reproduits et analysés. Deux circuits appartenant au *benchmark* utilisé par [13] et [15] (MCNC [18]) ont été synthétisés sur une architecture standard. Celle-ci était composée de CLBs de taille 4 et de LUTs de taille 4 entourés par un routage unidirectionnel dont les segments s'étendaient sur 1 CLB. Le nombre d'entrées est ensuite augmenté jusqu'à atteindre un taux d'occupation des CLBs supérieurs à 98%. Les synthèses sont menées à l'aide du *CAD Flow* détaillé dans la Section 2.1.3. Dans un premier temps, un circuit combinatoire a été synthétisé (apex4 comprenant 1262 BLEs pour K=4). Le second circuit synthétisé est un circuit séquentiel composé d'un grand nombre de registres (s38417 comprenant 6406 BLEs pour K=4 et 1636 registres répartis dans des BLEs dont la logique n'est pas toujours utilisée). Comme représenté à la Figure 3.6, le circuit combinatoire est plus exigeant en terme de nombre d'entrées pour les CLBs car le circuit séquentiel comprend un certain nombre de BLEs qui ne sont utilisés que pour leur registre et donc qui n'ont besoin que d'une seule entrée. En effet, suivant l'architecture des grappes utilisées, le registre n'est accessible qu'à travers la LUT qui ne sera que partiellement utilisée. Certains FPGAs commerciaux contournent ce problème en plaçant un multiplexeur de contournement de la LUT qui permet à un certain nombre d'entrées du bloc logique d'atteindre directement le registre [27].

La Figure 3.6 montre qu'il existe une bonne corrélation entre la loi empirique dressée dans [13] et les résultats obtenus sur le circuit combinatoire du *benchmark*. La demande du circuit séquentiel en entrées est, en revanche, surestimée par l'Equation 3.3 à cause du nombre de BLEs qui ne sont utilisés que pour leur registre. L'utilisation de l'Equation 3.3 comme borne supérieure de la demande en entrées pour garantir un taux de remplissage raisonnable semble donc être correcte et permet d'éliminer un des trois degrés de liberté du problème d'optimisation.



(a) Evolution du nombre d'entrées du CLB nécéssaire pour maintenir le taux d'accessibilité cible de 98% pour un circuit combinatoire (APEX4).

(b) Evolution du nombre d'entrées du CLB nécéssaire pour maintenir le taux d'accessibilité cible de 98% pour un circuit séquentiel (S38417)

FIGURE 3.6: Etude de la relation entre le nombre d'entrées d'un CLB et la taille des LUTs. Lignes continues : modèle d'accessibilité de l'Equation 3.3. Lignes pointillées : mesure de l'accessibilité après groupement en CLB.

3.2.2.2 Impact de K, N sur la vitesse

Pour établir l'évolution des performances en vitesse avec K et N, on étudie l'évolution du délai moyen sur les chemins critiques d'une série de circuits de *benchmarks*. Il est très difficile de définir un modèle de l'évolution de ce délai avec K et N car il est composé de deux types de contributions : le délai intra-CLB et le délai inter-CLB.

Le délai intra-CLB est influencé par $F_{c,in} \times W$, K, N et I comme l'indique la Figure 3.7. Chacun de ces paramètres influence la taille des multiplexeurs et donc leurs délais. Ces délais dépendent également des performances de l'implémentation qui sont inconnues au moment de la conception de l'architecture.

Le délai inter-CLB dépend de $F_{c,in}$, $F_{c,out}$, F_s, des topologies et des performances de l'implémentation des *switches*. Cette composante du délai dépend également de W, la largeur des canaux de communication qui doit être adaptée à la taille du FPGA. W est inconnu lors de la conception d'un eFPGA dont la taille doit être modulable selon la quantité de logique à intégrer au SoC ainsi que selon le budget de puissance disponible.

Au delà de ces dépendances, la répartition du délai total en délai intra- et inter-CLB sur le chemin critique moyen est également influencée par ces paramètres.

Pour prendre une décision sur les degrés de liberté restants, il faut prendre des hypothèses simplificatrices supplémentaires. On suppose que l'évolution des délais intra- et inter-CLB avec les paramètres est faible par rapport à l'évolution de la répartition des types de délais sur le chemin critique. Cette hypothèse est validée par la domination du délai de routage inter-CLB sur le délai intra-CLB que l'on observe dans la littérature (cf. Section 2.1.4.2) ainsi que dans les résultats de cette étude (cf. Section 4.5). Cet écart implique que la variation de la répartition des types de délais avec les paramètres est plus importante que la variation des délais en eux-mêmes. On va donc fixer les délais intra- et inter-CLB à des valeurs raisonnables et on va mesurer l'évolution du délai moyen total sur le chemin critique avec K et N.

La seule manière de lever l'inconnue liée à l'implémentation est de simuler l'évolution des délais pour différentes valeurs des paramètres.

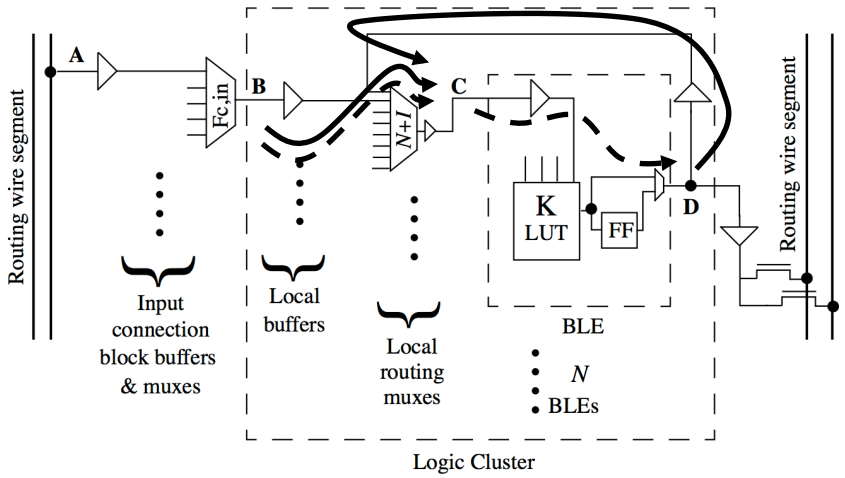


FIGURE 3.7: Mise en évidence des dépendances des délais intra-CLB envers K et N. En continu : chemins dont le délai dépend de N, en pointillé : chemin dont le délai dépend de K.

Six circuits du *benchmark* MCNC (3 circuits séquentiels et 3 circuits combinatoires) ont été synthétisés sur une architecture possédant un routage unidirectionnel et des SBs de topologie Wilton. K a été varié de 2 à 7 et N de 2 à 10. La moyenne de leurs délais totaux intra- et inter-CLB est reprise sur la Figure 3.8 (a).

On remarque que les délais totaux intra- et inter-CLB diminuent avec K, résultat de l'augmentation de la capacité logique des LUTs qui provoque une diminution du nombre de LUTs sur le chemin critique

et une réduction globale de la demande en ressource de routage. Le délai intra-CLB augmente si N augmente car une partie des connexions auparavant réalisées sur le routage externe est alors intégrée dans le routage interne provoquant une augmentation du délai total intra-CLB. Pour la même raison, le délai inter-CLB diminue avec N.

La Figure 3.8 (b) présente l'évolution de la proportion de délai intra-CLB total dans le délai total moyen. On constate que cette proportion n'évolue pas avec K. Ceci indique que l'augmentation de la capacité logique des LUTs n'influence pas significativement la proportion de routage local par rapport au routage global. Ceci est dû à la dominance du routage global sur le délai. Cette proportion augmente fortement avec N illustrant le transfert du délai du routage global vers le routage local.

Remarque : L'augmentation de K et N provoque aussi une augmentation du délai dans les LUTs et le routage local qui n'a pas été prise en compte dans ces mesures. La diminution du délai intra-CLB est donc surestimée. Mais, étant donné que le délai des LUTs est linéairement proportionnel à K, les tendances observées restent pertinentes. Le délai du routage local varie avec le nombre de multiplexeurs sur un chemin le traversant qui évolue en $\lceil\sqrt{N+K/2\times(N+1)}\rceil$. L'évolution de la contribution du routage local au délai intra-CLB est, elle aussi, monotone, ce qui valide les tendances.

En conclusion, au vu des simulations réalisées et en gardant à l'esprit les simplifications effectuées, l'augmentation de K et N permet d'augmenter les performances en vitesse de la structure. Le modèle empirique du délai moyen sur le chemin critique ne présente pas d'optimum selon K ou N.

3.2.2.3 Impact de K, N sur la puissance consommée

A. Modèle de consommation statique :

La puissance statique a été évaluée en mesurant le nombre de multiplexeurs 2 :1 nécessaire à l'implémentation d'une tuile. L'évolution de ce nombre avec K et N permet d'obtenir une mesure de la puissance statique en terme de puissance statique équivalente de multiplexeurs 2 :1.

Dans le CLB, on distingue trois contributions :

1. Routage local d'entrée : $K\times N\times(I+N-1)$ car il est composé de $K\times N$ multiplexeurs de $I+N$ entrées.
2. Routage local de sortie : $N\times(N-1)$ car il est composé de N multiplexeurs de N entrées.
3. BLE : $N\times\left\{(K^2-1)+\frac{P_{STAT,REG}}{P_{STAT,MUX}}\right\}$ car il est composé de N multiplexeurs de K^2 entrées (LUTs) ainsi que de N registres.

Au niveau du routage global, on distingue deux contributions :

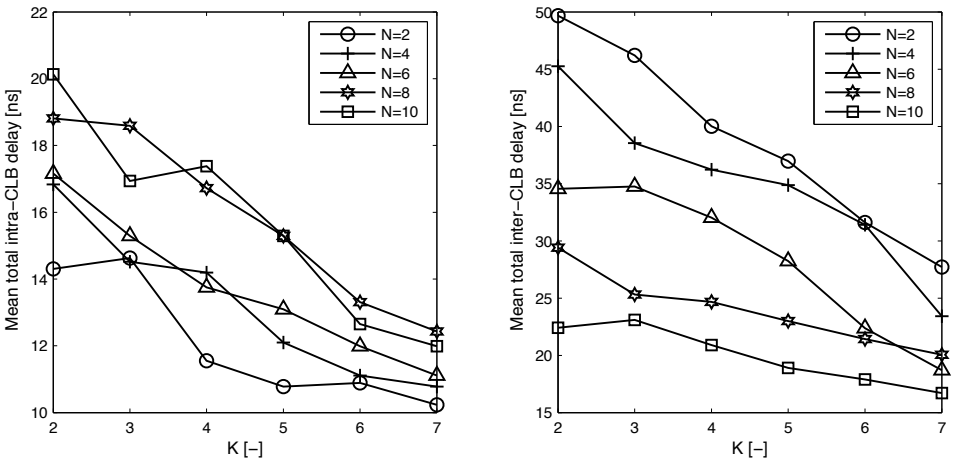
1. CB : $\frac{I}{2}\times(F_{c,in}W-1)$ car il est composé de $I/2$ multiplexeurs de $F_{c,in}W$ entrées.
2. SB : $\frac{W}{2}\times(F_s+\frac{N}{2}-1)$ car il est composé de $W/2$ multiplexeurs de $F_s+N/2$ entrées.

La puissance statique totale du FPGA selon ce modèle s'écrit donc :

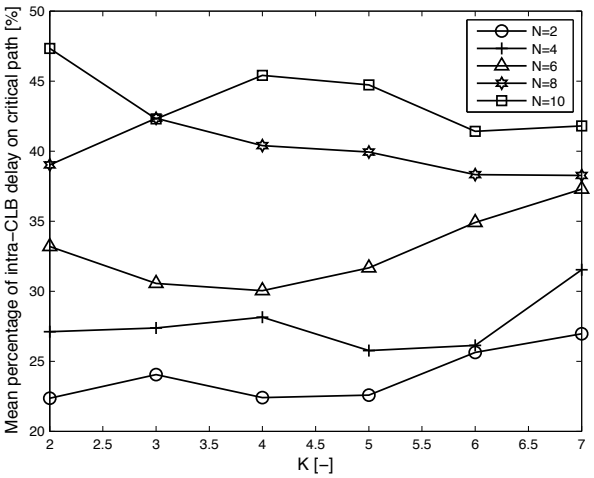
$$\begin{aligned}P_{stat,FPGA} &= D_X D_Y\times[KN(I+N-1)+N(N-1)+N\left\{(K^2-1)+\frac{P_{stat,reg}}{P_{stat,mux}}\right\}\\ &\quad+\frac{I}{2}(F_{c,in}W-1)+(F_s+\frac{N}{2}-1)\frac{W}{2}]\end{aligned}\tag{3.4}$$

avec, D_X et D_Y, les dimensions de la structure nécessaire pour implémenter le circuit ciblé. On suppose, pour les comparaisons des architectures avec différents K et N, que la taille du FPGA est adaptable en fonction de la taille minimale nécessaire à la synthèse. On compare donc des FPGAs de taille différentes mais possédant la même capacité de synthèse.

Les figures 3.9 (a) (b) et 3.10 (a) (b) reprennent l'évolution des consommations statiques des différents blocs avec K et N. Ces tendances ont été déterminées pour un unique circuit. Après synthèse, on obtient D_X, D_Y et W minimum que l'on utilise dans le modèle 3.4.



(a) Evolution des délais intra- et inter-CLB moyen avec K et N.



(b) Evolution du pourcentage de délai intra-CLB moyen avec K et N.

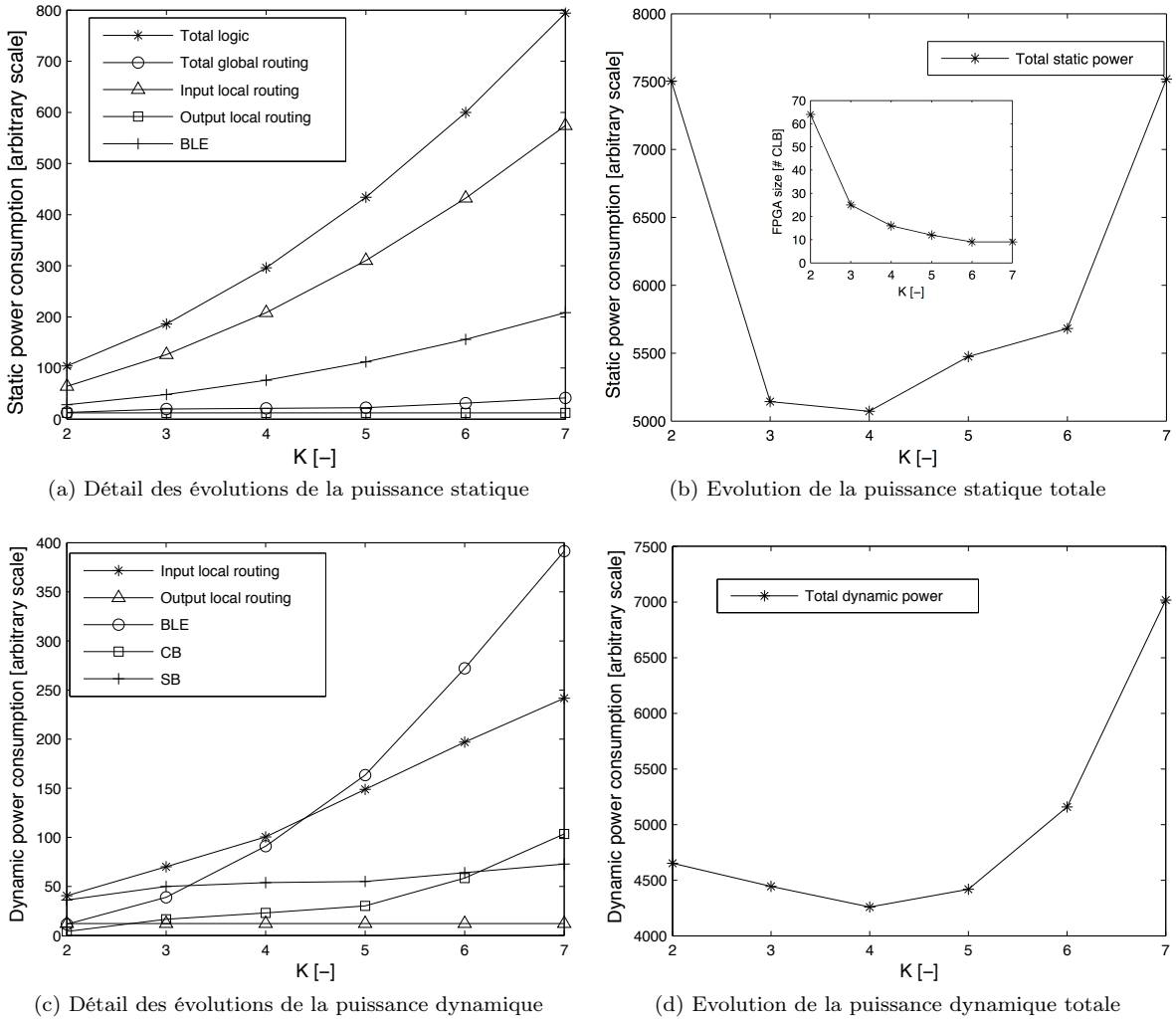
FIGURE 3.8: En haut : moyenne des délais sur le chemin critique après synthèse de 6 circuits du *benchmark*. En bas : répartition moyenne du délai pour les mêmes circuits.

Au niveau de la tuile, une augmentation de K provoque une augmentation de la taille de tous les multiplexeurs à l'exception des multiplexeurs du routage local de sortie. La Figure 3.9 (a) confirme cela et montre que l'augmentation la plus importante concerne le routage local d'entrée car l'augmentation de K provoque à la fois une augmentation de I pour maintenir l'accessibilité et une augmentation du nombre de multiplexeurs ($K \times N$). Au niveau du FPGA, l'augmentation de consommation est contrebalancée par la diminution du nombre de tuiles nécessaires pour synthétiser le circuit. La Figure 3.9 (b) montre que les K allant de 3 à 5 semblent se situer à l'optimum.

Lorsque N augmente (cf. Fig 3.10 (a) et (b)), toutes les contributions de consommation statique augmentent car la grappe grandit. Là aussi, c'est l'augmentation de consommation du routage local qui domine car N intervient dans le calcul de I ainsi que dans le nombre de multiplexeurs présents. La Figure 3.10 (b) montre cependant que la diminution du nombre de tuiles utilisées n'est pas suffisante pour compenser la hausse de la consommation avec N. Il ne semble pas y avoir d'optimum en N pour la puissance statique.

B. Modèle de consommation dynamique :

Dans un FPGA, deux informations sont nécessaires pour évaluer précisément la consommation



(a) Détail des évolutions de la puissance statique

(b) Evolution de la puissance statique totale

(c) Détail des évolutions de la puissance dynamique

(d) Evolution de la puissance dynamique totale

FIGURE 3.9: Evolution des consommations statiques et dynamiques d'un multiplieur RCA 8 bit synthétisé sur des architectures de K variant et N=4. Architecture de routage : unidirectionnel, *single-driver*, SB Wilton ($F_s = 3$).

dynamique d'un circuit. Comme chaque noeud n'est pas utilisé, il faut d'abord connaitre la position des ressources utilisées. Il faut ensuite connaitre la densité de transition pour chacun de ces noeuds. Ces deux informations sont difficiles à obtenir, elles demandent une modification importante des outils du *CAD Flow*. De plus, les méthodes déterministes de calcul des densités de transition demandent d'importantes ressources de calcul lorsque la taille des circuits et des structures reprogrammables augmente. Puisque seule l'évolution de la puissance dynamique avec K et N est ciblée ; certaines simplifications peuvent être réalisées. Pour éviter de devoir simuler le circuit en activité, on pose la densité de transition d'un noeud utilisé, constante et identique pour tous les noeuds utilisés. Pour éviter de devoir dresser une carte des noeuds utilisés sur tout le FPGA, on base le modèle sur les statistiques de sorties du *CAD Flow*. En plus de D_X, D_Y et W minimum, on obtient après synthèse :

- Le nombre moyen d'entrées des CLBs qui sont utilisées.
- Le pourcentage moyen de sorties des BLEs qui sont réutilisées localement sur les entrées des autres BLE dans un CLB.
- Le nombre moyen d'entrées des LUTs qui sont utilisées.
- L'occupation moyenne des canaux de routages selon l'orientation (X ou Y).

– Le nombre moyen de sorties des CLBs qui sont utilisées.
– Le nombre de BLEs utilisés par CLB

La puissance dynamique d'un multiplexeur est alors estimée en pondérant la taille du multiplexeur par le pourcentage de ses entrées qui sont utilisées. L'intuition ici est d'utiliser comme unité la consommation dynamique d'un multiplexeur 2 :1. Cette hypothèse est forte. En effet, selon la répartition des entrées, ce n'est pas parce que $\alpha\%$ de ses entrées sont utilisés que $\alpha\%$ de ses multiplexeurs 2 :1 vont effectuer une transition. Cependant, les résultats sont cohérents avec les modèles plus complexes présentés dans la Section 2.1.4.3. Il est également important de remarquer que, puisque les unités utilisées dans le modèle statique et dynamique ne sont pas les mêmes, il n'est pas possible d'obtenir le rapport de consommation statique et dynamique pour la structure.

Tout comme pour le modèle statique, on distingue trois contributions dans le CLB :

1. Routage local d'entrée : $KN(I+N-1)\frac{\alpha}{(I+N)}$, avec α, la somme du nombre moyen d'entrées des CLBs utilisées et du produit de N et du pourcentage de sorties des BLEs réutilisées localement. Cette contribution est la somme de KN multiplexeurs de $I+N$ entrées dont seule α sont utilisés.
2. Routage local de sorties : $N(N-1)(\#\_BLE\_used)/N$. Résultat de N multiplexeurs de N entrées dont seules celles qui sont reliées aux sorties des BLEs utilisés sont susceptibles d'effectuer des transitions.
3. BLE : $N(\beta^2-1)$, avec β, le nombre moyen d'entrées des LUTs utilisées. Pour les LUTs, le calcul est différent car les entrées de la LUT sont les signaux de sélection des multiplexeurs. De plus, aucune information sur les bits de configuration n'est supposée connue. Un bon algorithme de placement forcera les entrées non-utilisées au début de la LUT. Une entrée non utilisée réduit donc de 1 la taille effective de la LUT selon le modèle développé.

On distingue deux contributions dans le routage global :

1. CB : $I/2(F_{c,in}W-1)\frac{F_{c,in}W\times avg\_occ}{F_{c,in}W}$, avec avg_occ, l'occupation moyenne des canaux toute direction confondue. Cette contribution comptabilise les $I/2$ multiplexeurs vers les entrées du CLB (on suppose que les entrées sont réparties sur tout le pourtour du bloc). Ces multiplexeurs possèdent $F_{c,in}\times W$ entrées dont $F_{c,in}W\times avg\_occ$ sont actives.
2. SB : $W/2(F_s+N/2-1)(F_s\frac{avg\_occ}{W}+\frac{\delta}{2})$, avec δ, le nombre moyen de sorties des CLBs utilisés. Si les segments ont une longueur de 1 et le SB a une flexibilité $F_s=3$, on trouvera $W/2$ multiplexeurs (pour des segments unidirectionnels *single-driver*) de $F_s+N/2$ entrées. La probabilité que les F_s segments soient actifs est donnée par $F_s(avg\_occ/W)$ et la probabilité que les sorties du CLB soient actives est donnée par $\delta/2$ car seule la moitié des sorties arrivent sur le SB de la tuile.

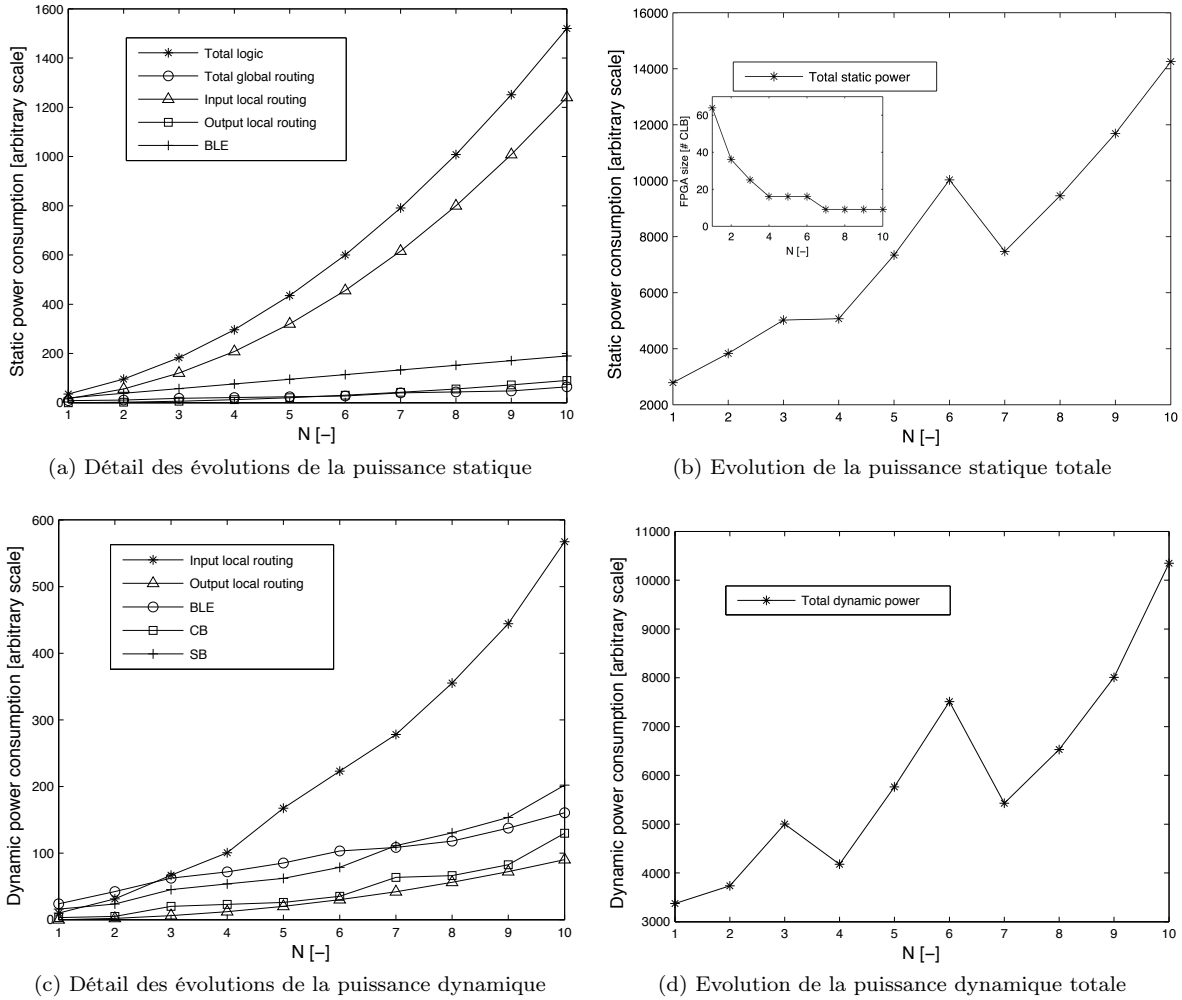
La puissance dynamique totale du FPGA selon ce modèle s'écrit donc :

$$\begin{aligned} P_{dyn,FPGA} &= D_X D_Y \times [KN(I+N-1)\frac{\alpha}{(I+N)} + N(N-1)\frac{\#\_BLE\_used}{N} + N(\beta^2-1) \\ &\quad + \frac{I}{2}(F_{c,in}W-1)\frac{F_{c,in}W\times avg\_occ}{F_{c,in}W} + \frac{W}{2}(F_s+N/2-1)(F_s\frac{avg\_occ}{W}+\frac{\delta}{2})] \quad (3.5) \end{aligned}$$

avec, D_X et D_Y, les dimensions de la structure nécessaire pour implémenter le circuit ciblé.

Les figures 3.9 (c) (d) et 3.10 (c) (d) reprennent l'évolution des consommations dynamiques des différents blocs avec K et N.

On constate sur la Figure 3.9 (c) que la consommation dynamique des LUTs augmente plus vite avec K que celle du routage local. Ce phénomène, combiné avec la réduction des dimensions du FPGA lorsque K augmente, fait apparaitre un optimum pour la puissance dynamique lorsque K est situé entre 3 et 5 (cf. Fig : 3.9 (d)).



(a) Détail des évolutions de la puissance statique

(b) Evolution de la puissance statique totale

(c) Détail des évolutions de la puissance dynamique

(d) Evolution de la puissance dynamique totale

FIGURE 3.10: Evolution des consommations statiques et dynamiques d'un multiplieur RCA 8 bit synthétisé sur des architectures de N variant et K=4. Architecture de routage : unidirectionnel, *single-driver*, SB Wilton ($F_s = 3$).

Lorsque N augmente (cf. Fig : 3.10 (c) (d)), on constate que la consommation dynamique du routage local d'entrée domine les autres contributions. Comme lors de l'étude de la consommation statique, la réduction de la taille de la structure ne compense pas la hausse de consommation induite par l'augmentation de la taille des grappes.

3.2.3 Description du CLB sélectionné

Sur base des résultats des modèles décrits dans les Sections 3.2.2.2 et 3.2.2.3, une architecture de CLB est proposée.

Au niveau des performances en vitesse, l'augmentation de K et N semble être toujours profitable. Le modèle de la puissance indique que le K optimal est situé entre 3 et 5. Pour ce travail, K est fixé à 4, ce qui est également cohérent avec l'état de l'art de la Section 2. L'optimisation de N ne débouche pas sur un optimum clair comme c'est le cas pour K. Les performances en vitesse augmentent avec N mais la consommation statique et dynamique augmente également. Dans l'optique d'une structure ULP soumis à des contraintes de performances relâchées, N est fixé à 4.

La Figure 3.11 présente la structure du routage local utilisé dans la suite de ce travail. Pour atteindre

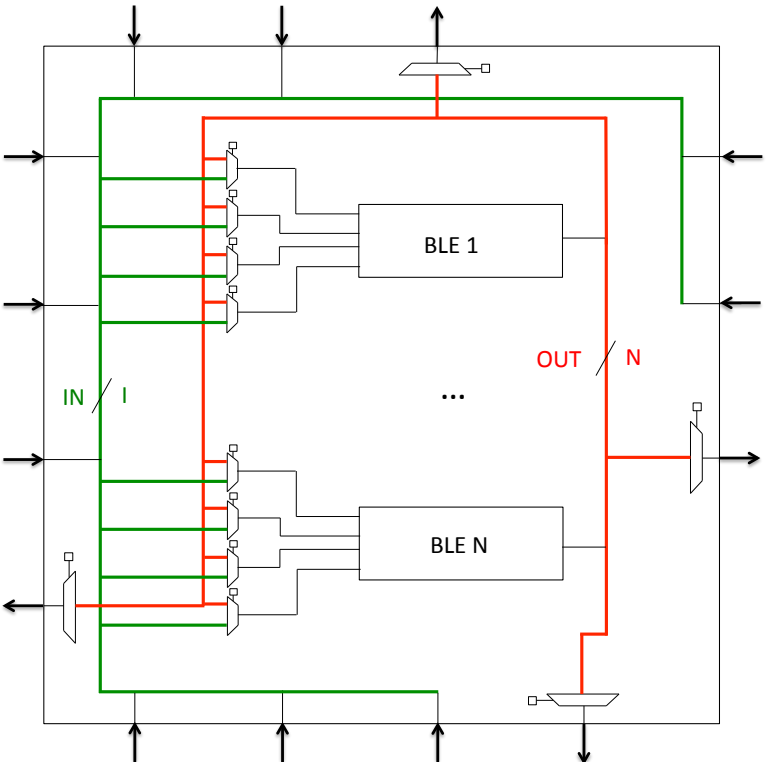


FIGURE 3.11: Architecture de CLB développée dans ce travail. Le routage local riche en multiplexeurs, permet de chainer les BLEs à la suite pour pouvoir placer dans la même grappe deux BLEs liés.

une grande flexibilité, un bouclage des sorties des BLEs sur les entrées est possible. De plus, les entrées et sorties sont réparties sur le pourtour du CLB et les pins de sorties peuvent être commandées par tous les BLEs.

3.3 Choix de l'architecture de routage

L'élaboration de l'architecture de routage passe par le choix de différents paramètres telles que la topologie des SBs ou les flexibilités d'entrée et de sortie. Les trois choix les plus importants portent sur la longueur des segments, la directionalité des segments et sur le nombre de *driver* par segment.

3.3.1 Longueur des segments

La distribution des segments est soumise à un compromis : si les segments sont courts, les connexions longues seront composées d'une succession de petits segments et de *drivers* d'où une vitesse réduite. Si les segments sont longs, ils imposeront des détours lors de l'implémentation des petites connexions d'où une vitesse réduite et des contraintes supplémentaires sur le routage.

La Figure 3.12 présente l'évolution du délai moyen sur le chemin critique avec L pour deux circuits du *benchmark*. On constate que choisir $L = 4$ permet de réduire le délai.

On peut remarquer que la situation optimale consisterait à répartir les longueurs des segments pour correspondre à la distribution de longueurs des connexions des circuits synthétisés. Dans ce travail, les distributions des longueurs à travers le FPGA ont été considérées uniformes. Il serait intéressant de considérer, par exemple, une distribution de 50% de segments de longueur 4 et de 50% de segments de longueur 8 uniformément sur la surface. Ou encore de répartir les segments longs sur les bords du FPGA et les segments courts dans le centre du FPGA comme c'est le cas dans certains FPGAs commerciaux.

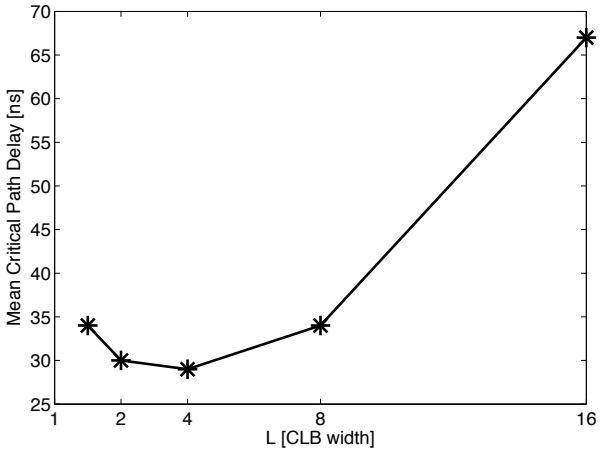


FIGURE 3.12: Evolution du délai moyen sur le chemin critique (*benchmark :* ALU4 et elliptic encryption) avec la longueur des segments. Tous les segments des canaux possèdent la même longueur.

3.3.2 Directionnalité des segments

Comme décrit dans la Section 2.1.2, il existe différentes architectures de segments. Les segments bidirectionnels, composés de deux *tri-state drivers* inversés (cf. Fig. 3.13) peuvent être programmés pour envoyer un signal dans les deux directions. Le *driver* qui ne pointe pas dans la bonne direction est placé en haute impédance. Cela laisse 50% des *drivers* inutilisés après programmation. Les segments unidirectionnels ne possèdent qu'un seul *tri-state driver* et ne peuvent donc être programmés que pour envoyer un signal dans une seule direction. En faisant l'économie d'un *driver*, ce type d'architecture permet d'atteindre une vitesse plus importante pour la même surface car le *driver* restant peut être agrandi. Cependant, pour atteindre le même potentiel de connexion, il faut deux fois plus de segments unidirectionnels qu'il n'en faudrait de bidirectionnels.

Il est important de remarquer que doubler la largeur du canal lors du passage de segments bidirectionnels à segments unidirectionnels est généralement superflu. En effet, en bidirectionnel, un seul signal occupe le segment tandis qu'en unidirectionnel, il est possible d'envoyer deux signaux indépendants pour un seul segment équivalent en bidirectionnel. Il y aura un "partage" de segments qui permet de ne pas doubler la largeur des canaux pour une même capacité de routage.

(a) Bidirectionnel (b) Unidirectionnel

FIGURE 3.13: Illustration du concept de directionnalité des segments.

La Figure 3.14 illustre l'architecture de routage bidirectionnelle. Pour éviter de devoir implémenter plusieurs SBs différents suivant la position des segments qui s'arrêtent par rapport aux segments qui continuent, on opère une rotation des segments qui sortent du SB. On peut alors maintenir la position des points de connexion du SB au même endroit. Les segments sont groupés par groupe de L et les rotations sont effectuées au sein de chaque groupe (cf. Fig. 3.14). Cette figure présente également le détail d'implémentation de la connexion. Pour un signal arrivant, trois peuvent repartir ; la flexibilité du SB conçu est donc de 3. Chaque *driver* possède la fonctionnalité *tri-state* pour pouvoir être placé en haute impédance si un signal doit être émis plus loin sur le segment. Les multiplexeurs sélectionnent la source du signal qui sera transmis au *driver*. Pour chaque segment sortant, la source peut être l'un

des trois segments qui s'interrompent dans le SB, soit le signal d'un autre segment du groupe qui ne s'arrête pas dans le SB (voire même d'un segment n'appartenant pas au même groupe).

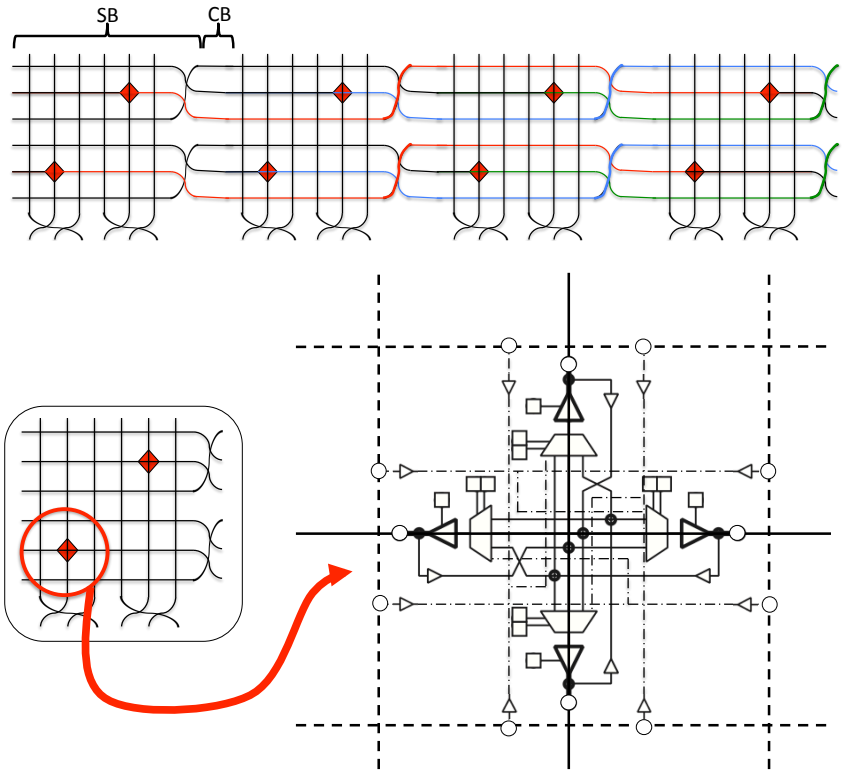


FIGURE 3.14: Illustration d'un point de connexion d'un SB bidirectionnel pour $L = 3$. Le croisement des segments permettant de réutiliser la même topologie pour tous les SBs est présenté. Les segments pointillés ne sont pas interrompus lors du passage dans le SB. Les connexions pointillé-point représentent la possibilité d'utiliser un signal passant sur un segment non-interrompu pour l'envoyer dans un segment qui démarre dans le SB (*mid-point connections*). Les petits triangles représentent les *sense buffers* et les grands triangles représentent les *tri-state drivers* avec leur bit de configuration.

Il est évident que l'implémentation des connexions entre le milieu d'un segment et le début d'un autre augmente la flexibilité et diminue la contrainte sur l'algorithme de routage en enrichissant le graphe des ressources de routage (cf. Section 3.1). Cependant, l'implémentation de ces connexions complexifie le SB et augmente la charge sur chaque segment, ce qui le ralentit.

La Figure 3.15 illustre l'architecture de routage unidirectionnelle avec le même croisement. Le nombre de segments a été doublé pour maintenir le même nombre de segments dans chaque direction. Les segments sont groupés par $2 \times L$. Les *drivers* utilisés dans la configuration unidirectionnelle peuvent être *tri-state* ou normaux (cf. Section 3.3.3). L'implémentation d'un point de connexion unidirectionnel est similaire au cas bidirectionnel à l'exception des *sense buffers* qui sont séparés des *drivers*. Le compromis entre flexibilité et chargement est également d'application.

3.3.3 Organisation des *drivers*

La répartition des *drivers* sur les segments est un autre facteur important de la structure de routage. Le segment peut être *multi-driver* ou *single-driver* (cf. Fig. 2.9). Si plusieurs *drivers* coexistent, on peut lancer le signal sur le segment de plusieurs endroits, ce qui augmente la flexibilité mais tous les *drivers* doivent être *tri-state* pour pouvoir être mis en haute-impédance lorsqu'ils ne sont pas maître du segment.

Lorsqu'il n'y a qu'un seul *driver* par segment, il ne doit pas implémenter la fonctionnalité *tri-state*. Pour le même niveau de performances, on peut donc diminuer sa taille et donc sa consommation. Cette

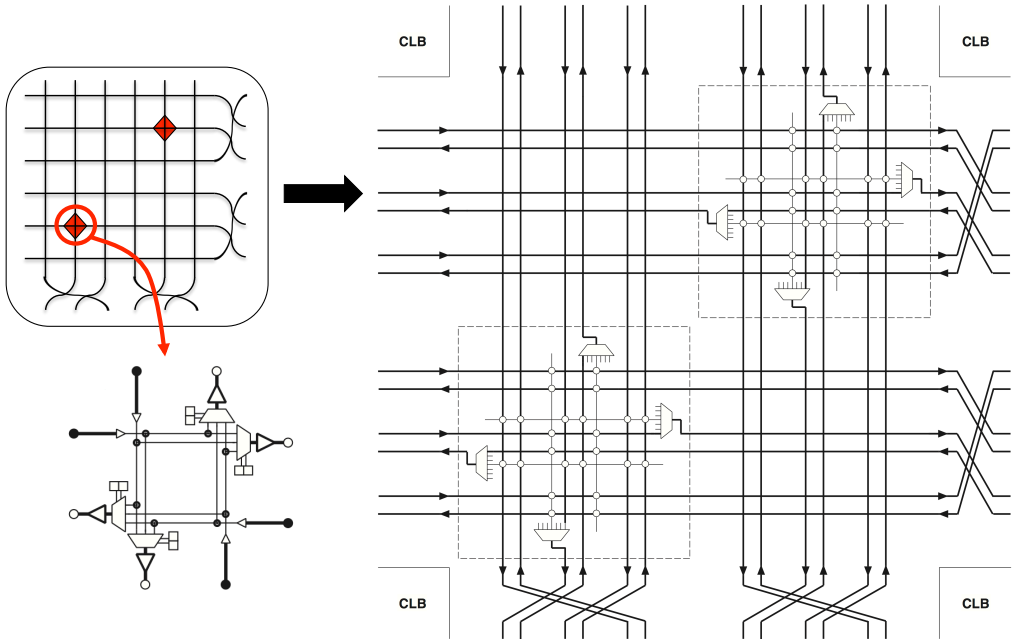


FIGURE 3.15: Illustration d'un SB unidirectionnel pour $L = 3$ [22]. A droite : le point de connexion unidirectionnel. A gauche : position des *mid-point connections*. Toutes les entrées des multiplexeurs sont représentées par les cercles blancs. Les *sense buffers* et *driver* présents avant et après les multiplexeurs ne sont pas représentés.

solution n'est réalisable que sur une structure unidirectionnelle et les *drivers* sont situés dans les SBs, à l'origine des segments (cf. Fig. 3.15). Il faut également ramener les sorties des CLBs vers le SB le plus proche pour les connecter aux multiplexeurs de sélection du signal à envoyer. Le passage de *multi-driver* à *single-driver* complexifie la structure de routage.

3.3.4 Segments bidirectionnels *multi-driver* et segments unidirectionnels *single-driver* : comparaison

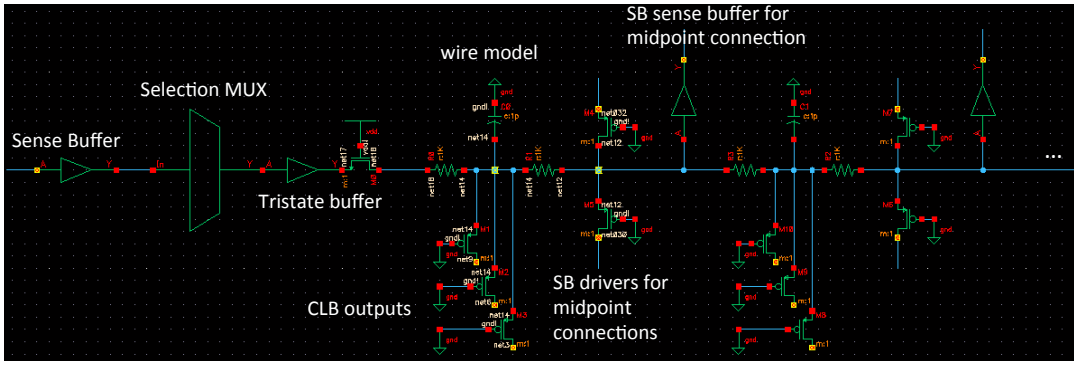
Pour pouvoir comparer les performances en vitesse des deux types d'implémentation de segments, deux modèles ont été développés. La Figure 3.16 les présente. Ils reprennent toutes les connexions que le segment rencontre en partant du *sense buffer*.

Le segment bidirectionnel de la Figure 3.16 (a) démarre au *sense buffer* puis passe dans le petit multiplexeur qui sélectionne le signal à envoyer. Le *tri-state buffer* est constitué d'un inverseur et d'un NMOS qui permet de le placer en haute-impédance si sa grille est à '0'. Le segment qui s'étend sur la longueur d'un CLB est modélisé par une résistance et une capacité équivalente. Au centre de ce modèle de segment, on trouve les N sorties du CLB qui peuvent également être maître sur le segment. Lorsqu'elle ne le sont pas, le NMOS de la fonctionnalité *tri-state* charge le segment. Après avoir passé le CB, le segment entre dans le second SB où il est chargé par les drivers qui permettent des *mid-point connections* ainsi que par les *sense buffers* qui permettent d'envoyer ce signal sur d'autres segments prenant leur origine dans ce SB. Ce modèle est reproduit L fois.

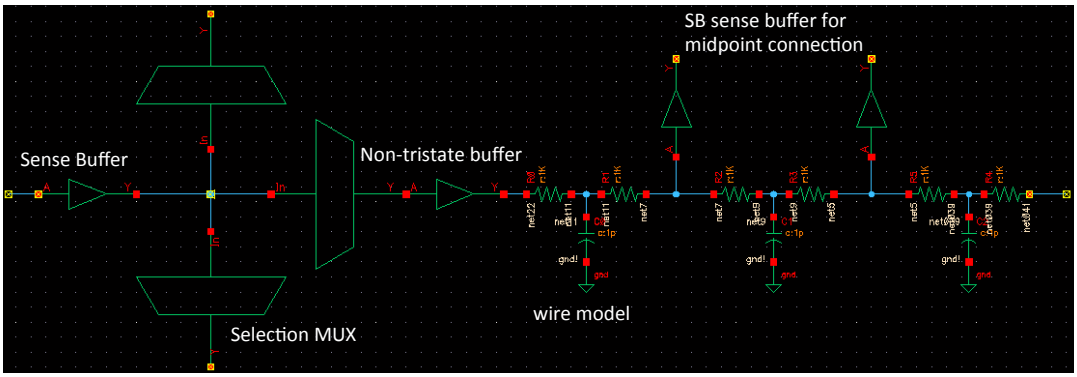
Le début du segment unidirectionnel de la Figure 3.16 (b) est similaire à celui du segment bidirectionnel à l'exception du transistor de *tri-state* qui disparait et au multiplexeur de sélection qui est plus grand car il doit permettre la sélection parmi toutes les sorties des CLBs voisins. La disparition de tous les NMOS de *tri-state* partout sur le segment diminue considérablement la charge répartie.

Les simulations physiques montrent que le segment unidirectionnel *single-driver* est 55% plus rapide que le segment bidirectionnel *multi-driver*. Cette réduction de délai s'explique par la réduction de

la charge distribuée sur le segment. C'est donc cette topologie de routage qui est sélectionnée pour l'implémentation du Chapitre 4.



(a) Bidirectionnel *multi-driver*.



(b) Unidirectionnel *single-driver*.

FIGURE 3.16: Modèles des segments utilisés pour la comparaison des délais entre les deux configurations.

Chapitre 4

Implémentation ULP-ULV

Ce chapitre reprend la description de l'implémentation des différents blocs du FPGA. Les résultats de la comparaison du style logique le plus adapté à l'implémentation des multiplexeurs sont présentés. Pour chaque bloc, des simulations physiques sont réalisées pour valider leurs fonctionnalités et pour caractériser leurs performances. La possibilité d'ajouter un dispositif de power gating est ensuite évaluée. Ce type de dispositif permet en effet d'effectuer de larges économies de puissance statique si un pourcentage suffisamment important des structures non-utilisées du FPGA en est équipé. Une technique de contrôle du compromis performance/vitesse au travers d'une variation du V_T de chaque bloc est ensuite présentée. L'optimisation de la répartition du V_T est réalisée pour une tension d'alimentation de $0.5V$ qui permet de réaliser d'importantes économies de consommation statique. Les performances du FPGA conçu sont ensuite évaluées et comparées à des implémentations similaires en ASIC et sur un microcontrôleur.

4.1 LUT

La *Look-Up Table* permettant l'implémentation des différentes fonctions logiques est un élément critique du FPGA.

La LUT est un multiplexeur dont les entrées sont les signaux de sélection et dont les signaux sélectionnés vers la sortie proviennent des bits de configuration (cf. Fig. 4.1). Le nombre d'entrées de la LUT fixe le nombre d'étages de multiplexeurs 2 : 1 nécessaire pour l'implémentation. Il faut K^2 bits de configuration et $K^2 - 1$ multiplexeurs 2 : 1 pour chaque LUT. L'évaluation de la meilleure implémentation pour la LUT passe donc par l'optimisation du multiplexeur 2 : 1.

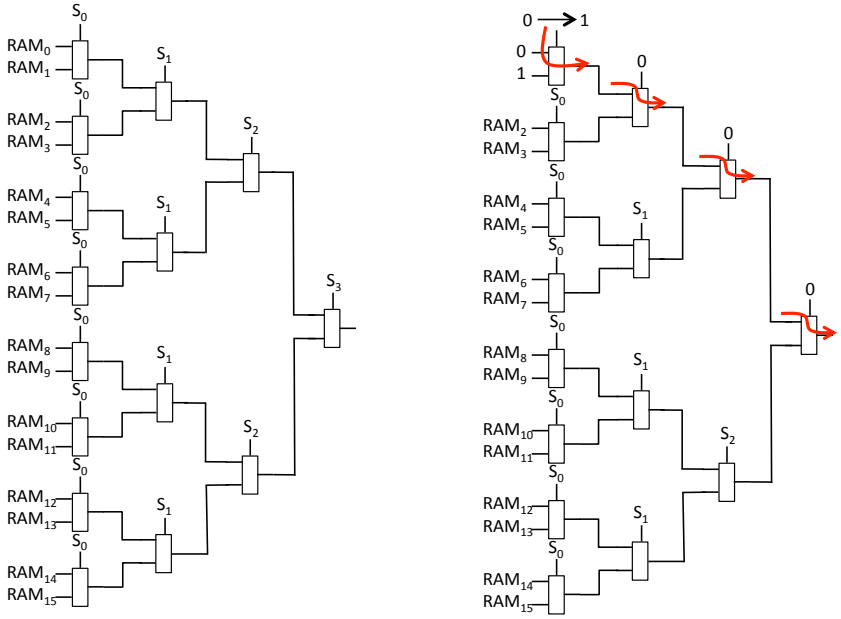


FIGURE 4.1: A gauche : illustration de l'architecture d'une LUT à 4 entrées. Les entrées sont les signaux de sélection $S_0 - S_3$ et les bits de configuration sont notés $RAM_0 - RAM15$. A droite : illustration du chemin critique à travers la LUT après *static timing analysis*.

Dans ce travail, deux types de logique ont été évalués pour déterminer quelle était l'implémentation qui permet d'atteindre le meilleur rapport entre le délai et la puissance consommée. Les deux types de logique sélectionnés sont la logique CMOS statique et la logique de transmission (*Transition Gate Logic*) dérivée de la logique de passage. Les deux architectures de multiplexeurs sont présentées à la Figure 4.2.

Les grilles de tous les transistors ont été fixées à 80 nm pour diminuer le courant de fuite. Les autres transistors ont été dimensionnés pour minimiser l'écart entre les délais de descente et de montée du noeud de sortie d'un multiplexeur 2 : 1.

Pour évaluer les performances des deux multiplexeurs 2 : 1, un *testbench* a été développé (cf. Fig. 4.3). Sur base d'une description des multiplexeurs, une LUT de 4 entrées est assemblée. Pour pouvoir effectuer des simulations réalistes, il faut imposer des conditions réalistes aux entrées et sorties. Pour cela, deux inverseurs sont placés aux entrées pour obtenir une pente de transition réaliste et aux sorties pour charger la sortie de la LUT. Toutes les simulations physiques sont réalisées par le simulateur ELDO sur la technologie 65nm de STMicroelectronics.

L'évaluation de la puissance consommée par la LUT est réalisée en imposant des transitions sur les entrées à 50MHz. Bien que la fréquence à imposer dans un FPGA dépend du circuit synthétisé, on cible 50MHz comme ordre de grandeur. Cette fréquence cible s'aligne sur un objectif d'intégration dans des EAS ambitieux dont les exigences en vitesse sont élevées. La puissance dynamique est évaluée en

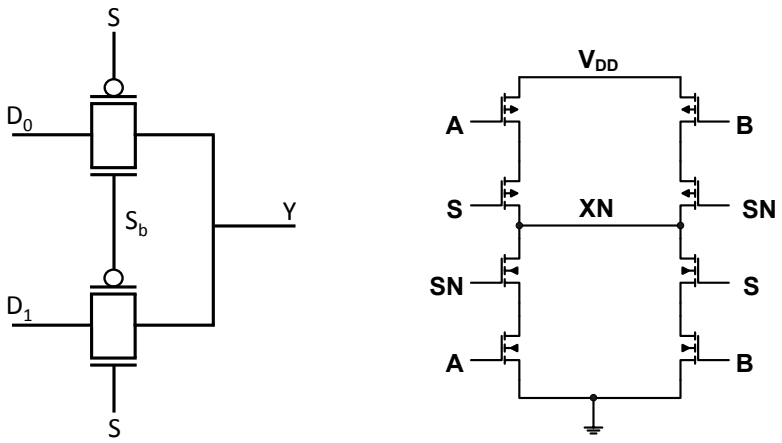


FIGURE 4.2: Deux implémentations d'un multiplexeur 2 : 1. A gauche : implémentation en logique de transmission. A droite : implémentation en logique CMOS statique.

effectuant la moyenne sur 10 transitions du courant total consommé et en la multipliant par la tension d'alimentation. La puissance statique est évaluée en multipliant le courant consommé lorsqu'aucune transition n'est présente avec la tension d'alimentation.

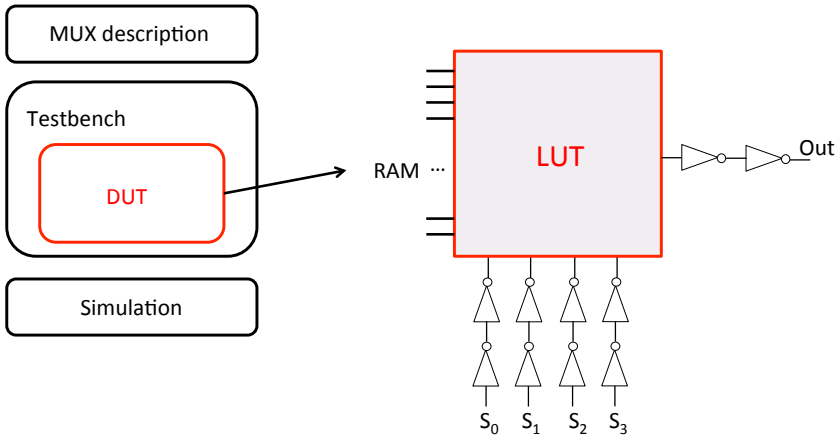


FIGURE 4.3: Schéma du *testbench* développé.

La mesure des délais de propagation d'un signal à travers la LUT est moins standard car il faut s'assurer que c'est bien le pire cas qui est mesuré. Pour obtenir une estimation précise des transitions des entrées qui génèrent le plus grand délai en sortie de la LUT, une *Static Timing Analysis* (STA) est réalisée. On commence par analyser les multiplexeurs 2 : 1. Deux situations sont à distinguer :

- Transition sur le signal de sélection, les entrées sont constantes.
- Transition sur un signal d'entré, le signal de sélection est constant.

Pour chacune de ces situations, les combinaisons entrée/sélection sont testées exhaustivement et les délais entre la transition d'entrée ou de sélection et la sortie sont mesurés. Cette étude est réalisée à une tension intermédiaire ($0.6V$) et permet de repérer le ou les chemin(s) les plus long(s) à travers le multiplexeur. On observe alors que les cas suivants :

- Une transition sur le signal de sélection lorsque les deux entrées sont différentes
- Une transition sur une entrée lorsqu'elle est sélectionnée

provoquent environ le même délai sur la sortie. La Figure 4.1 illustre le chemin critique qui sera utilisé pour mesurer les délais dans la LUT.

La Figure 4.4 reprend les résultats des mesures de la puissance dynamique, statique, du délai ainsi que du produit entre délai et puissance totale en fonction de la tension d'alimentation. On constate que la LUT implémentée en logique de transmission est plus performante que l'implémentation en logique CMOS statique. Cette observation est d'autant plus vraie à mesure que V_{DD} diminue et devient inférieur à la tension de seuil comme l'illustre la Figure 4.4 (d). Le produit délai-puissance totale de l'implémentation CMOS statique est au minimum 2 fois plus élevé que celui de l'implémentation TGL voire plus lorsque $V_{DD} < V_T$.

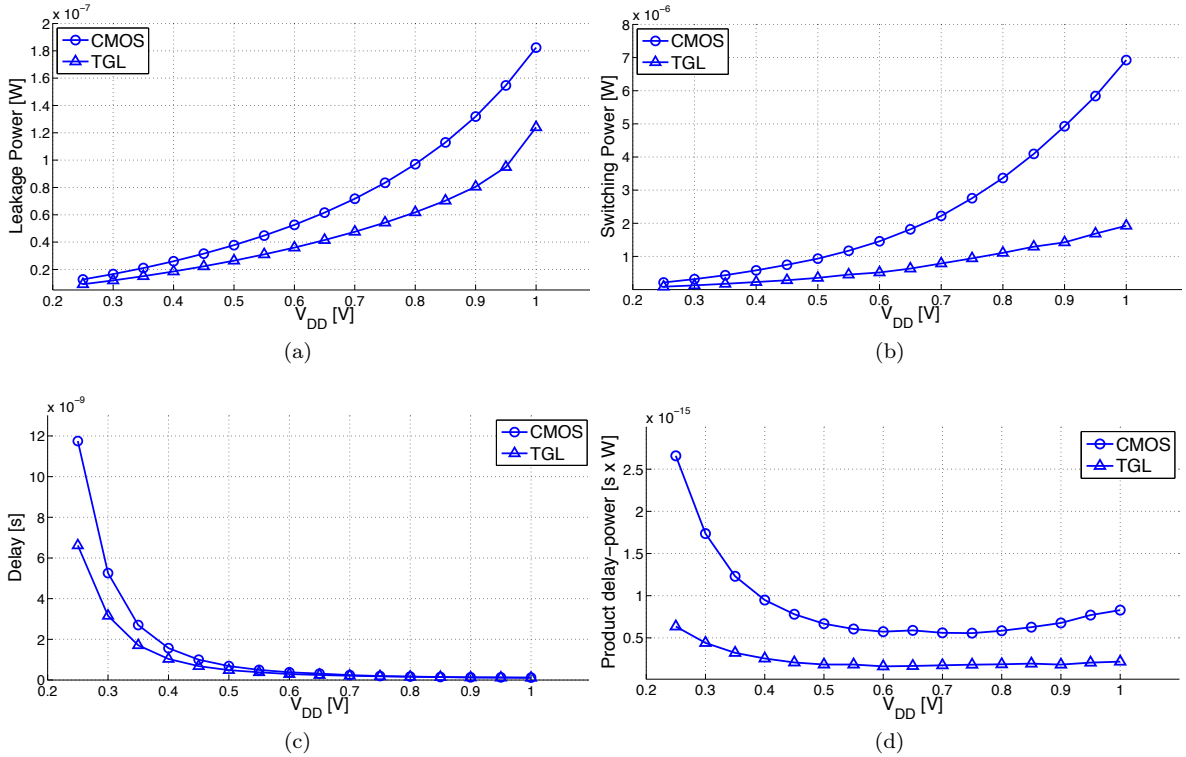


FIGURE 4.4: Comparaison des performances d'une *Look Up Table* de 4 entrées implémentées en logique standard CMOS et en logique de transmission TGL. On constate que, selon tous les facteurs de mérite, l'implémentation TGL semble plus performante. Tous les transistors sont de type *General Purpose* (GP) et de V_T standard. Les grilles ont été allongées à $80nm$.

Si on fixe 5 comme moyenne du nombre de LUTs sur le chemin critique des circuits synthétisables et qu'on estime que le délai intra-CLB équivaut environ à 40% du délai total sur un chemin, on peut estimer la tension d'alimentation minimale pour atteindre la contrainte de $50MHz$. En effet, si on estime que le délai dans le routage local est environ équivalent au délai de la LUT, il faut que le délai de celle-ci n'excède pas $1ns$ pour respecter la contrainte. L'implémentation en logique de transmission permet d'atteindre cet objectif pour $V_{DD} = 0.4V$ alors que la logique CMOS statique ne l'atteint que pour $V_{DD} = 0.45V$.

Après avoir fixé l'architecture des multiplexeurs 2 : 1, il faut déterminer le type de MOS le plus adapté. La Figure 4.5 illustre l'évolution des délais et des puissances consommées par un multiplexeur avec la tension d'alimentation pour deux différentes épaisseurs d'oxyde. Les MOS *Low-Power* (LP) possèdent une épaisseur d'oxyde importante, ce qui se traduit par une hausse de la capacité d'oxyde et donc par une baisse de la pente sous-seuil comme indiqué par l'expression 2.9. Si S diminue, $1/S$ augmente et le rapport I_{ON}/I_{OFF} augmente. Cette augmentation permet de diminuer le courant de fuite et donc la consommation statique du multiplexeur. On constate sur la Figure 4.5 (a) que la consommation statique du multiplexeur LP est inférieure de deux ordres de grandeur à celle du multiplexeur

TABLE 4.1: Valeur des tensions de seuils ([V]) mesurées à $V_{DD} = 0.5V$ selon le dopage du canal et l'épaisseur de l'oxyde

	GP	LP
LV_T	0.315	0.510
SV_T	0.378	0.600
HV_T	0.392	0.739

GP. L'évolution de la puissance dynamique est en revanche semblable pour les deux épaisseurs d'oxyde (Fig. 4.5 (b)).

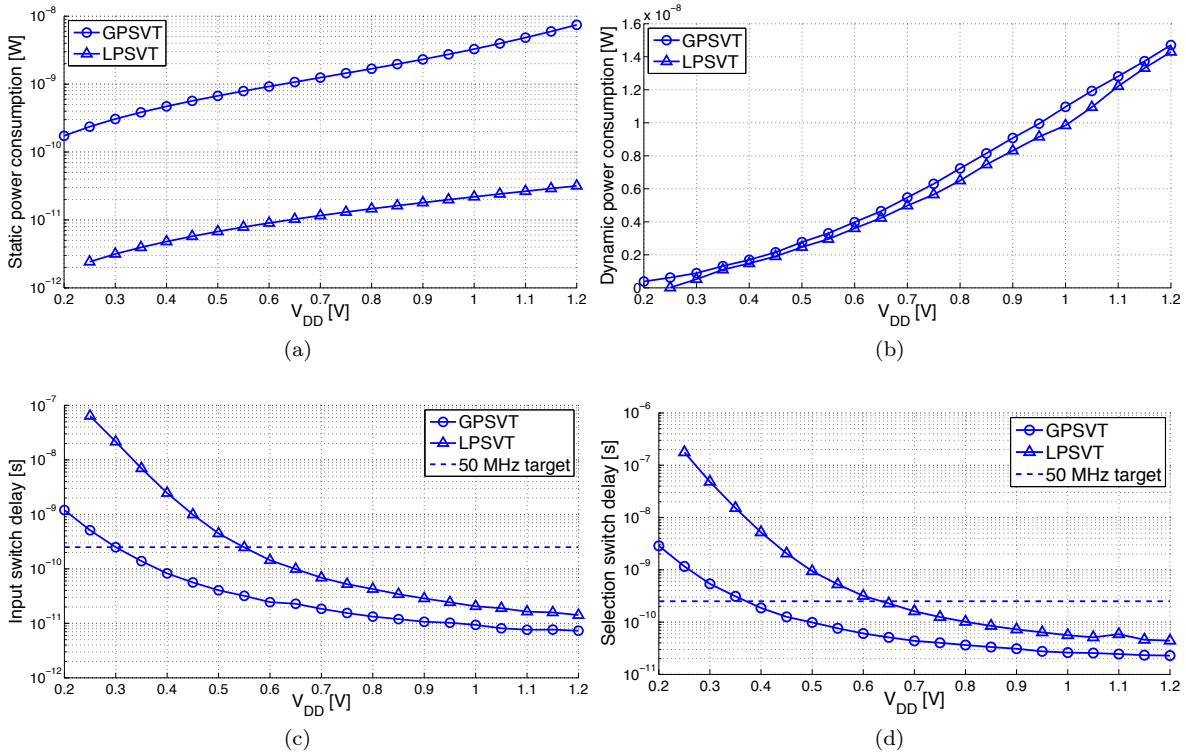


FIGURE 4.5: Comparaison des performances d'un multiplexeur 2 : 1 implémenté en logique de transmission pour deux épaisseurs d'oxyde. En estimant qu'il y a 4 multiplexeurs par LUT et 5 LUTs sur un hypothétique chemin critique, la limite de délai maximum pour atteindre la contrainte de $50MHz$ a été indiquée (ligne pointillée).

L'augmentation de l'épaisseur d'oxyde provoque une hausse importante de la tension de seuil du transistor. La Table 4.1 reprend l'évolution du V_T pour tous les dopages proposés par STMicroelectronics (LV_T, SV_T et HV_T) lorsque l'on passe de GP à LP. Cette hausse de V_T dégrade le courant I_{ON} lorsque la tension d'alimentation diminue, ce qui provoque une augmentation des délais. Les figures 4.5 (c) et (d) montrent que l'implémentation utilisant des transistors LP souffre d'un handicap en vitesse considérable par rapport à l'implémentation utilisant des transistors GP. Pour maintenir un même niveau de performances, il faut augmenter de $200mV$ la tension d'alimentation de l'implémentation LP. Cette augmentation se traduit par une consommation dynamique deux fois plus importante illustrée dans la Figure 4.5 (b). Une analyse détaillée du choix des types de transistor est réalisée dans la Section 4.5 ; on peut d'ores et déjà constater que l'utilisation de transistors LP plus lents nécessite une augmentation du V_{DD} et donc une augmentation de la consommation dynamique qui compense la

diminution de la consommation statique.

4.2 BLE

Le *Basic Logic Element* est composé d'une LUT de 4 entrées suivie d'un registre et d'un multiplexeur pour sélectionner la sortie du registre ou la sortie de la LUT. La Figure 4.6 illustre le fonctionnement du BLE. Le signal V_{S0} impose la transition à la sortie de la LUT visible sur le signal $V_{output\ LUT}$. Le signal sortant de la LUT est inversé car la transition des registres de configuration vers les entrées de la LUT est séparée par des inverseurs. Ces inverseurs permettent de séparer les circuits de configuration à haut-V_T dont le courant I_{ON} est faible, de la partie de FPGA effectuant des transitions qui a besoin d'un courtant I_{ON} important.

La sortie du BLE simulée dans la Figure 4.6 a été connectée sur la sortie du registre et on peut constater le décalage jusqu'au flanc montant suivant de l'horloge.

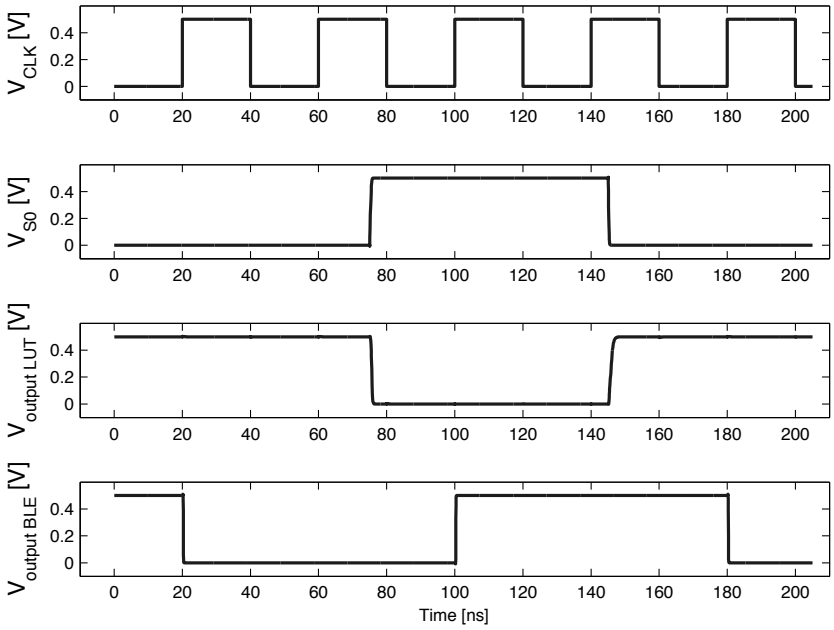


FIGURE 4.6: Simulation de fonctionnement d'un BLE.

4.3 CLB

Le CLB illustré dans la Figure 3.11 a été simulé avec ELDO. Un registre à décalage chainant les DFF des BLEs a été synthétisé sur le CLB et le résultat est présenté sur la Figure 4.7. Les signaux $V_{OUT\ BLE\ 1}$, $V_{OUT\ BLE\ 2}$ et $V_{OUT\ BLE\ 3}$ sont les sorties des registres des BLEs. On peut constater que l'entrée du CLB est bien décalée d'un coup d'horloge après chaque BLE.

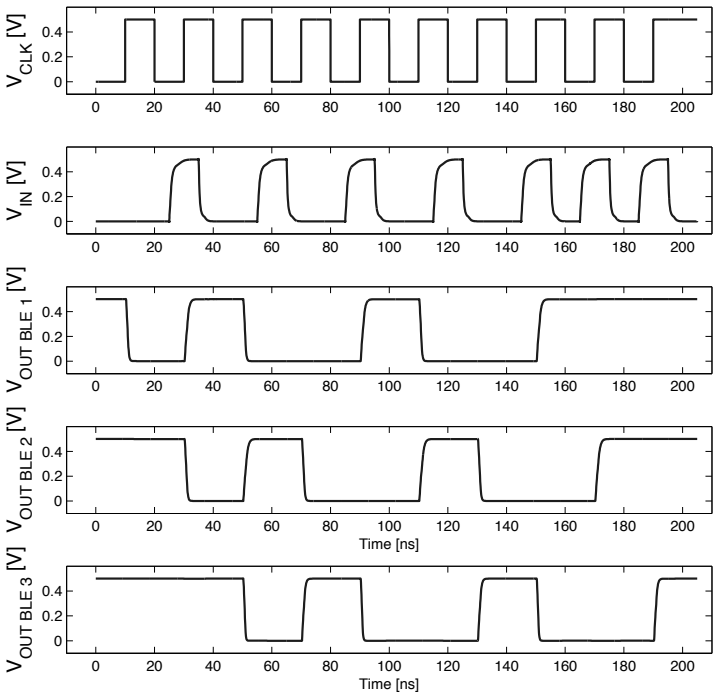


FIGURE 4.7: Simulation de fonctionnement d'un CLB. Un registre à décalage chaînant les 4 registres des 4 BLEs du CLB a été synthétisé.

4.4 *Power gating*

Le *power gating* permet de limiter la consommation statique du FPGA en réduisant le courant de fuite consommé par les blocs qui ne sont pas actifs. Le *power gating* d'un ASIC passe par un examen détaillé de l'activité de chaque partie qui le constitue. Une fois que l'on a déterminé les périodes pendant lesquelles le circuit ou un sous-circuit est inactif, on peut concevoir un *switch* qui coupe son alimentation. Le *power gating* d'un FPGA est basé sur des considérations différentes. Après synthèse et configuration, la vaste majorité des ressources de logique ou de routage n'est pas utilisée. La réduction des courants de fuite dans ces sections permettrait donc de rapprocher les performances du FPGA de celle d'un ASIC qui ne dispose que des ressources indispensables à l'implémentation du circuit.

4.4.1 Types de *power gating*

Il existe différentes manières de distribuer les *switch* d'alimentation dans la structure. Du plus grossier au plus fin :

- FPGA : on peut couper l'alimentation de toute la structure lorsqu'elle n'est pas active. Il faut cependant prendre garde aux états des registres internes qui doivent être préservés dans la structure ou sauvegarder à l'extérieur (entrainant une pénalité lors du redémarrage pour rapatrier ces valeurs dans le FPGA). La même remarque s'applique pour la configuration si elle est volatile. Il est cependant plus facile de maintenir l'alimentation des cellules de configuration si elles sont implémentées avec des transistors $HVTLP$ présentant un faible courant de fuite.
- Tuile : on peut envisager de couper l'alimentation dans une tuile mais cette situation n'est acceptable que si une large portion de la structure n'est pas utilisée. Si la structure n'est pas largement surdimensionnée par rapport à la taille moyenne des circuits à synthétiser, il sera rare d'observer une tuile entière non utilisée.
- CLB : on peut couper l'alimentation de toutes les grappes inutilisées.
- BLE : on peut couper l'alimentation de tous les blocs élémentaires inutilisés.
- Routage global & local : une fois configuré, un multiplexeur de routage de X entrées (composé

de X-1 multiplexeurs 2 : 1) ne comporte que $\sqrt{X}$ multiplexeurs 2 : 1 actifs. Deux solutions pour réduire le courant de fuite dans ces multiplexeurs sont proposées dans la Figure 4.8.

Plus le *power gating* est implémenté avec un grain fin, plus l'économie de puissance statique sera importante et plus le mimétisme des performances énergétiques d'un ASIC sera élevé. Cependant, le *power gating* à fin grain représente un challenge technologique important car il dégrade les performances en vitesse de la structure.

La Figure 4.8 représente deux types de répartition des *switch* pour les multiplexeurs de routage. Dans la Figure 4.8 (a), seule une moitié de l'arbre de multiplexeur est active tandis que l'autre est coupée selon la valeur du dernier bit de sélection. Cette situation n'épargne pas toute la puissance statique dépensée par les parties inactives mais ne demande pas de logique supplémentaire pour générer les signaux de commande des *switches* de *power gating*. La répartition de la Figure 4.8 (b) permet d'épargner plus de puissance statique car tous les multiplexeurs non-actifs y sont coupés. Il faut cependant ajouter de la logique de contrôle pour générer tous les signaux *SLEEP*. Sur un *power gating* à fin grain, l'ajout de logique de contrôle est un handicap important car celle-ci réduira le bénéfice que le dispositif apporte.

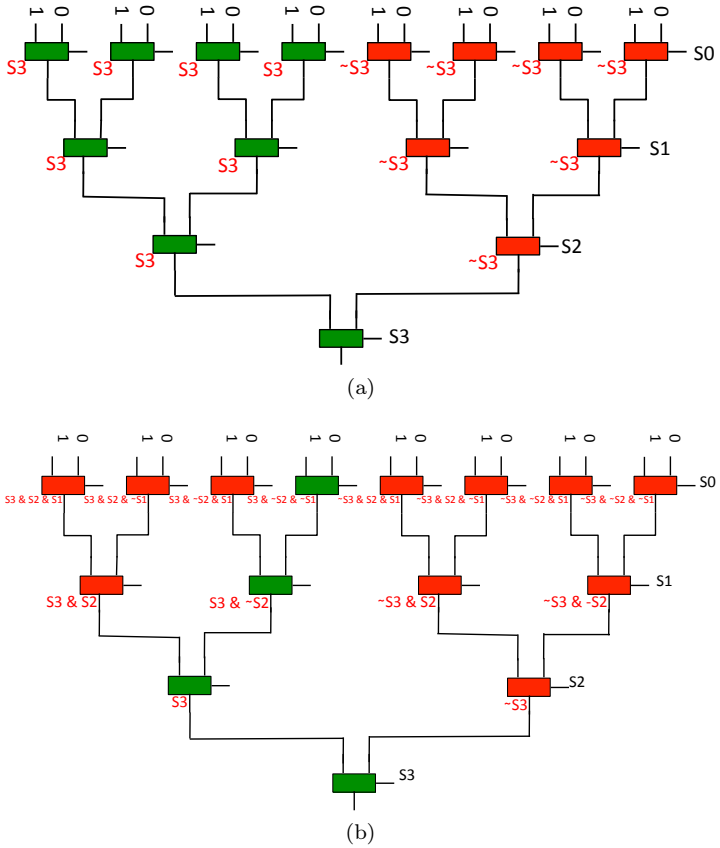


FIGURE 4.8: Illustration de deux répartitions des *switch* de *power gating*. Chaque multiplexeur est associé à un signal *SLEEP* contrôlant son alimentation (en bas à gauche de chaque bloc).

On peut remarquer que si les multiplexeurs 2 : 1 sont réalisés en logique de transmission, il faut insérer un inverseur devant chaque entrée et placer le *switch* sur son alimentation.

Comme indiqué dans la Section 2.2.5, le dimensionnement d'un dispositif de *power gating* consiste à optimiser les dimensions et les caractéristiques du transistor de *switch* par rapport à trois facteurs de mérite :

- La réduction du courant de fuite dans le bloc lorsqu'il est coupé : K_{leak}
- L'augmentation des délais à travers le bloc lorsqu'il est actif : K_{delay}
- La réduction des marges de bruit dans le bloc : K_{NM}

4.4.2 *Power gating* au niveau des BLE

Dans ce travail, un *power gating* intermédiaire portant sur les BLEs a été choisi pour évaluer la possibilité d'intégrer une telle structure sur un FPGA dont l'alimentation est proche de la tension de seuil.

La première étape de la conception du *switch* de *power gating* concerne la sélection du type de transistor. La Figure 4.9 reprend les différentes courbes caractéristiques $I_D - V_G$. Le transistor de *power gating* doit idéalement posséder le plus petit I_{OFF} possible pour avoir un bon K_{leak} et le plus grand I_{ON} possible pour avoir un K_{delay} proche de 1.

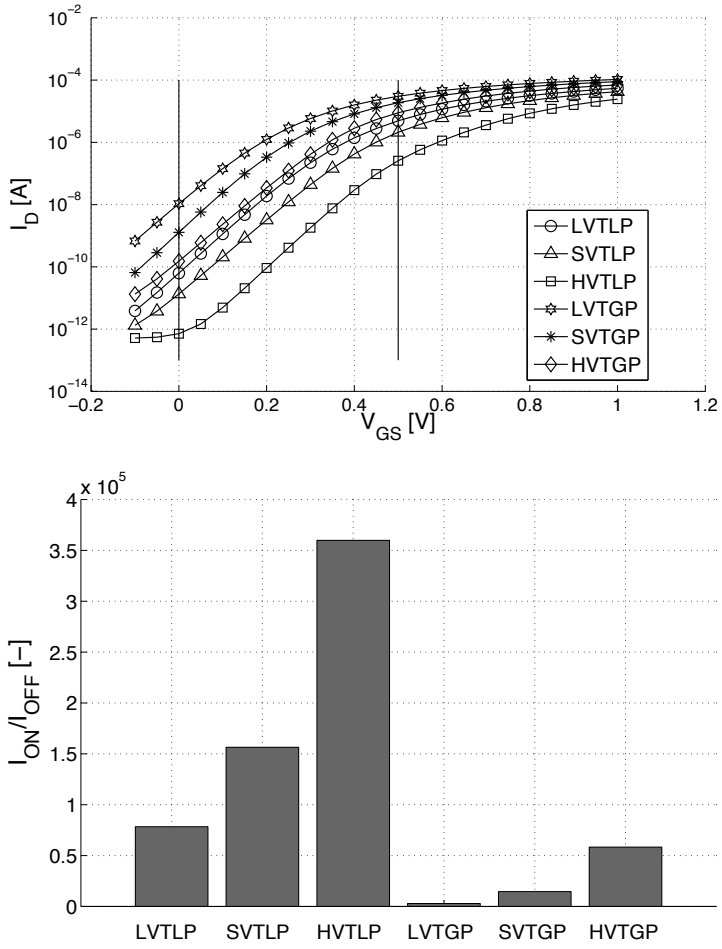


FIGURE 4.9: En haut : caractéristique $I_D - V_G$ des différents types de NMOS pour des longueurs de grilles de $80nm$. En bas : rapport I_{ON}/I_{OFF} pour les différents types de MOS.

La Figure 4.9 présente également les rapports I_{ON}/I_{OFF} pour tous les types de transistors. C'est le transistor $HVTLP$ qui présente le meilleur "effet interrupteur" ; il permettra d'obtenir le meilleur K_{leak} mais aura un impact important sur les délais du circuit car il introduira une résistance importante entre l'alimentation extérieure et l'alimentation locale qui sera fortement dégradée. Un transistor $LVTGP$ possède les caractéristiques inverses. Les transistors $SVTLP$ et $HVTLP$ n'ont pas été retenus car ils dégradaient trop le délai du BLE. Pour choisir un compromis, c'est un transistor $LVTLP$ qui a été choisi. Il possède un faible courant de fuite (inférieur à $0.1pA$) et son courant I_{ON} ne cède qu'un seul ordre de grandeur par rapport au $LVTGP$ (voire moins si la tension de grille est augmentée de $0.5V$ à $1V$).

Après avoir fixé les caractéristiques du transistor de *switch*, il reste à déterminer sa largeur. Il est

possible d'effectuer un ajustement de la résistance que le *switch* laisse entre l'alimentation réelle et l'alimentation virtuelle distribuée dans le circuit ciblé. Plus le transistor est grand, plus la résistance série est faible et plus le K_{delay} sera proche de 1, la valeur idéale. En revanche, plus le transistor est petit, plus le courant de fuite qu'il laisse passer lorsqu'il est bloqué, sera petit et le K_{leak} sera meilleur. Pour optimiser le dimensionnement du MOS, un algorithme a été développé.

Step 1 `/* Evaluation of maximum delay penalty acceptable */`

foreach V_{DD} **do**

Measure $t_{delay,\ w/o\ PG}$

Compute $K^{-1}_{delay,\ MAX} = \frac{t_{delay\ MAX}}{t_{delay\ w/o\ PG}}$

end

Step 2 `/* Optimization through` K_{leak} `*/`

foreach V_{DD} **do**

Find W**, width of power gating MOS s.t. :**

- Error $|K^{-1}_{delay} - K^{-1}_{delay,\ MAX}|$ is minimum
- $K^{-1}_{delay} > 0.8$
- Failure rate $< 10^{-5}$ & $K_{NM} > 0.8$

end

Algorithm 1: Power gating sizing algorithm

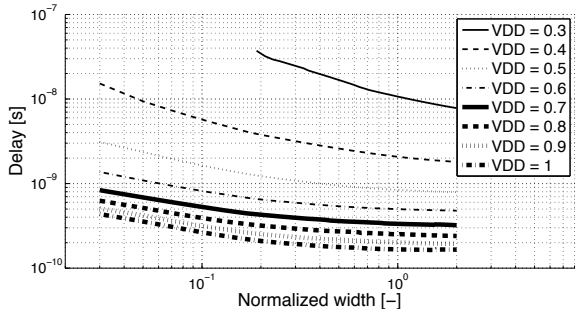
Le principe retenu pour le dimensionnement est de diminuer au maximum la taille du transistor de *power gating* pour limiter le courant de fuite qui le traverse lorsqu'il est coupé. Si le transistor est trop petit, l'augmentation du délai et la réduction des marges de bruit rendent le *power gating* inacceptable. Comme indiqué dans l'algorithme 1, trois conditions ont été imposées. Tout d'abord, le délai maximum acceptable dans le BLE pour respecter la contrainte des $50MHz$ est calculé. Sur base de celui-ci ainsi que du délai sans *power gating*, on calcule le $K^{-1}_{delay,\ MAX}$ acceptable pour respecter la contrainte de fréquence. C'est cette limite qui déterminera la longueur minimale du *switch*. L'augmentation de $K^{-1}_{delay,\ MAX}$ est cependant arbitrairement limitée à 20% par la seconde contrainte de l'algorithme pour ne pas accepter un impact trop important du dispositif sur les performances. Pour finir, la troisième condition imposée est que le dispositif ne réduise pas de plus de 20% les marges de bruit du circuit et que le taux de niveaux logiques mal interprétés ne dépasse pas 10^{-5}. L'optimisation du K_{leak} est réalisée pour chaque V_{DD}.

Les marges de bruits sont évaluées pour la technologie indépendamment de la structure étudiée par la méthode décrite dans la Section 2.2.4.

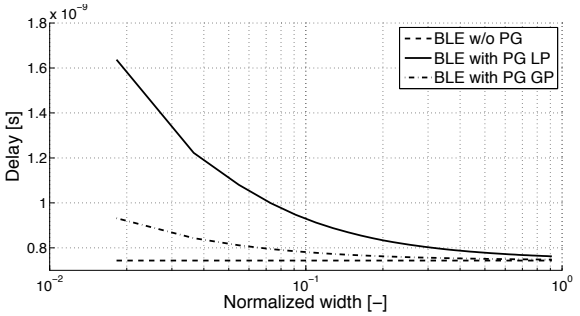
La Figure 4.10 (a) illustre l'évolution du délai du BLE en fonction de la largeur du transistor de *switch* pour différentes tensions d'alimentation. Cette largeur est normalisée en fonction de la somme des largeurs des NMOS (resp. PMOS) présents dans le BLE si le *switch* de *power gating* est un NMOS (resp. PMOS) comme effectué dans [38]. La Figure 4.10 (b) compare l'impact du *power gating* sur le délai pour des *switches* $LVTLP$ et $LVTGP$. On constate que le *switch* $LVTGP$ ralentit moins le BLE car la résistance parasite qu'il introduit est plus faible que celle d'un $LVTLP$.

La Figure 4.11 reprend les résultats de l'optimisation. On constate que l'écart entre le $K^{-1}_{delay,\ MAX}$ et le K^{-1}_{delay} est minimum lorsque la largeur normalisée est proche de 0.05 pour un *power switch* $LVTLP$. Cependant, les 20% de réduction du K^{-1}_{delay} sont atteints avant. Si le *switch* est un PMOS $LVTLP$, la réduction du K^{-1}_{delay} avec la réduction de la largeur est plus lente et les 20% de réduction du K^{-1}_{delay} ne sont atteints que pour une largeur normalisée de 0.01.

L'explication de ces observations se situe dans le principe de fonctionnement du *power gating*. Le transistor de *switch* doit fournir le courant total consommé par le circuit à tout moment. Lorsque

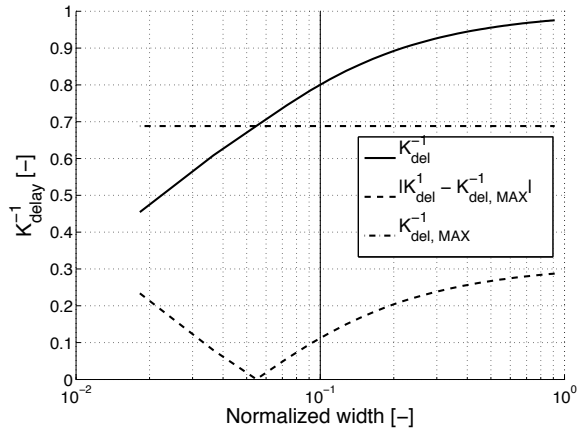


(a) Evolution du délai à travers le BLE équipé d'un *power gating* selon la largeur normalisée du *switch* $LVTLP$ pour différentes tensions d'alimentation.

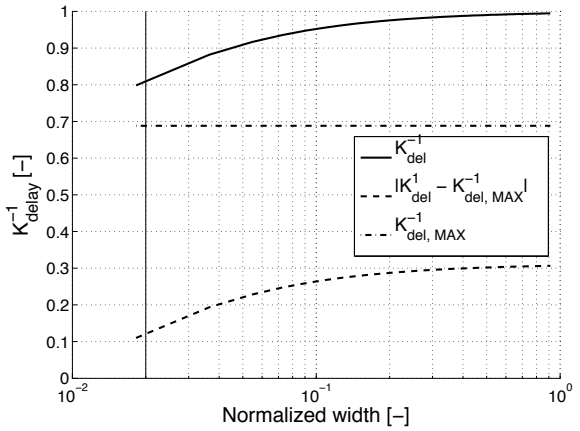


(b) Comparaison de l'évolution du délai à travers le BLE en fonction de la largeur normalisée du *switch* pour des *switches* $LVTLP$ et $LVTGP$ ($V_{DD} = 0.5V$).

FIGURE 4.10: Impact du *switch* sur le délai dans le BLE.



(a) *Power switch* : $LVTLP$, $V_{DD} = 0.5V$.



(b) *Power switch* : $LVTGP$, $V_{DD} = 0.5V$.

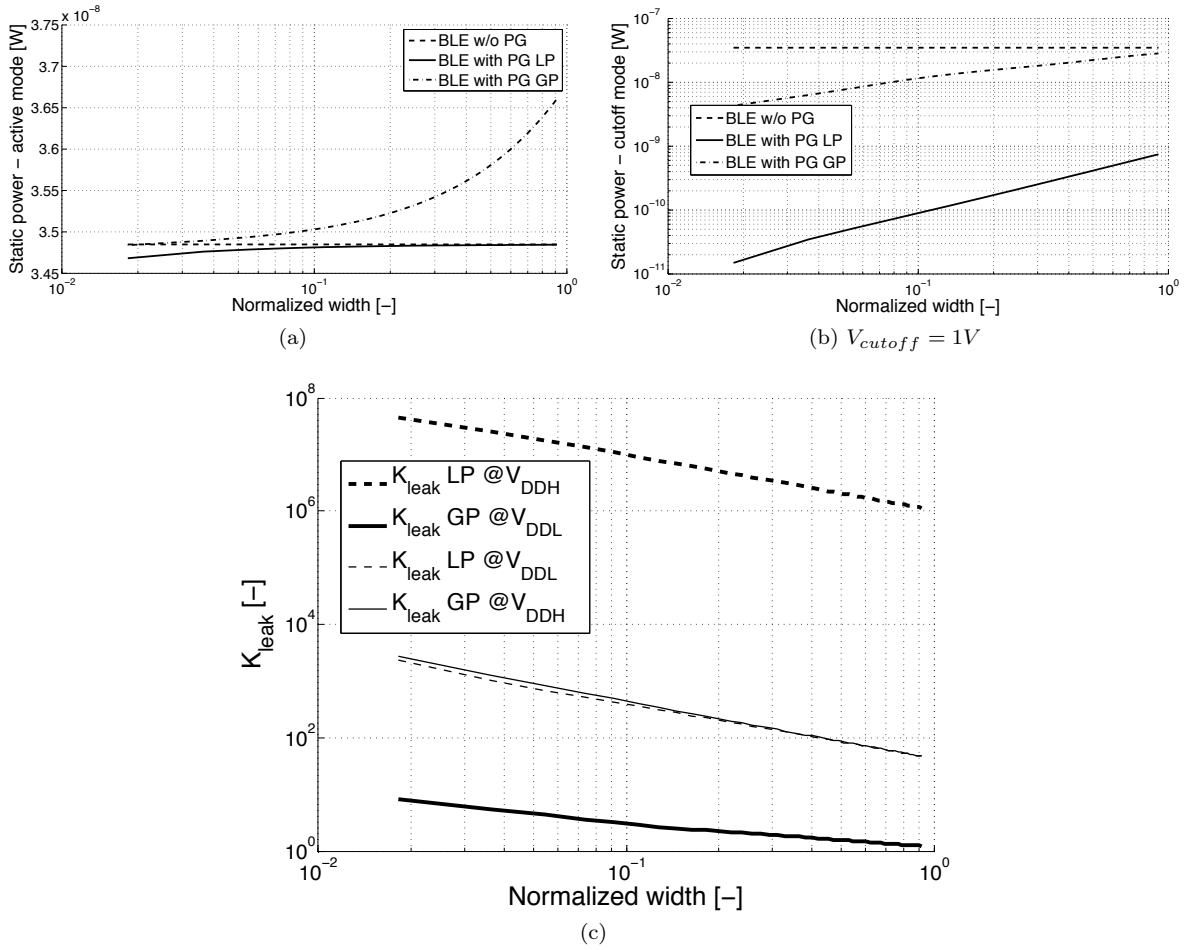
FIGURE 4.11: Evolution du K_{delay}^{-1} selon la largeur normalisée du *switch*. Les contraintes de délai de l'algorithme 1 et le K_{leak} sont représentés pour $V_{DD} = 0.5V$. La ligne verticale représente la contrainte arbitraire annonçant 20% de réduction du K_{delay}^{-1}.

la consommation est grande, la chute de tension à ses bornes augmente et la tension d'alimentation virtuelle que voit le circuit diminue. Si le circuit est suffisamment grand, les charges/décharges des noeuds n'auront pas lieu au même moment et il y aura un effet de moyenne sur le courant consommé. C'est cet effet qui permet l'utilisation d'un *switch* plus petit que la somme de tous les NMOS (ou PMOS) présents dans le circuit car ils ne seront pas tous conducteurs au même moment. En revanche, si le circuit est trop petit, une transition à l'entrée impose une transition d'une majorité des noeuds peu de temps après. Sans l'effet de moyenne, la largeur du *switch* doit être du même ordre que la somme de tous les NMOS (ou PMOS) présents car une majorité de ceux-ci sont conducteurs en même temps. Cet effet pourrait empêcher la réalisation d'un *power gating* plus fin que le niveau des BLEs. Une étude détaillée devra être réalisée pour répondre à cette question.

Les figures 4.12 (a) et (b) présentent l'évolution de la puissance statique dissipée en mode actif (*power switch* fermé) et en en mode coupé (*power switch* ouvert). On constate que l'ajout d'un transistor $LVTLP$ entre les deux tensions d'alimentation ne modifie que peu la puissance statique consommée lorsque le circuit est actif. Etant donné que son courant I_{ON} est limité, lorsqu'il est trop petit on constate une petite réduction de la consommation statique du circuit. Pour le *switch* $LVTGP$, c'est l'inverse. Il possède un courant I_{ON} élevé qui provoque une augmentation de la consommation statique

en mode actif.

Lorsque le circuit est coupé, le *switch* $LVTLP$ réduit fortement la consommation statique grâce à son faible I_{OFF} tandis que le *switch* $LVTGP$ est moins performant (Fig. 4.12).



(a) (b) $V_{cutoff} = 1V$ (c)

FIGURE 4.12: Impact du *switch* de *power gating* sur la puissance statique en mode actif ou coupé ainsi que sur le K_{leak}. $V_{DDL} = 0.5V$, $V_{DDH} = 1V$.

Pour réduire davantage le courant I_{OFF}, on peut augmenter la tension de grille des transistors de *switch* lorsqu'ils sont bloqués. La Figure 4.12 (c) montre l'évolution du K_{leak} pour les deux *switch* lorsque l'on passe la tension de grille de $V_{DDL} = 0.5V$ à $V_{DDH} = 1V$.

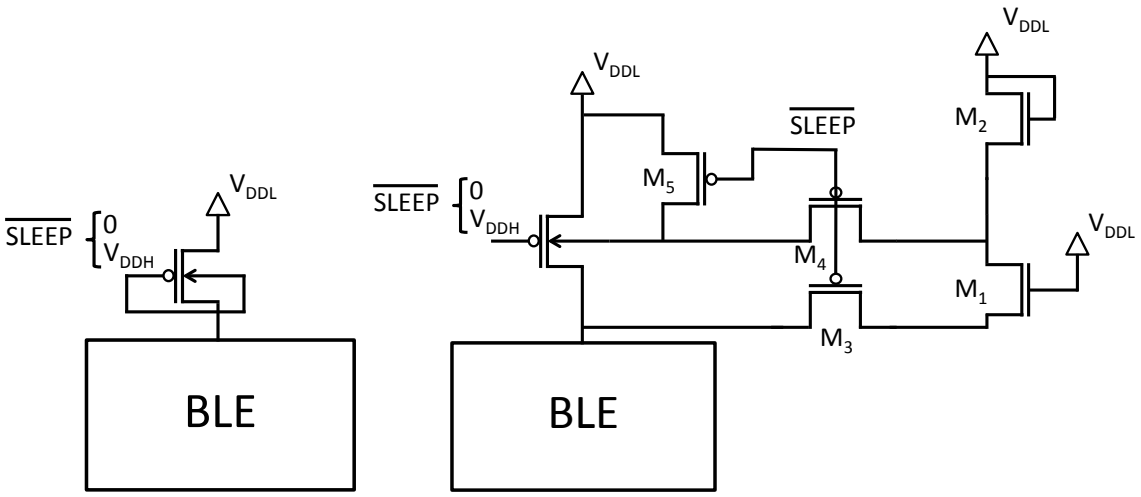
Le *power gating* standard d'un BLE avec un transistor $LVTGP$ dont la tension de grille en mode inactif est supérieure à la tension d'alimentation permet donc de réduire la consommation statique d'un facteur allant de 2700 à 50 selon la pénalité en délai que l'on s'accorde.

4.4.3 Amélioration des performances du *power gating*

Pour améliorer les performances de ce dispositif et réduire son impact sur la vitesse, deux solutions on été envisagées. Le but est d'augmenter le courant I_{ON} lorsque le circuit est en mode actif sans augmenter le courant I_{OFF} en mode coupé. Pour cela, un dispositif a été développé sur le principe du *Forward Body Biaising* (FBB). Le FBB consiste à utiliser l'effet de *body* pour faire varier le V_T du transistor de *switch*.

La réduction de la tension du *bulk* qui est généralement fixée à V_{DD} permet de réduire le V_T du

transistor et augmenter le courant I_{ON}. Il suffit donc de connecter la tension de *bulk* du *switch* au signal d'activation du circuit pour réaliser du FBB en mode actif comme illustré dans la Figure 4.13 (a).



(a) *Forward Body Biasing.* (b) *Forward Body Biasing* adaptatif.

FIGURE 4.13: Schémas de l'utilisation du FBB pour améliorer les caractéristiques du *power switch*.

La Figure 4.14 montre que l'utilisation du contact de *body* comme une seconde grille pour augmenter le courant I_{ON} présente de bons résultats. Pour une largeur normalisée de 0.01, on obtient une réduction de 25% du délai par rapport à un BLE dont le *power gating* n'utilise pas le FBB.

Les caractéristiques du *power gating* en mode coupé ne changent pas par rapport aux résultats présentés dans la Figure 4.12 car le FBB est désactivé lorsque le signal $SLEEP$ est inversé. On parvient donc bien à améliorer I_{ON} sans handicaper I_{OFF}.

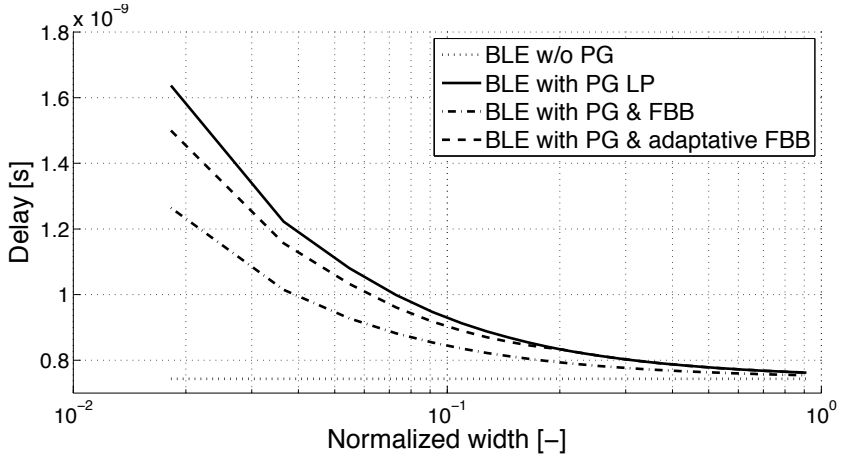


FIGURE 4.14: Evolution du délai dans le BLE suivant la largeur normalisée du *power switch* avec ou sans le *forward body biaising*.

Le FBB présente néanmoins un désavantage. La polarisation du *bulk* ne peut pas être trop élevée sous peine de voir augmenter le courant de fuite passant à travers la diode coupée Source-*Bulk*. La tension de seuil de la diode est proche de $0.7V$; en polarisant le *bulk* à GND, il y a un risque d'observer un courant de fuite important dans la jonction si le *core voltage* du circuit vaut $V_{DDL} = 0.5V$.

La Figure 4.15 (a) présente l'évolution de la consommation statique en mode actif. On constate

que la consommation statique du circuit est supérieure de 10% lorsque le FBB est utilisé à cause du courant dans la jonction. Celui-ci augmente linéairement avec la largeur du *power switch* car la surface de jonction bloquée augmente linéairement.

Pour compenser cette augmentation de la puissance statique consommée, un *power gating* à FBB adaptatif a été développé. Le but est d'activer le FBB lorsque la demande en courant est importante, donc lorsque le transistor de *switch* dont le *bulk* est polarisé à V_{DDL} impose une trop grande chute de tension sur l'alimentation du circuit. Le circuit illustré dans la Figure 4.13 (b) permet de contrôler la tension de *bulk* du *switch* en mesurant la chute de potentiel sur le *power switch*.

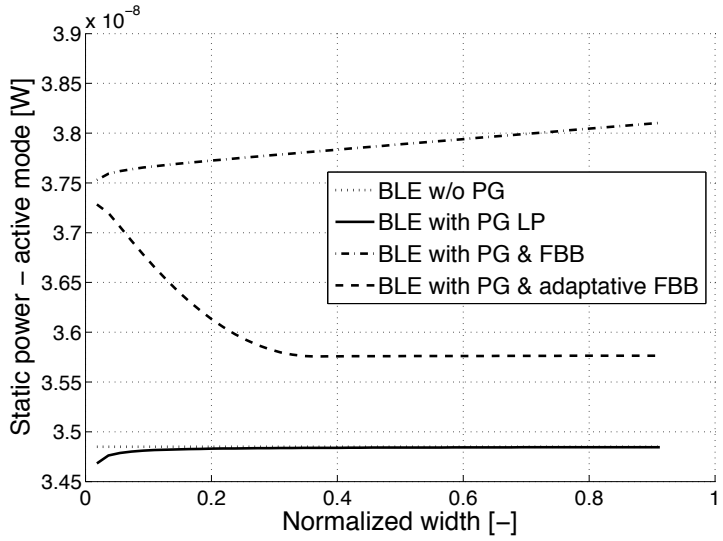
Les transistors M_1 et M_2 forment un amplificateur mesurant la différence de potentiel entre la tension d'alimentation réelle et la tension d'alimentation vue par le circuit. Si cette différence augmente, ce qui arrive si le circuit consomme trop de courant par rapport à la résistance parasite du transistor de *power switch*, le courant de drain de M_1 augmente. La tension à la source de M_2, monté en diode pour servir de charge à M_1, va diminuer. La tension de *bulk* du *switch* diminue donc lorsque le courant demandé par le circuit augmente. Le FBB ne sera donc activé que si le circuit est en activité et sera désactivé lorsqu'il n'est pas utilisé.

Les transistors $M_3 - M_5$ permettent de couper le dispositif lorsque le signal de *power gating* est haut, ce qui permet de maintenir de bonnes caractéristiques en mode coupé.

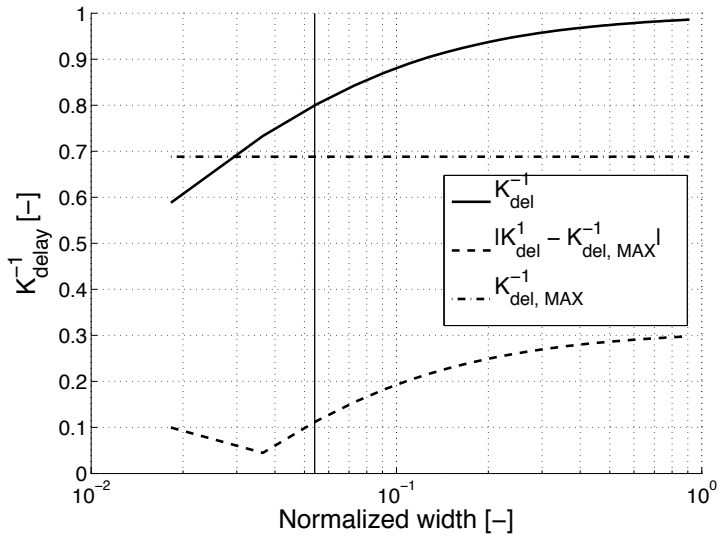
La Figure 4.14 montre que le délai du BLE est diminué par rapport à la situation où le FBB n'est pas utilisé mais l'est moins que lorsque le *bulk* est connecté à GND. En effet, le *bulk* connecté au circuit de FBB adaptatif ne descend pas jusqu'à $0V$. La performance du FBB adaptatif peut être contrôlée par le gain de l'amplificateur de contrôle mais celui-ci doit avoir une faible consommation pour ne pas contrebalancer la réduction du courant de fuite de jonction. La Figure 4.15 (a) montre que lorsque le transistor de *switch* est petit, donc lorsque sa résistance est grande, le FBB s'active pour augmenter le courant I_{ON} ; ce qui cause une augmentation du courant de fuite dans la jonction. En revanche, plus le *switch* est large, moins l'utilisation du FBB est nécessaire, d'où une réduction du courant à travers la jonction bloquée car la tension de *bulk* est de moins en moins réduite. Cette réduction se stabilise lorsque le FBB n'est plus nécessaire et l'écart qui sépare la consommation statique du BLE équipé avec le dispositif adaptatif par rapport au BLE équipé du *power gating* standard, est causé par la consommation statique de l'amplificateur de contrôle.

La Figure 4.15 (b) montre que le BLE utilisant un FBB adaptatif, peut être équipé d'un *switch* plus petit (0.04 au lieu de 0.1 cf. Fig. 4.11 (a)). Selon la Figure 4.12 (c), cela permet d'augmenter K_{leak} de 9.1×10^6 à 2×10^7 soit plus de 200% d'augmentation. Ou bien, pour un même K_{leak}, on peut augmenter la taille du *switch* pour réduire la pénalité en vitesse.

En conclusion, Le FBB adaptatif semble permettre un intermédiaire intéressant entre l'utilisation du FBB et le *power gating* standard tout en réduisant la consommation statique lorsque le circuit est activé. La consommation statique du circuit de contrôle reste cependant non négligeable et le dispositif gagnerait donc de l'intérêt pour un *power gating* à un plus haut niveau. En effet, la consommation du circuit de contrôle serait ainsi contrebalancée par une plus grande réduction du courant de jonction. L'utilisation du FBB adaptatif à fin grain nécessite une analyse détaillée pour évaluer la réduction de délai ou l'augmentation de K_{leak} qu'il peut apporter.



(a)



(b) FBB adaptatif, $LPLVT$

FIGURE 4.15: En haut : evolution de la puissance statique pour les différents dispositifs de *power gating*. En bas : résultat du dimensionnement du *switch* de *power gating* équipé du dispositif de *forward body biaising* adaptatif. Ce résultat doit être comparé au résulat de dimensionnement de la Figure 4.11 (a) ou (b).

4.5 Répartition des types de transistors et des V_T

Afin d'optimiser finement la répartition des délais et des puissances dissipées dans la structure, l'impact du type de transistor a été étudié. La structure a été divisée en deux parties : la partie logique à l'intérieur du CLB (cf. Section 3.2) et la partie routage à l'extérieur du CLB détaillée dans la Section 2.1.2.

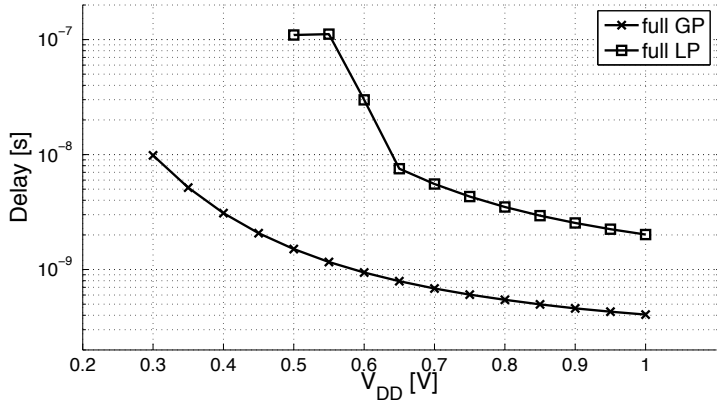
4.5.1 Optimisation de la structure de routage

Plusieurs facteurs entrent en jeu dans cette optimisation. L'augmentation du V_T des transistors permet de réduire le courant de fuite mais ralentit la structure. Il faut donc augmenter le V_{DD} pour

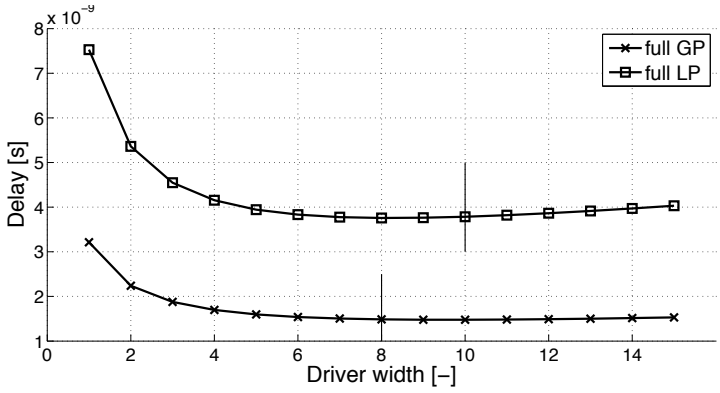
compenser et conserver des performances acceptables. La Figure 4.16 (a) illustre l'augmentation de délai subie par un segment de longueur 3 lorsque ses transistors $SVTGP$ sont remplacés par des transistors $SVTLP$. On remarque que pour atteindre un délai de $1ns$, il faut augmenter la tension d'alimentation de $0.5V$ à $1V$ et quitter le domaine du sous-seuil.

En parallèle du choix des transistors pour l'implémentation, il faut dimensionner le *driver* pour minimiser le délai sur le segment comme indiqué dans [22]. La Figure 4.16 (b) reprend l'évolution du délai dans un segment de longueur 3 pour deux implémentations en fonction de la taille du *driver*. On constate que l'optimum se déplace selon l'implémentation. Il faudra donc adapter le *driver* pour rester à l'optimum lors des comparaisons d'implémentation afin d'éviter de fausser les comparaisons consommation/délais.

La Figure 4.17 montre la répartition des délais dans le segment. Les *Sense Buffers* sont maintenus à la largeur minimale car ils représentent un faible pourcentage du délai. De plus, cela permet de limiter la charge sur le segment. L'optimisation se concentrera sur les multiplexeurs et les *drivers* qui représentent la majorité du délai.



(a)



(b)

FIGURE 4.16: Evolution du délai de propagation le long d'un segment parcourant 3 CLBs en fonction de la tension d'alimentation et de la largeur du *driver*. La largeur du *driver* est exprimée en multiple de la largeur minimale de la technologie ($0.135\mu m$ pour la technologie $65nm$ distribuée par STMicroelectronics). Ce délai a été simulé pour des segments consitués de MOS $SVTLP$ et $SVTGP$ pour comparaison.

L'optimisation consiste à déterminer la meilleure répartition du V_T et du type de transistor pour les trois éléments : *sense buffer*, multiplexeur et *driver*. Cette optimisation doit être réalisée de manière exhaustive car l'utilisation de logique de transmission implique que les caractéristiques des blocs sont liées les unes aux autres.

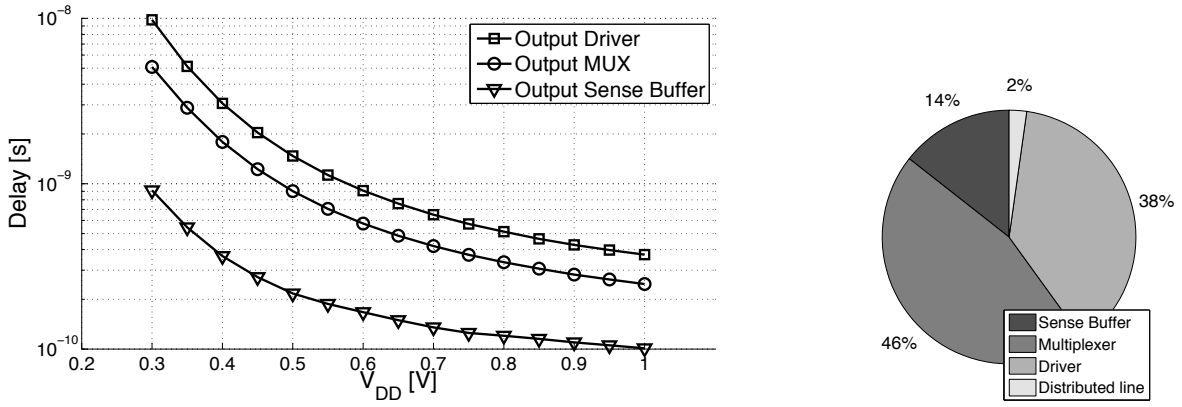


FIGURE 4.17: Evolution du délai le long du segment en fonction de la tension d'alimentation. La proportion des délais dans chaque bloc pour $V_{DD} = 0.5V$ est indiquée dans le diagramme de droite.

Toutes les répartitions sont comparées sur base de 3 facteurs de mérite : la consommation statique, la consommation dynamique et le délai entre l'entrée du *sense buffer* et l'entrée du *sense buffer* situé 3 CLBs plus tard.

TABLE 4.2: Performances relatives en vitesse et en consommation pour différentes répartitions du V_T. Toutes les répartitions sont comparées avec la situation constituée uniquement de transistors SV_T.

V_T for SB/Mux/Driver	Delay @0.5V	Leakage power @0.5V	Total power @iso-delay	Leakage power @iso-delay	V_{DD} [V] for iso-delay
LV_T, SV_T, LV_T	-27%	+47.5%	-17.4%	+22.8%	0.451
SV_T, SV_T, LV_T	-15%	+23.9%	-9.1%	+12.7%	0.475
LV_T, SV_T, SV_T	-13%	+23.7%	-7.9%	+14.9%	0.479
SV_T, SV_T, SV_T	-	-	-	-	0.5
LV_T, HV_T, LV_T	+17%	-45%	+7.9%	-33.5%	0.525

La Table 4.2 reprend les 5 répartitions les plus performantes. On constate que les répartitions les plus performantes sont celles qui utilisent des transistors SV_T ou HV_T pour les multiplexeurs en logique de transmission. Ceci permet de garder une R_{OFF} élevée qui empêche le chargement et déchargement du grand nombre de noeuds inutilisés dans les multiplexeurs. Pour les *sense buffers* et les *driver*, les transistors HV_T sont évités pour conserver de bonnes performances en vitesse.

La Figure 4.18 présente la puissance dynamique et statique des 5 répartitions. La puissance statique semble négligeable et la puissance dynamique est environ constante pour toutes les répartitions mais deux considérations doivent être gardées à l'esprit :

- Il y a entre 20 et 40 segments par canal de routage selon la valeur de W.
- La consommation statique des systèmes autonomes en énergie (EAS) doit être très réduite car leurs facteurs d'activité sont généralement réduits et c'est la durée de vie sur une batterie limitée qui est ciblée.

En tenant compte de ces remarques, la puissance statique consommée par tuile est 20 à 40 fois supérieure à la puissance statique exposée dans la Table 4.2 et la Figure 4.18. Pour réduire cette

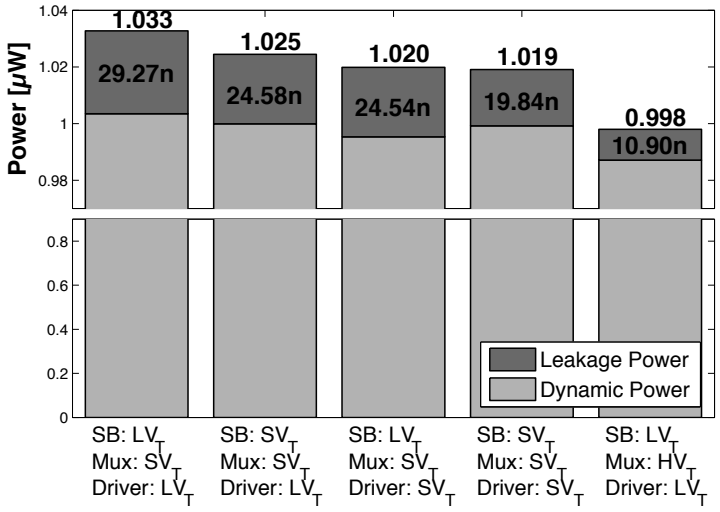


FIGURE 4.18: Consommation dynamique et statique des 5 meilleures répartitions des V_T.

puissance statique, la dernière répartition a été choisie. Elle est constituée des *sense buffers* en LV_T, des multiplexeurs en HV_T et des *drivers* en LV_T. Cette répartition présente un petit handicap en délai et permet une réduction de 35% de la puissance statique (après correction pour arriver aux mêmes performances de vitesse).

4.5.2 Optimisation du CLB

Le CLB a été étudié de la même façon que l'a été la structure de routage (cf. Section 4.5.1). La grappe a été divisée en 2 parties : le routage local et la logique (BLE).

Pour les deux parties, toutes les configurations de type de transistor et de V_T ont été simulées. La Table 4.3 regroupe les 6 répartitions les plus performantes. Là aussi, la recherche exhaustive est nécessaire car l'implémentation du routage local en logique de transmission rend le délai de ce routage dépendant du courant I_{ON} produit par les transistors du BLE.

L'évolution des performances en vitesse du CLB est estimée statistiquement à travers des simulations ELDO du CLB séparé de la structure de routage. Des circuits du *benchmark* MCNC ([18]) suffisamment petits pour tenir dans 4 LUTs sont synthétisés sur le CLB. Des vecteurs d'entrée et des transitions à $50MHz$ sont appliqués au circuit et le délai sur le chemin critique est mesuré. Ce chemin critique comporte jusqu'à 2 BLEs et 3 multiplexeurs de routage local.

Les répartitions E et F nécessitent une tension d'alimentation légèrement supérieure à $0.5V$ à cause de l'utilisation de transistors LP qui ont un V_T élevé et donc un I_{ON} trop faible à $0.5V$. Les performances des CLBs implémentés avec des transistors LP sont trop faibles pour atteindre le MHz à l'exception des répartitions E et F avec un V_{DD} augmenté.

La répartition C présente une bonne réduction de la consommation statique mais la pénalité en délai est également importante.

A l'inverse, la répartition D subit une augmentation réduite du délai mais la réduction de la consommation statique est limitée à 11%.

La solution A a été choisie comme compromis entre une bonne réduction de la consommation statique (30%) et une pénalité limitée de délai (60%). Cette répartition est composée de transistors à faible V_T qui sont rapides et qui sont utilisés pour le routage local et des transistors lents à haut V_T pour les BLEs. De cette façon, tous les chemins entre V_{DD} et GND passent par au moins un transistor à haut V_T imposant un faible courant de fuite.

La Figure 4.19 détaille la répartition des transistors $LVTGP$ et $HVTGP$ dans les différents blocs. On constate qu'après optimisation, il y a un peu plus de transistors $HVTGP$ à faible courant de fuite

TABLE 4.3: Puissance consommée relative et délai relatif pour différentes répartitions de V_T par rapport à la situation $SVTGP$ pour un CLB @$0.5V^*$ ou @$0.55V^\dagger$

	Routing	BLE	Total power	Leakage power	Delay
A^*	GPLV_T	GPHV_T	-15.4%	-28.9%	+60%
B^*	GPSV_T	GPSV_T	-	-	-
C^*	GPSV_T	GPHV_T	-26.3%	-71.7%	+85.9%
D^*	GPHV_T	GPSV_T	+7.9%	-10.9%	+39.5%
$E^\dagger$	LPLV_T	LPSV_T	+63.4%	-34.4%	+42.6%
$F^\dagger$	LPLV_T	LPHV_T	-9.4%	-83.6%	+99.6%

que de transistors rapides $LVTGP$. Si les multiplexeurs avaient été réalisés en logique CMOS à la place de la logique de transmission, la proportion de $LVTGP$ aurait été plus faible. L'architecture retenue permet en effet d'insérer de longues chaines de transistors rapides en logique de passage entre deux transistors lents en logique CMOS sans obtenir un grand courant de fuite.

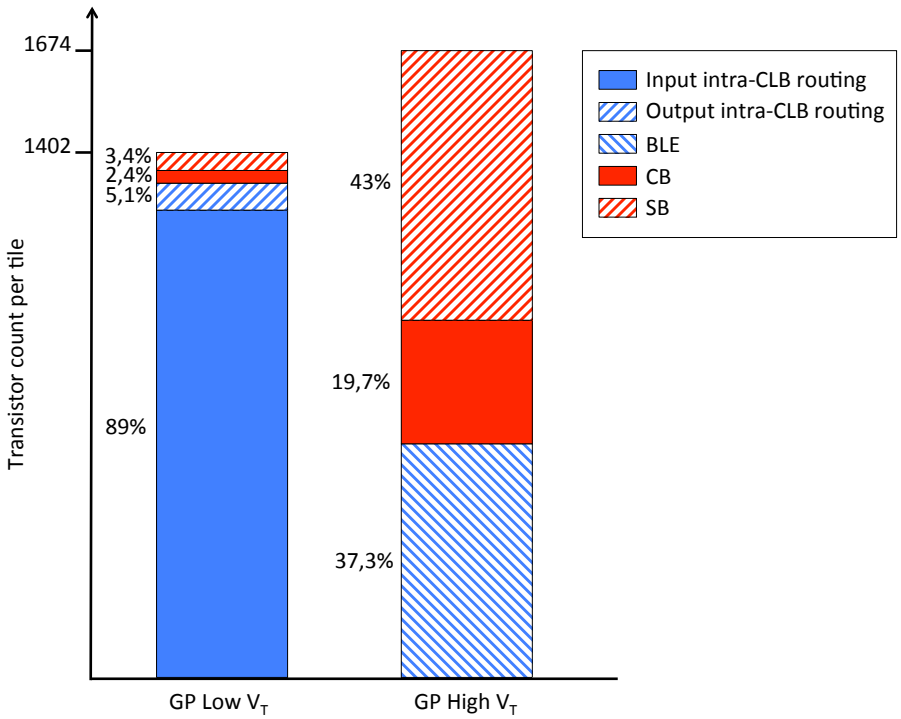


FIGURE 4.19: Histogramme de répartition des différents types de MOS dans chaque bloc après optimisation.

4.6 Performances

Sur base des répartitions développées dans la Section 4.5, une tuile a été composée en assemblant le CLB avec deux canaux de routage (un CB vertical et un CB horizontal) et en les connectant à travers un *switch box*. La structure conçue dans les Chapitres 3 et 4 a été simulée.

La puissance consommée a été simulée à bas niveau par le simulateur ELDO. Tout comme pour l'évolution des performances du CLB ; la tuile a été stimulée à $50MHz$ après avoir synthétisé un petit circuit du *benchmark*. Les entrées du circuit ont été routées dans les CBs et le SB entourant le CLB. Les segments inutilisés ont été stimulés à $50MHz$. De cette façon, on peut obtenir une borne supérieure de la consommation dynamique d'une tuile qui serait insérée dans un FPGA complet.

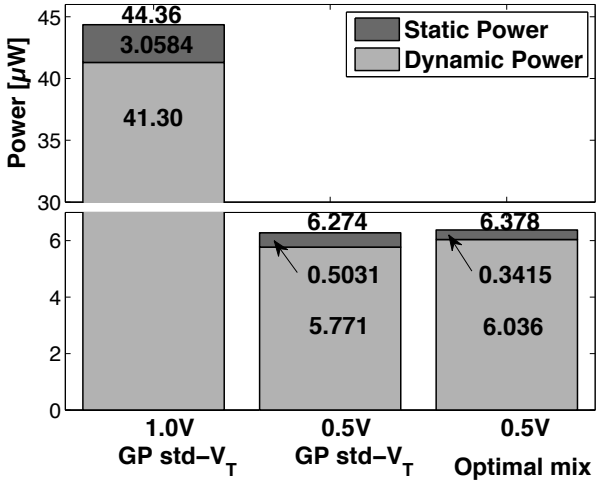


FIGURE 4.20: Histogramme des puissances consommées par une tuile pour la répartition optimale @$0.5V$ ainsi que pour la tuile $SVTGP$ alimentée @$1V$ ou à @$0.5V$.

La Figure 4.20 compare les consommations de puissance statique et dynamique pour une tuile optimisée et pour une tuile $SVTGP$ (alimentée à $1V$ ou à $0.5V$). La réduction de la tension d'alimentation de $1V$ à $0.5V$ permet la réduction de plus de 86% de la consommation totale. La répartition optimale a été conçue pour réduire la consommation statique. Cela permet d'obtenir une faible puissance consommée lorsque le FPGA est inactif dans l'optique de synthèse de circuits à faible facteur d'activité. La répartition optimale permet de réduire de 33% la consommation statique au prix d'une augmentation de 2% de la puissance totale. Cette augmentation est due à l'augmentation des délais dans la structure qui étendent les durées des transitions.

Sur base de ces simulations, les paramètres à fournir au simulateur VPR pour compléter le fichier de description de l'architecture ont été extraits. Les paramètres architecturaux extraits pour V_{DD} allant de $0.5V$ à $1V$ sont :

- Délai à travers le *driver* du *switch block*.
- Délai combinatoire à travers le BLE.
- Délai séquentiel entre l'entrée du BLE jusqu'à l'entrée du DFF (en comptant le délai de *setup* du DFF).
- Délai séquentiel entre le flanc montant de l'horloge jusqu'à la sortie du BLE.
- Délai dans le CLB : de la sortie d'un BLE vers l'entrée d'un autre BLE.
- Délai dans le CLB : d'une entrée du CLB vers l'entrée d'un BLE.
- Délai dans le CLB : de la sortie d'un BLE vers une sortie du CLB.

Un multiplieur RCA sur 16 bits a été synthétisé sur la structure conçue grâce au *CAD Flow* décrit dans la Section 3.1. La synthèse a été réalisée sur des architectures dont le V_{DD} s'étend de $0.5V$ à $1V$. La Figure 4.21 montre l'évolution du délai sur le chemin critique. Ce délai est distribué équitablement entre le routage global et le délai intra-CLB.

La synthèse du multiplieur sur la structure nécessite 4×4 tuiles. La puissance consommée par le circuit a été évaluée en multipliant par 16 la puissance consommée par une seule tuile stimulée à la fréquence maximale permise par le délai du chemin critique. La puissance dynamique a été simulée en appliquant des transitions sur toutes les entrées. Il existe d'autres méthodes pour estimer de manière

réaliste la puissance dynamique. Il est par exemple possible de prolonger le modèle développé dans la Section 3.2.2.3. Pour cela, on stimule les entrées de la tuile selon les taux moyens d'utilisation donné par le *CAD Flow*. La prise en compte de davantage d'informations permet de limiter la surestimation de la puissance dynamique mais impose une complexification du modèle.

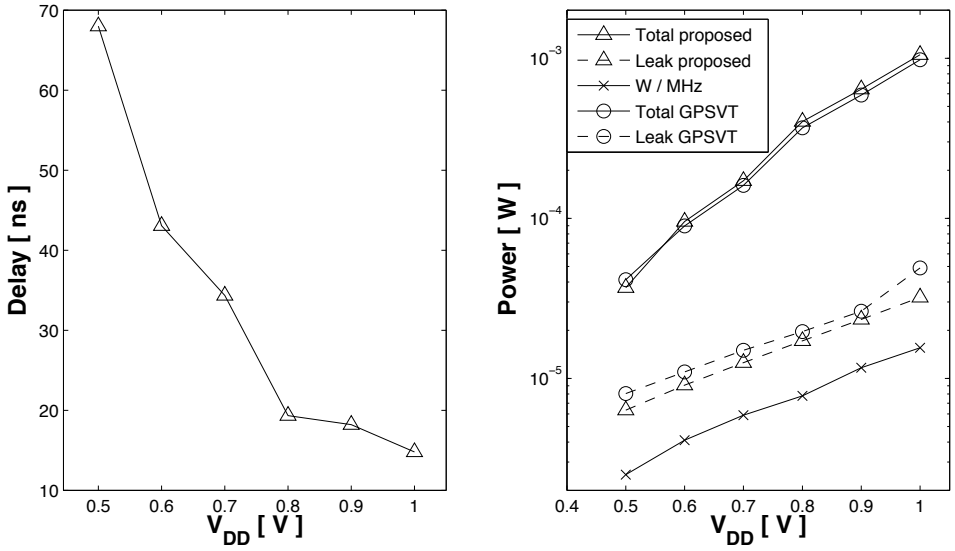


FIGURE 4.21: Simulation du délai sur le chemin critique et de la puissance consommée par un FPGA 4×4 sur lequel un multiplieur RCA 16 bit a été synthétisé. Le graphique de droite reprend également l'énergie par opération de ce multiplieur.

La Figure 4.21 indique que la puissance totale du FPGA *SVTGP* et celle du FPGA optimisé sont semblables. En revanche, la différence entre les consommations statiques des deux types de tuile est amplifiée par la taille du FPGA. Au total, le FPGA optimisé présente une réduction substantielle de la consommation statique.

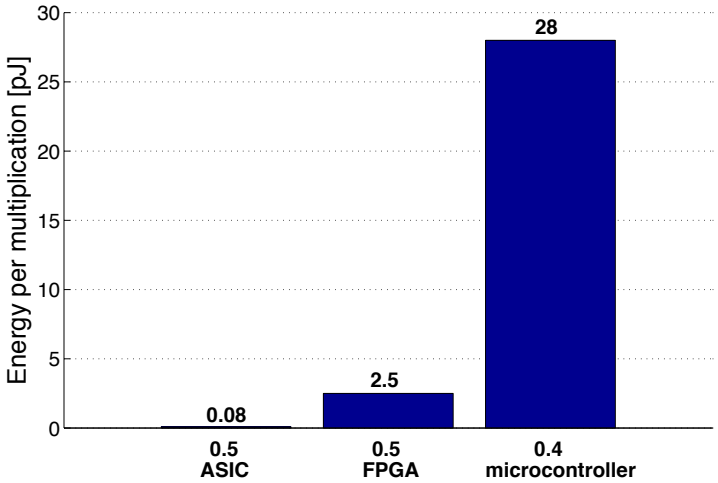


FIGURE 4.22: Comparaison des énergies nécessaires pour réaliser une multiplication sur 16 bits sur un ASIC ULV, sur le FPGA ULV et sur un microcontrôleur ULV.

La puissance par MHz correspond à l'énergie consommée par opération. Malgré les approximations réalisées pour estimer la puissance consommée, elle donne une borne supérieure intéressante pour comparer l'implémentation du multiplieur sur le FPGA à une implémentation ASIC ULV. L'énergie par

opération est sans doute le facteur de mérite le plus représentatif lorsque l'on cible des EAS (à l'exception des *RFID tags* pour lesquels il est plus intéressant d'étudier la puissance instantanée consommée). En effet, en connaissant l'énergie nécessaire par opération, on peut immédiatement convertir les caractéristiques de la batterie en une durée de vie du système. La Figure 4.22 reprend l'énergie nécessaire pour une multiplication 8 bits sur un ASIC ULV, sur le FPGA ULV de ce travail et sur un microcontrôleur ULV qui fait figure d'état de l'art dans le domaine.

L'ASIC utilisé pour comparaison provient de [36]. Les auteurs implémentent le même multiplieur en ASIC dans la même technologie et en utilisant les mêmes techniques d'optimisation de la puissance. Ils ont atteint une énergie par opération de $80fJ$ @$0.5V$ alors que le FPGA développé atteint $2.5pJ$ @$0.5V$. Le facteur 30 qui sépare ces deux valeurs mesure donc l'écart entre la logique reconfigurable et l'implémentation ASIC en $65nm$ et en ULV.

Les travaux de [57] décrivent un microcontrôleur ULP pouvant servir de plateforme pour implémenter une reconfiguration software. Il a été réalisé dans la même technologie $65nm$ et en utilisant l'approche multi-V_T et GP/LP pour réduire sa consommation. Il atteint $7pJ$ par opération et la multiplication demande 4 cycles, ce qui porte à $28pJ$ l'énergie nécessaire à la réalisation d'une multiplication sur 16 bits. La reconfiguration hardware développée dans ce travail est donc $11\times$ plus performante que la reconfiguration software dans la technologie $65nm$ et dans le domaine de l'ULV.

Chapitre 5

Validation, *layout* d'un circuit de test et simulations post-*layout*

Ce chapitre décrit les simulations physiques d'un FPGA complet. Pour éviter de configurer manuellement les tuiles, une couche software a été partiellement développée. Le circuit a été validé pour 16 *corners afin de vérifier la fonctionnalité et les variations de performances sont reprises dans ce chapitre. Le layout d'un CLB a été réalisé afin d'intégrer un test-chip de validation. Les layout des différents blocs sont détaillés ainsi que les simulations post-layout. Ces simulations ont permis de mesurer l'influence des parasites sur les facteurs de mérite de la structure.*

5.1 Adder 2b

Pour tester et caractériser la structure développée dans le Chapitre 4, un additionneur sur 2 bits dont le code Verilog est présenté ci-dessous, a été synthétisé sur la structure.

```
1
2  module bench_2bit_adder_with_carryout_and_overflow (carryin, X, Y, S,
       carryout, overflow);
3          parameter n = 2;
4
5          input carryin;
6          input [n-1:0] X, Y;
7          output [n-1:0] S;
8          output carryout, overflow;
9
10         reg [n-1:0] S;
11         reg carryout, overflow;
12
13         always @(X or Y or carryin)
14         begin
15                 S = X + Y + carryin;
16                 carryout = (X[n-1] & Y[n-1]) | (X[n-1] & ~S[n-1]) |
                       (Y[n-1] & ~S[n-1]);
17                 overflow = carryout ^ X[n-1] ^ Y[n-1] ^ S[n-1];
18         end
19
20 endmodule
```

Après passage dans le *CAD Flow*, les fichiers de placement et de routage ont été interprétés pour en extraire les valeurs de tous les bits de configuration. La Figure 5.1 illustre le schéma de placement et routage de l'additionneur sur le FPGA 2×2.

Pour utiliser le *framework* de simulation physique du FPGA développé, 2 fichiers doivent être fournis :

- quand la taille de la structure est connue, il faut remplir un fichier de connexion qui indique comment les pins de 2 tuiles adjacentes sont liées. Il indique également la charge placée sur les pins qui sortent de la structure. Pour finir, il contient les signaux d'entrée qui sont appliqués sur les pins d'entrés de la structure.
- le fichier de configuration contenant les $D_X \times D_Y \times 228$ bits de configuration doit être préparé manuellement.

Les simulations physiques valident la fonctionnalité. On constate que le passage de Y_1 de 0 à 1 provoque la transition de S_1 de 0 à 1. On peut également utiliser cette synthèse pour évaluer la capacité logique d'une tuile. Même si il est préférable d'avoir synthétisé un grand nombre de circuits larges sur le FPGA pour pouvoir faire une moyenne, la capacité d'une tuile est d'environ 20 portes logiques. En guise de comparaison, la famille IGLOO d'Actel est basée sur des tuiles ($VersaTile^{©}$) pouvant contenir 38 portes logiques équivalentes ([42]).

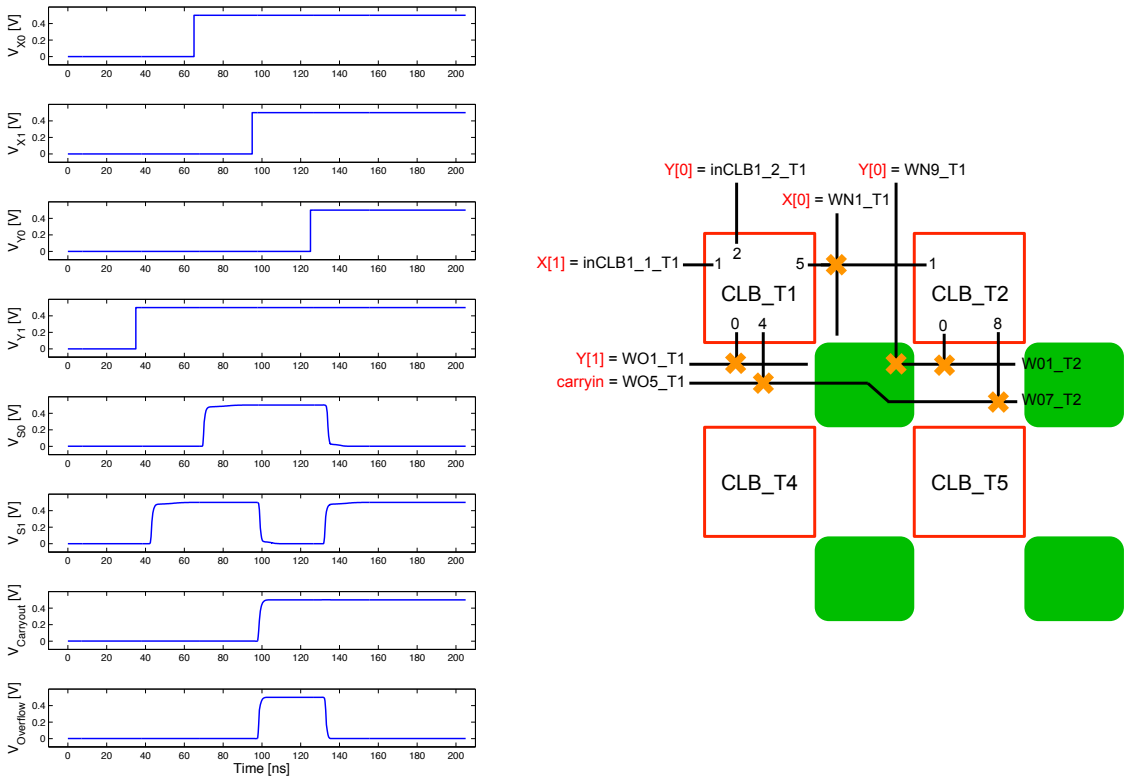


FIGURE 5.1: Synthèse et simulation d'un additionneur 2 bits sur 2×2 tuiles. Chaque entrée du circuit est routée jusqu'à une entrée du CLB.

5.2 VPR2ELDO

L'analyse des performances de la structure de FPGA développée dans ce travail est limitée par une contrainte pratique. Il faut pouvoir déterminer la valeur de 228 bits de configuration par tuile pour chacun des circuits à synthétiser. Dans le cas de l'additionneur sur 2 bits, cette opération a été réalisée manuellement mais cette méthode n'est pas applicable à des structures plus grandes que 2×2 tuiles.

Une couche *software* a été partiellement développée pour extraire une partie des bits de configuration des fichiers de placement et routage générés par VPR. La Figure 5.2 montre le fichier de configuration à générer pour réaliser la simulation physique.

L'utilitaire développé sous le nom de VPR2ELDO constitue les CLBs en rassemblant les BLEs listés dans le fichier .net sortant de T-VPACK et dans le fichier .blif entrant dans T-VPACK. L'algorithme génère alors les 16×4 bits de configuration des 4 LUTs et les 4 bits de configuration qui déterminent le contournement du DFF des BLEs. En partant des connections entre les entrées/sorties des CLBs et les entrées/sorties des BLEs, l'algorithme détermine les chemins passant à travers chaque multiplexeur de routage local et établit la succession de bits de configuration. L'algorithme n'est cependant pas encore capable d'extraire la configuration du routage global du fichier de routage de VPR. En revanche, il génère une représentation graphique du placement des CLBs sur la structure ainsi que du routage des différents *nets* à travers les CBs et SBs. La Figure 5.2 illustre le placement d'un multiplieur 16 bits et d'une des 28 connexions à router. Les Figures 3.4 et 3.5 montrent des placements et routages complets sur la structure développée.

A terme, ce logiciel, combiné au *CAD Flow* détaillé dans la Section 3.1, permettra de synthétiser un fichier verilog et d'en extraire la configuration à appliquer pour la simulation physique en utilisant le *framework* développé dans ce travail. Il sera également possible de générer le train de bits qui servira

```
* ------------- CLB ----------------
* BLE1
vCFBLE1_1_T1 CfBLE1_1_T1 gnd 0
vCFBLE1_2_T1 CfBLE1_2_T1 gnd ddram
vCFBLE1_3_T1 CfBLE1_3_T1 gnd ddram
vCFBLE1_4_T1 CfBLE1_4_T1 gnd 0
vCFBLE1_5_T1 CfBLE1_5_T1 gnd ddram
...
* MUXI2
vMUXI2_1_T1 CfMi2_1_T1 gnd ddram
vMUXI2_2_T1 CfMi2_2_T1 gnd 0
vMUXI2_3_T1 CfMi2_3_T1 gnd 0
vMUXI2_4_T1 CfMi2_4_T1 gnd 0
...
* ---------------------- CB_H ---------------
* MUXN1_H
vMUXN1_1_H_T1 CfMUXN1_1_H_T1 gnd 0
vMUXN1_2_H_T1 CfMUXN1_2_H_T1 gnd ddram
vMUXN1_3_H_T1 CfMUXN1_3_H_T1 gnd 0
vMUXN1_4_H_T1 CfMUXN1_4_H_T1 gnd 0
...
* ---------------------- SB ----------------
* Con1 Mux1
vCon1_Mux1_1_T1 CfCon1_Mux1_1_T1 gnd ddram
vCon1_Mux1_2_T1 CfCon1_Mux1_2_T1 gnd 0
vCon1_Mux1_3_T1 CfCon1_Mux1_3_T1 gnd ddram
vCon1_Mux1_4_T1 CfCon1_Mux1_4_T1 gnd 0
...
```

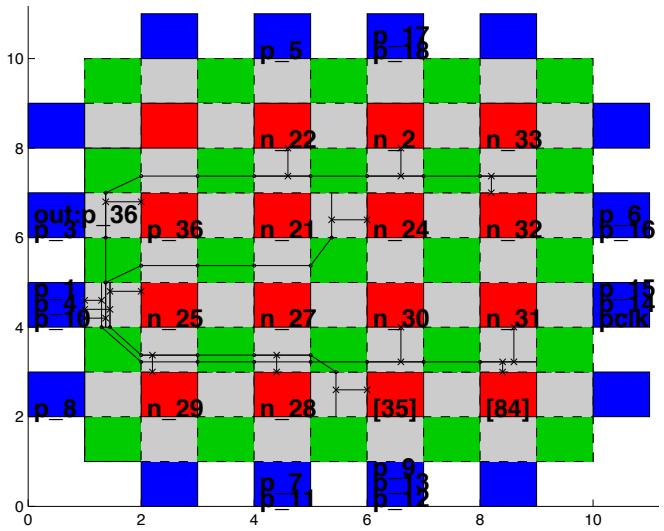


FIGURE 5.2: Gauche : portion d'un fichier de configuration d'une tuile servant à la simulation physique. A droite : illustration graphique du placement et routage d'un multiplieur 16 bits synthétisé sur un FPGA 4×4.

à la configuration du *test-chip* détaillé dans la Section 5.4.

5.3 *Corners* de fabrication

Avant de pouvoir réaliser un *test-chip* de la structure conçue, il faut garantir la fonctionnalité pour tous les *corners* de fabrication.

Les 4 *corners* sur les transistors ont été simulés :
- NMOS lent, PMOS lent.
- NMOS lent, PMOS rapide.
- NMOS rapide, PMOS lent.
- NMOS rapide, PMOS rapide.

Les simulations ont indiqué que la fonctionnalité d'un FPGA 2×2 sur lequel un additionneur 2 bits a été synthétisé, est préservée pour tous les *corners*. La Figure 5.3 reprend le délai sur le chemin critique de l'additionneur dans tous les *corners*.

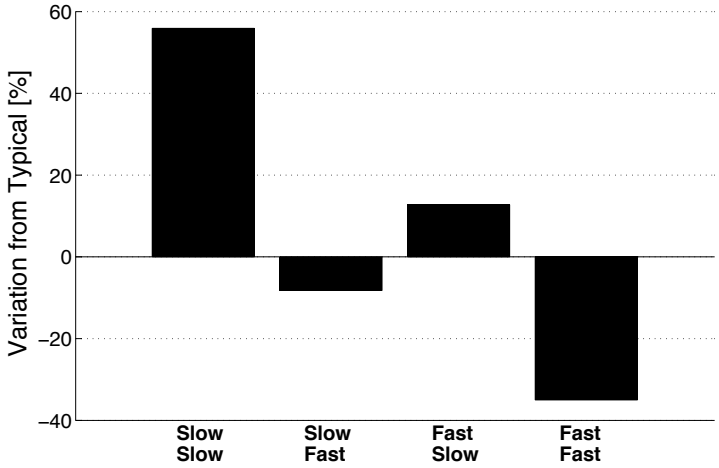


FIGURE 5.3: Evolution du délai sur le chemin critique pour chacun des 4 *corners* sur les transistors.

En plus des 4 *corners* sur les transistors, les *corners* sur la tension d'alimentation et sur la température ont été simulés. La tension d'alimentation a été augmentée et réduite de 10% par rapport à la tension d'alimentation nominale. Le FPGA a été simulé pour une température de $85^{\circ}C$ et de $-20^{\circ}C$. Pour ces 16 *corners*, les simulations ont prouvé que la fonctionnalité est garantie. L'Annexe A contient l'évolution des performances de l'additionneur 2 bits pour tous les *corners*.

La Figure 5.4 montre comment la répartition du délai sur le chemin critique entre délai intra-CLB et inter-CLB varie avec les *corners*. On constate que, à part pour le *corner Slow Slow*, les autres *corners* sont plus rapides que la simulation typique.

La Figure 5.5 montre comment le délai d'un BLE et d'un *switch block* évolue avec les *corners*.

La Figure 5.6 montre comment la puissance statique et dynamique évolue avec les *corners*. La puissance dynamique du FPGA 2×2 sur lequel l'additionneur 2 bits a été synthétisé, est évaluée en stimulant à $50MHz$ les entrées du circuit. Les segments et entrées des CLB non-utilisés sont laissés au repos. L'additionneur 2 bits occupe environ le quart de la structure 2×2, ce qui explique que la consommation statique est supérieure à la consommation dynamique.

La consommation statique du *corner Fast Fast* est supérieure à celle des autres *corners* à cause du courant de fuite supérieur des transistors de ce *corner*.

La consommation dynamique ne varie pas significativement avec les *corner* car les capacités de charge ne varient pas.

La consommation statique est dominée par le routage global pour tous les *corners*. La proportion de puissance dynamique consommée à l'intérieur des CLBs n'est pas représentative car, sur le circuit synthétisé, le routage global est peu utilisé. Il faut synthétiser un plus grand nombre de circuits de

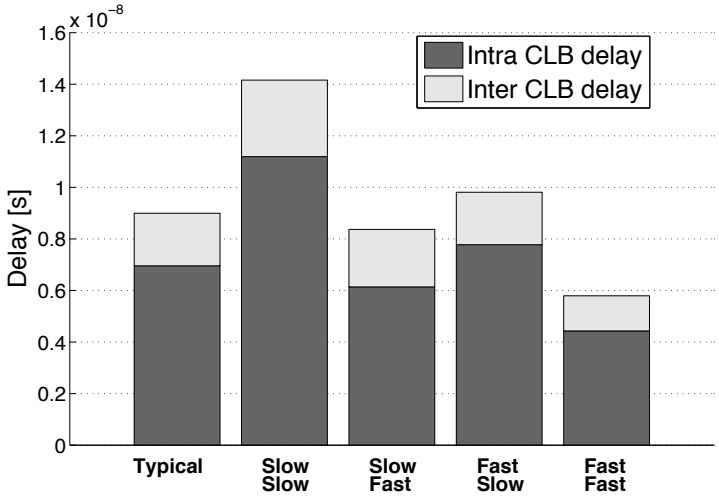


FIGURE 5.4: Détail du délai sur le chemin critique pour chacun des 4 *corners* sur les transistors.

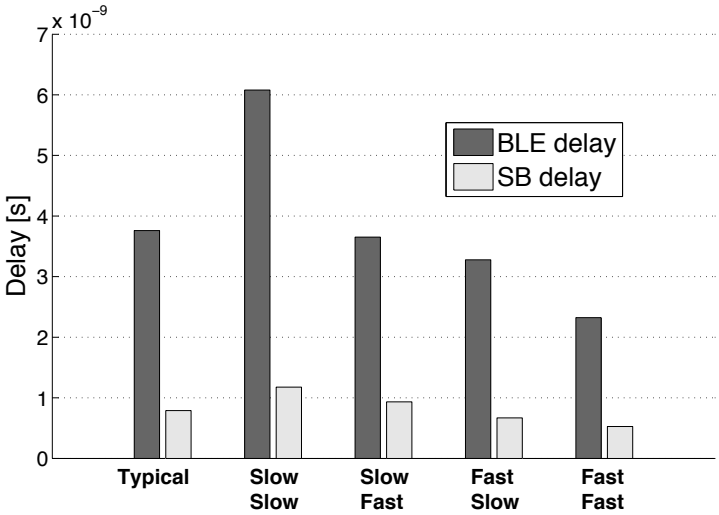


FIGURE 5.5: Evolution du délai dans un BLE et dans un *switch block* pour chacun des 4 *corners* sur les transistors.

benchmark plus complexes pour obtenir une indication significative de la répartition de consommation dynamique dans le FPGA. Ce type d'analyse sort du cadre de ce travail.

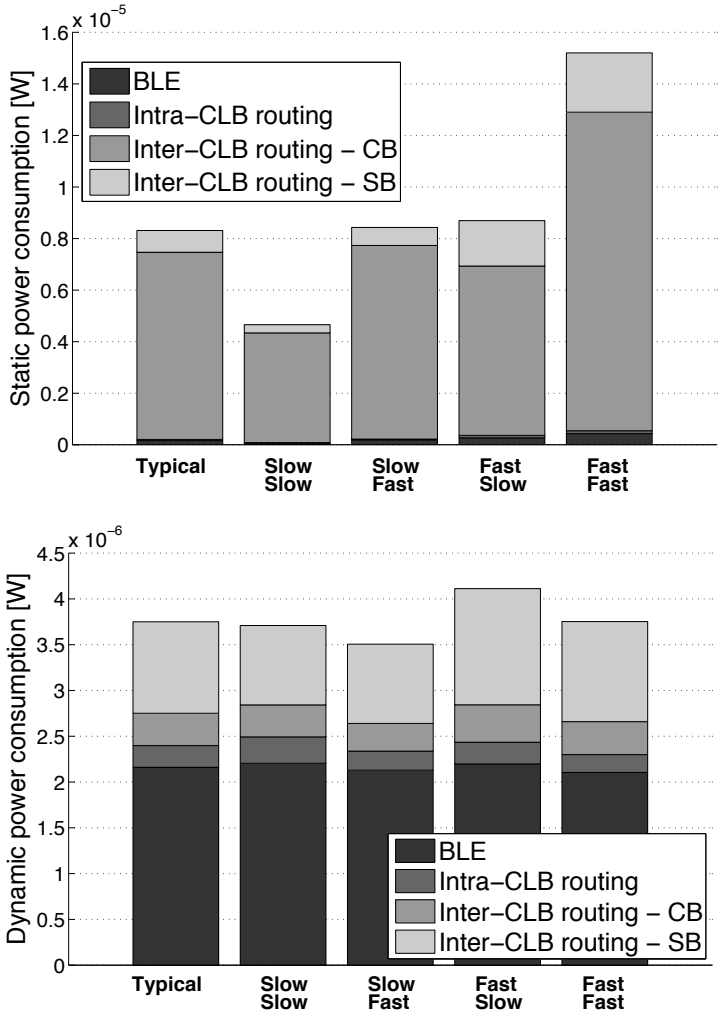


FIGURE 5.6: Evolution de la puissance statique et dynamique de la structure 2×2 simulée pour chacun des 4 *corners* sur les transistors.

5.4 Apercu du *test-chip*

Afin de valider les techniques de réduction de la puissance statique dissipée, la réalisation d'un *test-chip* en 65nm STMicroelectronics a été envisagée. Dans le cadre de cette étude, seul le *layout* du CLB a été réalisé pour accommoder la contrainte de temps avec les exigences du *design full custom*.

Le multiplexeur élémentaire 2 : 1 constitue la majorité de la surface d'un FPGA car il est utilisé pour former les LUTs et pour tous les multiplexeurs de routage local ou global. Deux types de multiplexeurs 2 : 1 ont été développés. Le premier, illustré en haut dans la Figure 5.7 (d), comporte un inverseur qui génère l'inverse du signal de sélection tandis que le second ne comprend pas cet inverseur. De cette façon, il a été possible de partager les lignes de sélection pour tous les multiplexeurs prenant le même signal de sélection.

Sur la Figure 5.7 (c), on repère facilement les lignes verticales de sélection en polysilicium. Sur cette illustration du *layout* du BLE, on distingue la LUT composée des 4 étages de multiplexeurs imbriqués et le registre en bas à droite.

La configuration dans ce prototype est conservée dans un registre à décalage. L'accès et la configuration du FPGA se fait à l'aide d'une horloge séparée. Le registre à décalage utilise les plus petits

Flip-Flop HVTLP de STMicroelectronics pour garder un courant de fuite faible (cf. Section 5.5). Bien que dense, ces *Flip-Flop* représentent une surface équivalente à la surface combinée des 4 BLEs et des 16 multiplexeurs de routage local. Afin de conserver un CLB dense, la configuration a été enroulée en spirale comme illustré dans la Figure 5.7 (a). Ce carré a environ un côté de $50\mu m$ de long.

La Figure 5.7 (b) reprend le *layout* final du CLB. On peut y repérer les 4 BLEs imbriqués les uns dans les autres en haut du carré pour limiter la surface totale. Au centre, on retrouve les 4 multiplexeurs 4 : 1 qui permettent de diriger les sorties des BLEs vers les sorties du CLB. En bas, on retrouve les 16 multiplexeurs 14 : 1 qui sélectionnent les entrées des BLEs. Ils sont groupés par 4 de façon à séparer les 4 multiplexeurs 14 : 1 correspondant à chaque BLE. Le routage de ces blocs, des 140 bits de configuration et des rails d'alimentation a été réalisé manuellement. La surface finale du CLB atteint :

$$S_{CLB} = 53.22\mu m \times 51.8\mu m = 0.002756mm^2 \tag{5.1}$$

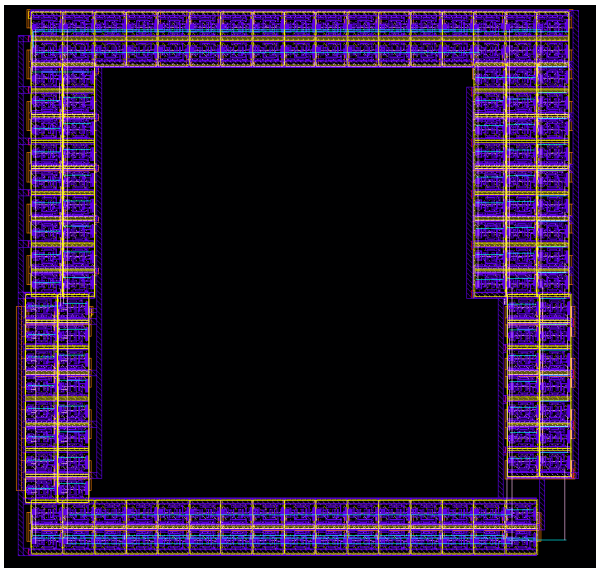
En évaluant la surface approximative de la tuile à $80\mu m \times 80\mu m$ avec l'ajout des canaux de routage verticaux et horizontaux et du SB, on peut conclure que le *layout* de ce FPGA permet de placer 156 tuiles par mm^2. En terme de portes logiques équivalentes, cela revient à 3120 portes par mm^2.

La densité atteinte par le FPGA conçu a été comparée à l'état de l'art en terme de FPGA à faible consommation. Les fabricants communiquent cependant peu sur la dimension des *dies*. Dans la même technologie, le *leader* en matière de faible consommation, SiliconBlue [1] propose le ICE65L08 ([58]) implémenté en 65nm *Low-Power* TSMC. Il s'agit du FPGA à faible consommation le plus proche du FPGA conçu dans ce travail. Le *die* de l'ICE65L08 mesure $4810\mu m \times 4394\mu m$ et comprend 7680 BLEs composés de LUTs de 4 entrées identiques à celles qui ont été conçues ici. SiliconBlue place donc 363 BLEs par mm^2 tandis que le *layout* proposé atteint 624 BLEs par mm^2. Le *layout full custom* réalisé dans ce travail est donc 40% plus dense. Ce calcul devrait cependant être affiné pour retirer du calcul les 2 cellules de RAM de $4Kbits$ que propose l'ICE65L08 mais SiliconBlue n'a pas communiqué leurs dimensions.

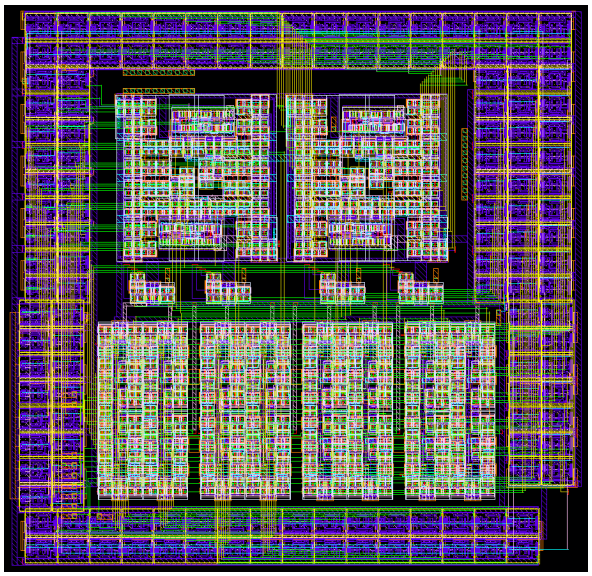
La structure contient différentes tensions d'alimentation. Le registre à décalage étant implémenté en $HVTLP$, sa tension d'alimentation est laissée à $1V$ car son courant de fuite sera faible. Les caractéristiques du registre à décalage sont présentées dans la Section 5.5. Le reste du CLB est alimenté à $0.5V$. Pour apporter les signaux d'entrées provenant des *pads* d'entrée vers le FPGA, il faut insérer des *down shifter* et pour amener les sorties du CLB vers les *pads* de sortie, il faut ajouter des *up shifter*. Ces blocs et le CLB équipés des *level shifters* sont illustrés dans l'Annexe B.

Le *test-chip* doit être envoyé en fabrication en octobre 2012 et sera alors constitué de 3×3 tuiles permettant la synthèse de circuits de près de 200 portes logiques.

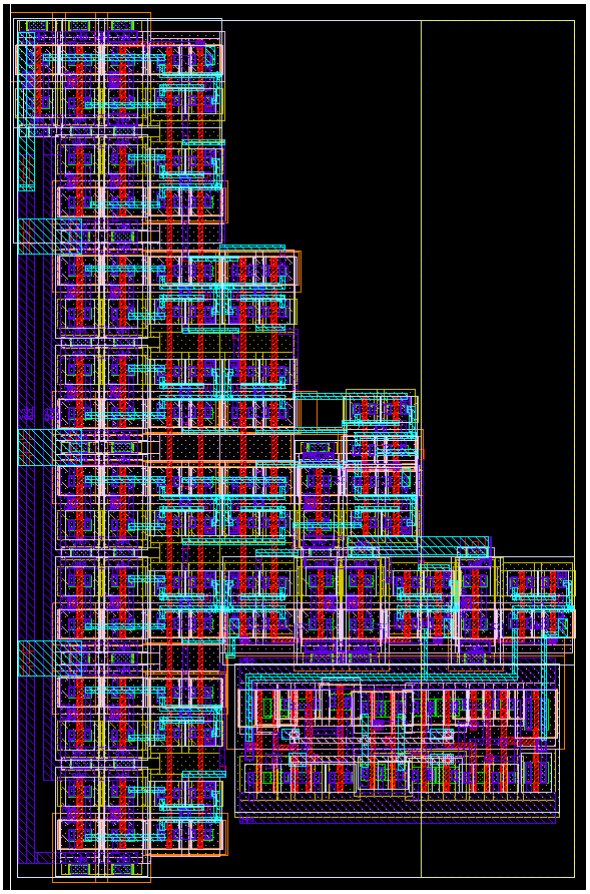
1. SiliconBlue est une *spin-off* spécialisée dans la conception de FPGAs à faible consommation basés sur des SRAM mais proposant une mémoire non-volatile *in-die*.SiliconBlue a été racheté en 2011 par Lattice, le 3^{me} acteur sur la scène des FPGAs.



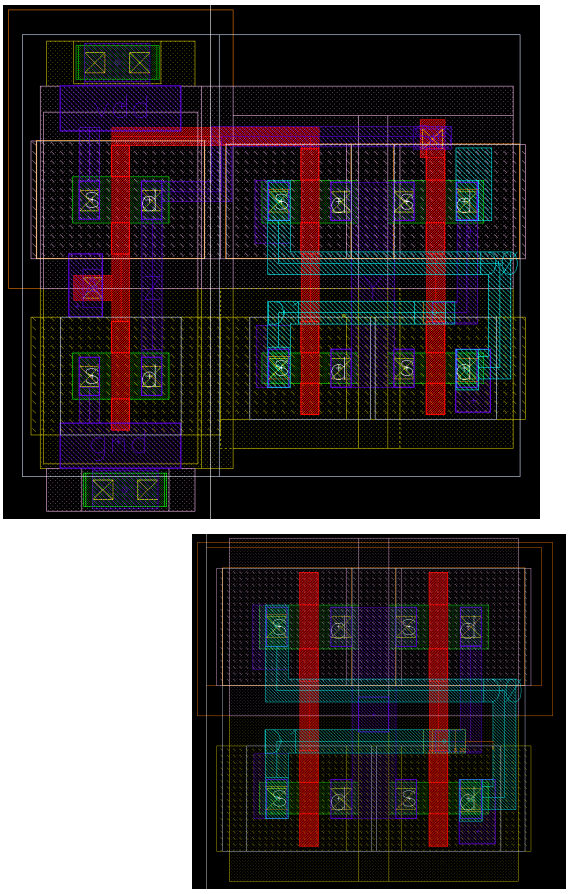
(a) Registre à décalage de configuration



(b) CLB



(c) BLE



(d) Multiplexeur 2 : 1 avec et sans inverseur de sélection

FIGURE 5.7: Illustration du *layout* des différents blocs consistuant le CLB et du *layout* du CLB lui-même.

5.5 Simulations post-*layout*

Afin de valider le circuit après *layout*, une série de simulations SPICE a été réalisée.

5.5.1 Configuration

Le registre à décalage contenant la configuration a été simulé pour valider sa fonctionnalité. Il n'est, en effet, pas directement lié aux performances du FPGA. La Figure 5.8 illustre la configuration du CLB.

TABLE 5.1: Performances du registre à décalage

$@20MHz$	Leakage Power	Total power	Configuration delay
one bit	$21.355[pW]$	$3.546[\mu W]$	$50[ns]$
one CLB	$2.989[nW]$	$0.496[mW]$	$7[\mu s]$
3×3 FPGA	$43.82[nW]$	$7.277[mW]$	$102[\mu s]$

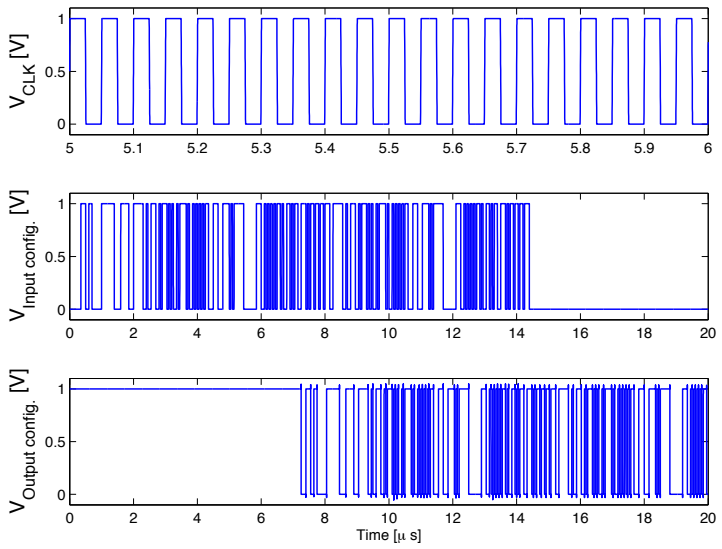


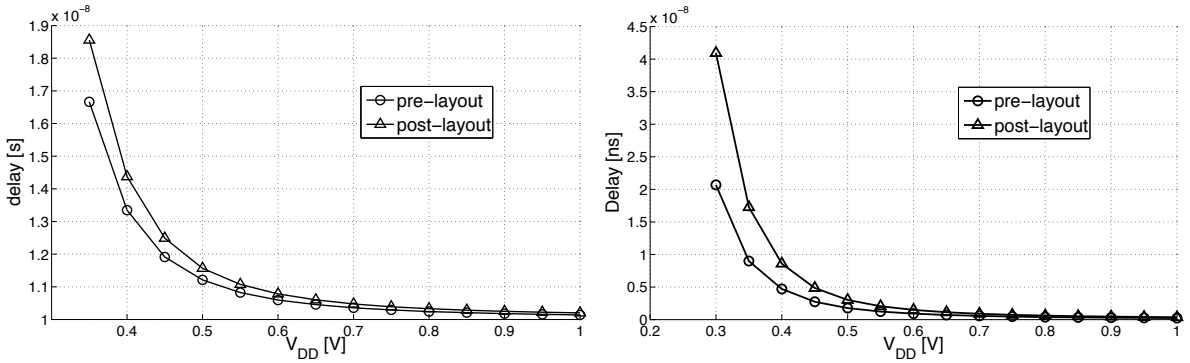
FIGURE 5.8: Simulation du registre à décalage lors de la programmation du FPGA. On constate qu'après 140 périodes de l'horloge de configuration ($20MHz$) soit $50ns \times 140 = 7\mu s$ dans la simulation, les premières entrées du registre sortent.

Comme l'indique la Table 5.1, l'utilisation de transistors LP pour la configuration permet de maintenir la puissance statique négligeable malgré l'augmentation de la taille du FPGA. En revanche, la consommation dynamique durant la configuration du FPGA est importante. La vitesse de l'horloge de configuration doit donc être adaptée selon la puissance disponible dans le système lors de la configuration.

5.5.2 LUT & BLE

Après *layout*, les parasites capacitifs et résistifs ont été extraits pour le BLE et la LUT. Le BLE et la LUT annotés de ces parasites ont été simulés et comparés aux résultats obtenus avant le *layout*. La Figure 5.9 illustre la dégradation du délai par les parasites. Le BLE montre une dégradation de son délai de 5% à $0.5V$ qui reste raisonnable. Sa fonctionnalité après *layout* est validée.

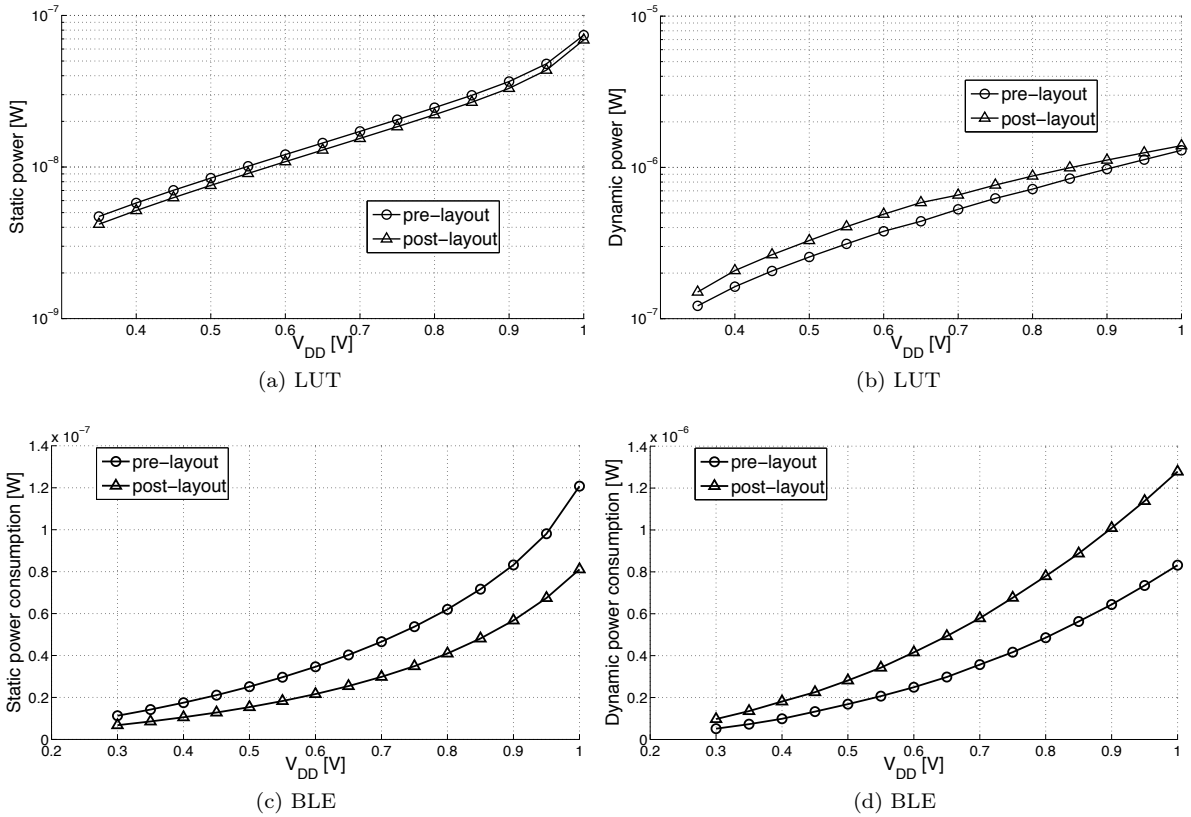
La Figure 5.10 illustre la variation des puissances statique et dynamique du BLE et de la LUT entre la simulation pré-*layout* et la simulation post-*layout*. On remarque que la consommation statique des



(a) BLE (b) LUT

FIGURE 5.9: Evolution du délai avant et après *layout* pour la LUT et le BLE.

deux blocs diminue légèrement après *layout* (−10% @ 0.5V pour la LUT) suggérant que les courants de fuite pré-*layout* ont été légèrement surestimés. La puissance dynamique a, quant à elle, augmenté de 65% @ 0.5V pour le BLE.



(a) LUT (b) LUT

(c) BLE (d) BLE

FIGURE 5.10: Evolution des puissances statique et dynamique avant et après *layout* pour la LUT et le BLE.

Chapitre 6

Conclusion

De nouveaux défis de mobilité et d'ubiquité émergent avec le développement des systèmes autonomes en énergie. La recherche des 10 dernières années a permis de rendre différents secteurs tels que le *processing* digital, la conversion analogique/digitale ou la transmission RF compatibles avec la densité d'énergie associée à des systèmes sans contact avec un réseau d'énergie.

En 2009, le programme CATRENE (*Cluster for Application and Technology Research in Europe on NanoElectronics* [59]) rendait un rapport indiquant que dans un futur proche, l'utilisateur lambda devrait être en mesure d'acheter un système d'*energy harvesting* de la même manière qu'il achèterait une batterie aujourd'hui. La course à l'autonomie est lancée.

Si la technologie ULP et la récupération locale d'énergie progressent à grand pas, la conception de SoC ambitieux en terme de fonctionnalité se heurte au problème de la flexibilité. Dans ce travail, la question de l'intégration de flexibilité *hardware* dans un budget de l'ordre du μW a été étudiée. Loin de la prétention d'avoir répondu à cette ambitieuse question, certains éléments de réponses ont été apportés et sont résumés ci-dessous.

Résultats principaux

Dans un premier temps, un double état de l'art a été dressé dans le Chapitre 2. Celui-ci comprend une partie consacrée aux FPGAs traitant des principaux points d'architecture et ciblant la course aux performances en vitesse et en surface que livrent les industriels face aux ASICs. La seconde partie de cet état de l'art se concentre sur la conception de circuits digitaux ULP. Là aussi, les différentes techniques développées sont citées avec une attention particulière sur les travaux du groupe ULP de l'Université catholique de Louvain qui est à la pointe dans ce domaine.

Le Chapitre 3 est centré sur une analyse haut-niveau de la structure reconfigurable. La série d'outils utilisés pour synthétiser, placer et router un circuit sur une structure reconfigurable, est analysée. L'architecture du FPGA, définie par un grand nombre de paramètres, est optimisée au travers d'une étude rigoureuse de l'impact de ces paramètres et de la topologie des différents blocs sur la puissance consommée et sur la vitesse. L'impact de la hiérarchisation des blocs logiques élémentaires en grappe sur la puissance est modélisé. Le modèle développé dans ce travail permet d'optimiser la puissance sur K et N sans avoir à estimer euristiquement l'activité en chaque noeud du FPGA pour différents circuits de *benchmark*. Le modèle replace ces informations d'activité, difficiles à obtenir, en intégrant les taux d'utilisation moyen des différentes ressources de routage et de logique du FPGA données par le *CAD Flow*. Cette technique permet de réduire considérablement le coût calculatoire de l'estimation de la puissance avec K et N tout en conservant de bons résultats en accord avec les optima cités dans la littérature. Une courte comparaison des différentes architectures de routage global a également été réalisée.

Le Chapitre 4 contient les contributions les plus significatives de ce travail. Il traite de l'implémentation ULP ULV de l'architecture développée dans le chapitre précédent. Le choix du type de logique à utiliser pour implémenter les LUTs alimentées sous-seuil est discuté et la logique de transmission est choisie réduisant ainsi de 50% le produit délai-puissance totale. La possibilité d'intégration d'un dispositif de *power gating* dans le FPGA est ensuite évaluée. Ce type de dispositif possède un large potentiel de réduction de la consommation statique car, si il est réalisé à fin grain, il peut couper toutes les ressources inutilisées. Différentes stratégies de *power gating* sont envisagées et un système de *power gating with adaptative forward body biais* est proposé. Il permet de réguler la résistance parasite introduite entre l'alimentation virtuelle du circuit et l'alimentation réelle selon la demande en courant du circuit. Les EAS sont généralement caractérisés par un faible facteur d'activité. Un noeud d'un *wireless sensor network* par exemple sera inactif pendant la majorité du temps. Dans cette optique, il est intéressant de pouvoir réduire la consommation statique. Dans ce travail, les différentes tensions de seuil mises à disposition par la technologie $65nm$ de STMicroelectronics ont été pleinement utilisées. Le choix du V_T de chaque bloc du FPGA a été sélectionné pour contrôler précisément le compromis consommation statique-vitesse. Grâce à cet outil de conception et en alimentant la structure à 0.5V, 86% de la puissance totale a pu être économisée. La répartition optimale du V_T, à elle seule, a permis une réduction de 33% de la puissance statique par tuile. A plus haut niveau, les simulations après synthèse d'un multiplieur 16 bits sur le FPGA conçu ont montré une consommation de $2.5pJ$ par multiplication soit $11\times$ moins que l'implémentation de la même multiplication sur un microcontrôleur ULP ULV de l'état de l'art utilisant les mêmes techniques de répartition du V_T. Ces résultats montrent que la reconfiguration *hardware* peut se poser en intermédiaire entre l'ASIC et la reconfiguration *software* dans le domaine de l'ULP, ce qui n'avait jamais été étudié.

Cette technique de partition du circuit a fait l'objet d'une publication présentée à la conférence FTFC 2012.

Le dernier chapitre concerne la validation des résultats énoncés dans le chapitre précédent. Afin de valider la fonctionnalité du FPGA, 4 tuiles sont jointes et un petit additionneur sur 2 bits est synthétisé manuellement. Pour pouvoir simuler et utiliser le FPGA conçu à plus grande échelle, une couche à l'interface entre le *CAD Flow* académique analysé dans le Chapitre 3 et le *framework* de simulation physique développé dans le Chapitre 4, a été réalisée. Ce logiciel permet de visualiser graphiquement le placement et routage d'un circuit sur le FPGA et fournit également les bits de configuration des CLBs. A ce stade du développement, les bits de configuration du routage global doivent cependant être fixés manuellement.

Ce prototype réduit a également permis de simuler les *corners* de fabrication sur les transistors, la température et les alimentations.

Pour valider les résultats obtenus par simulation, un *test-chip* a été conçu. Afin d'adapter les contraintes de temps liées à ce travail à l'effort du *design full custom*, seul le *layout* d'un CLB a été réalisé. Le *text-chip* final qui sera envoyé en fabrication en octobre 2012 sera constitué de 3×3 tuiles permettant la synthèse de circuits d'environ 200 portes logiques.

Recherche et applications : perspectives

Bien que les résultats soient encourageants, de nombreuses étapes restent à franchir avant d'atteindre l'objectif : une IP de logique reconfigurable performante et ULP intégrable dans un SoC autonome en énergie.

Dans un premier temps, la structure de FPGA optimisé proposée dans ce travail doit être exhaustivement testée en simulant physiquement de grands FPGAs 10×10 ou plus. Pour parvenir à synthétiser de grands circuits sur le FPGA, la couche logicielle située à l'interface entre le *CAD Flow* académique et le *framework* de simulation SPICE doit être terminée.

La possiblité d'utiliser un dispositif de *power gating* a été partiellement étudiée dans ce travail mais certaines questions demeurent. En premier lieu, quel est le grain optimal du *power gating* pour réduire au maximum la puissance statique des sections de FPGA inutilisées ?

Les FPGAs modernes incluent des blocs hétérogènes ou des structures arithmétiques comme des chaines de *carry* pour améliorer les performances des circuits synthétisés. Afin de synthétiser des applications arithmétiques à haut *throughput* qui sont utilisées dans de nombreux périphériques ou coprocesseurs, l'implémentation ULP ULV de ces blocs devrait être étudiée.

A haut niveau, une couche *CAD* devrait être développée pour intégrer des algorithmes de placement et routage qui prennent en compte l'objectif de réduction de la puissance consommée. L'implémentation ULV subit une réduction des marges de bruit qui peuvent générer des fautes dans le FPGA. La couche *software* peut être conçue pour contourner ces structures fautives pour maintenir la fonctionnalité du FPGA.

Finalement, toutes les *guidelines* ULP doivent être regroupées dans un outil de génération automatique d'IP permettant la génération de l'architecture optimale des blocs logiques et des canaux de routage en partant de spécifications haut-niveau telles que la capacité logique à atteindre, la surface approximative ou bien la consommation statique maximum acceptable. De la même manière, un outil de génération automatique de *layout* devrait être développé afin de permettre l'intégration de cette IP *full custom* dans le flot de conception d'un SoC.

Annexe A

Corners

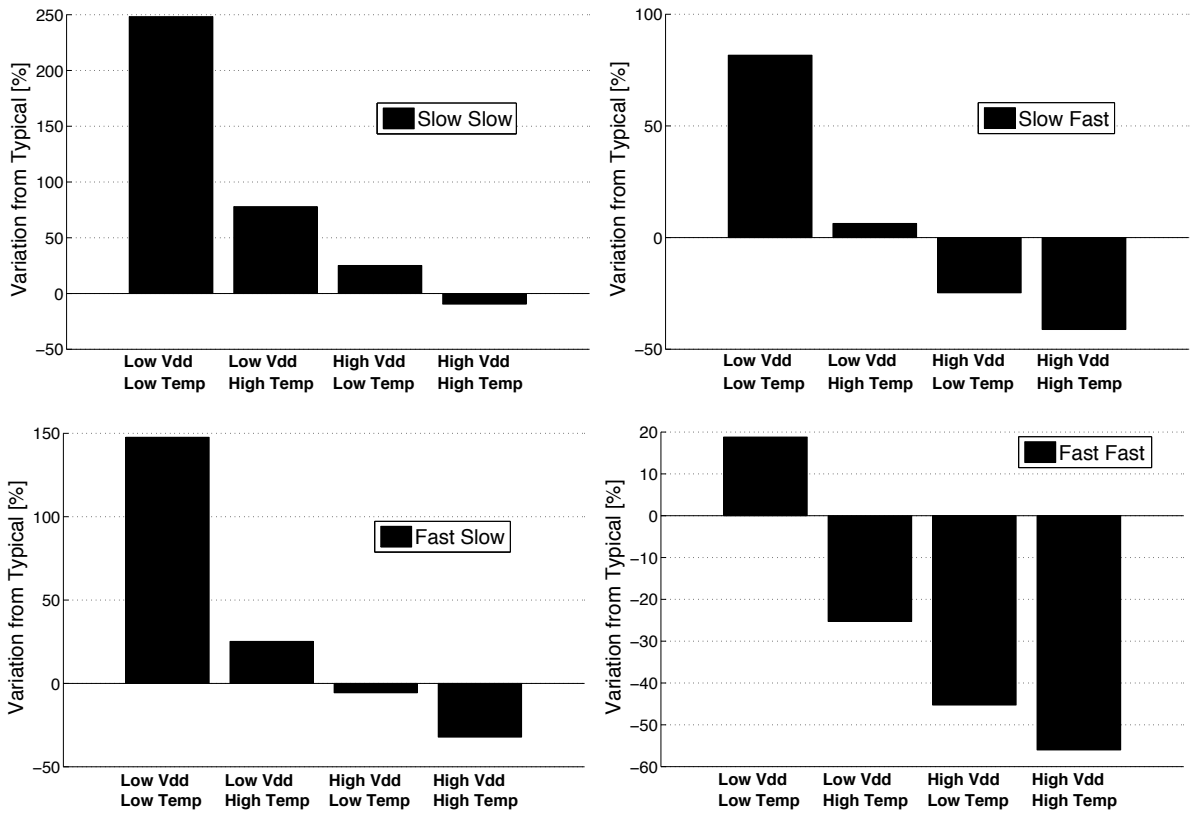


FIGURE A.1: Evolution du délai sur le chemin critique pour chacun des 16 *corners* (2 pour les NMOS, 2 pour les PMOS, 2 pour la tension d'alimentation et 2 pour la température).

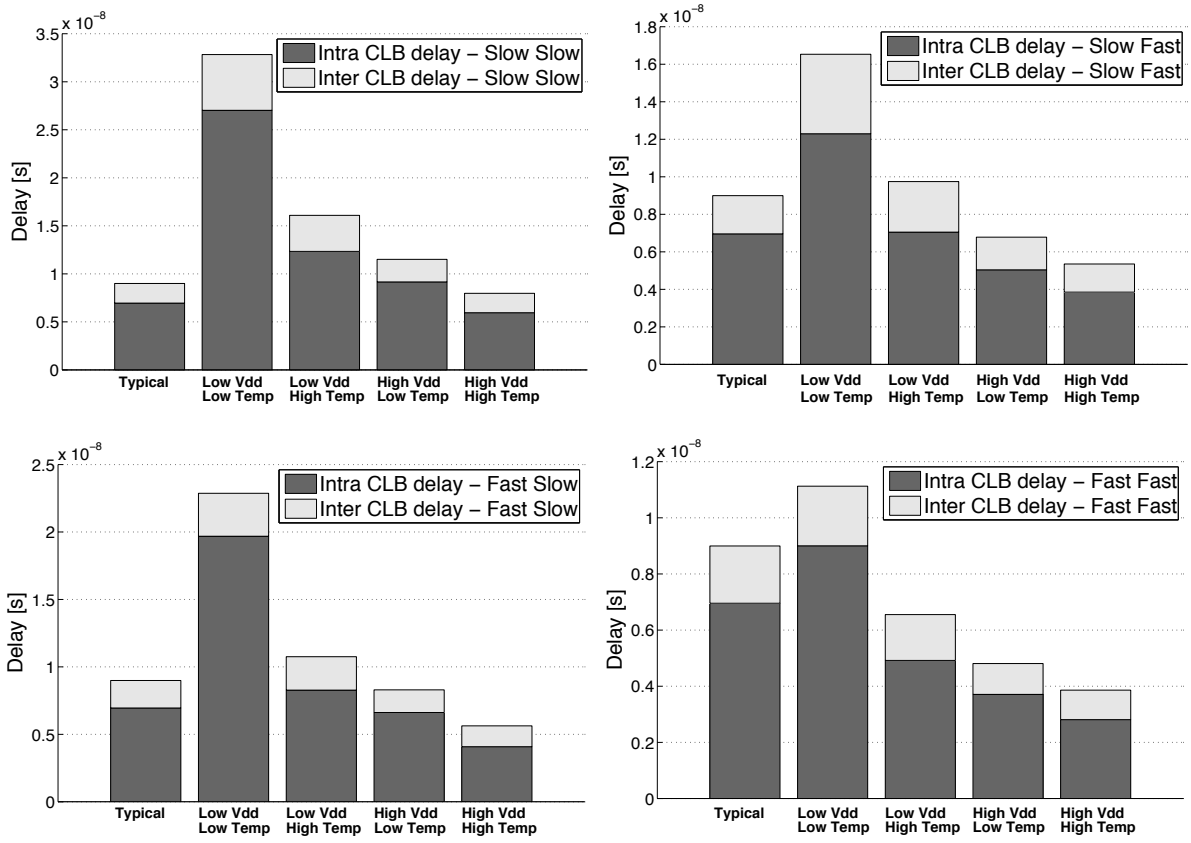


FIGURE A.2: Détail du délai sur le chemin critique pour chacun des 16 *corners*.

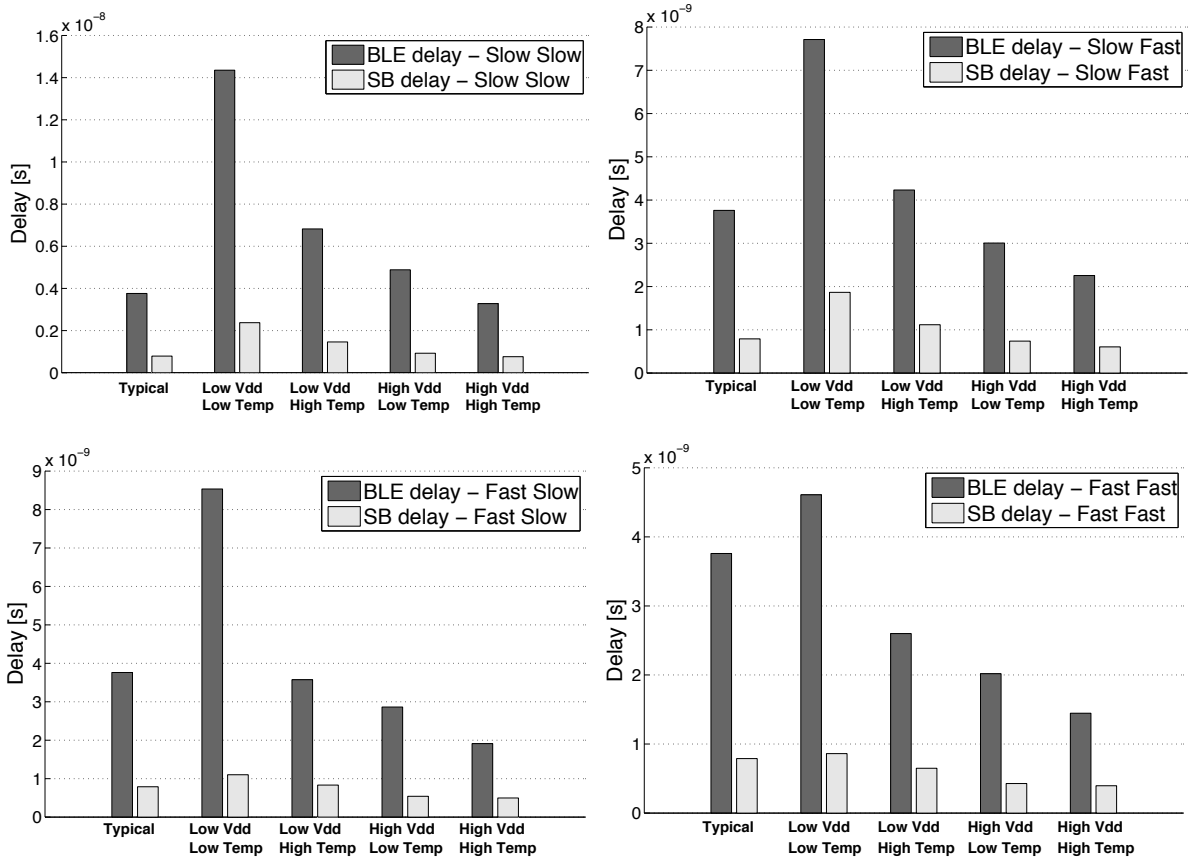


FIGURE A.3: Evolution du délai d'un BLE et d'un *switch block* pour chacun des 16 *corners*.

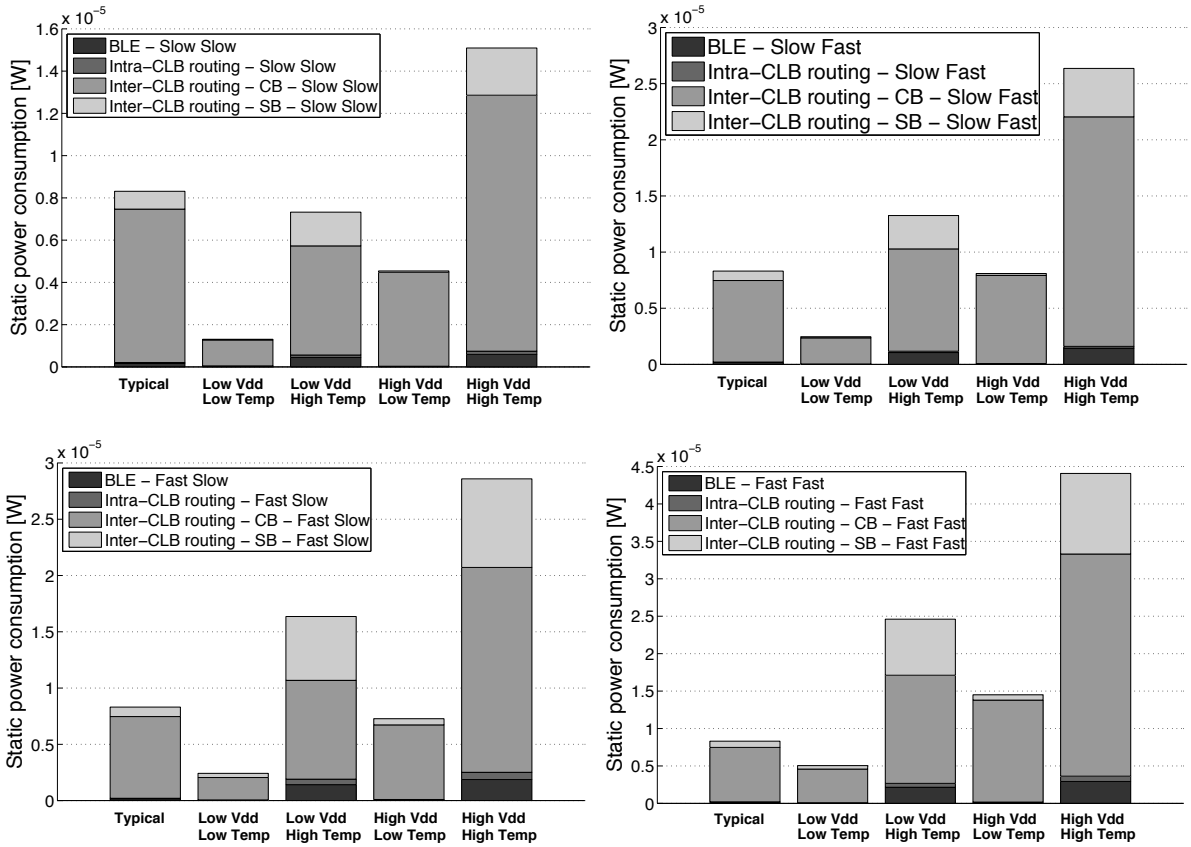


FIGURE A.4: Evolution de la puissance statique consommée par la structure 2×2 pour chacun des 16 *corners*.

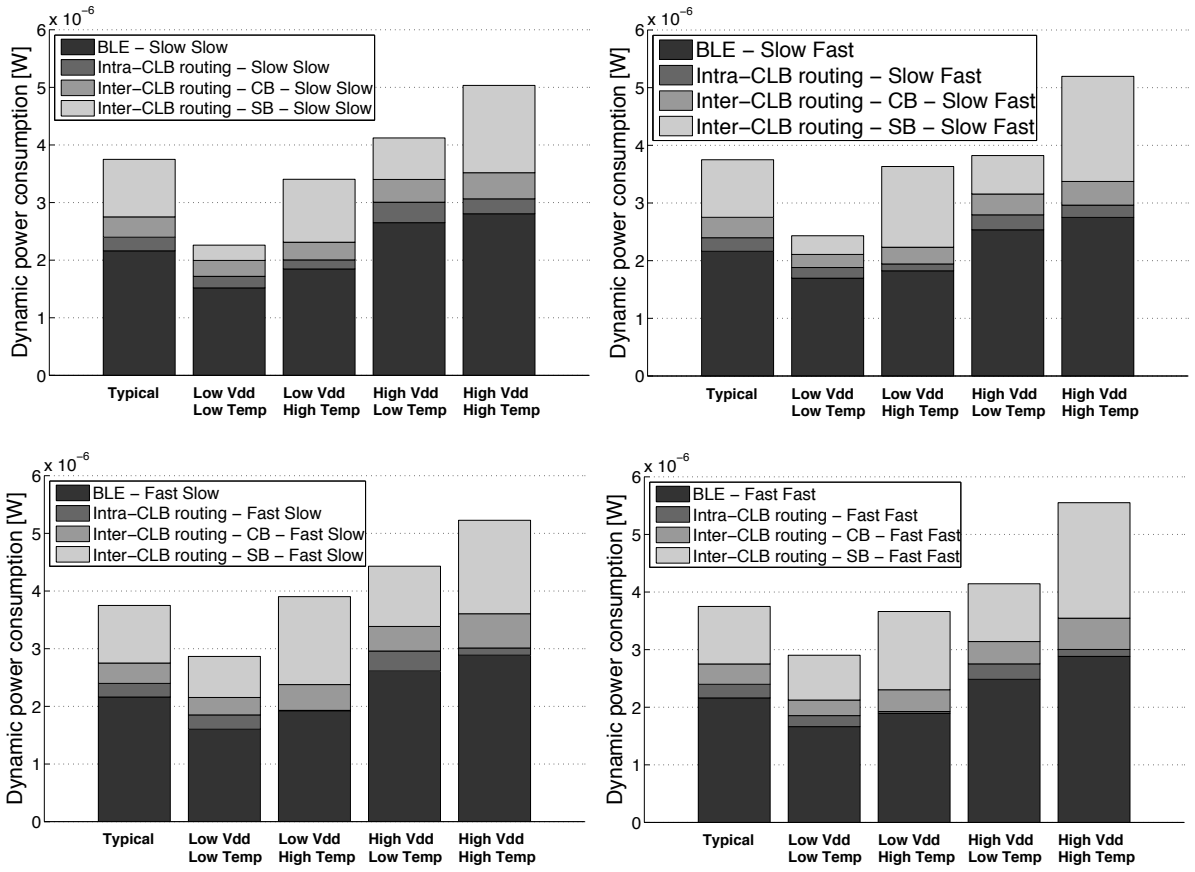


FIGURE A.5: Evolution de la puissance dynamique consommée par l'additionneur 2 bit pour chacun des 16 *corners*.

Annexe B

Layout

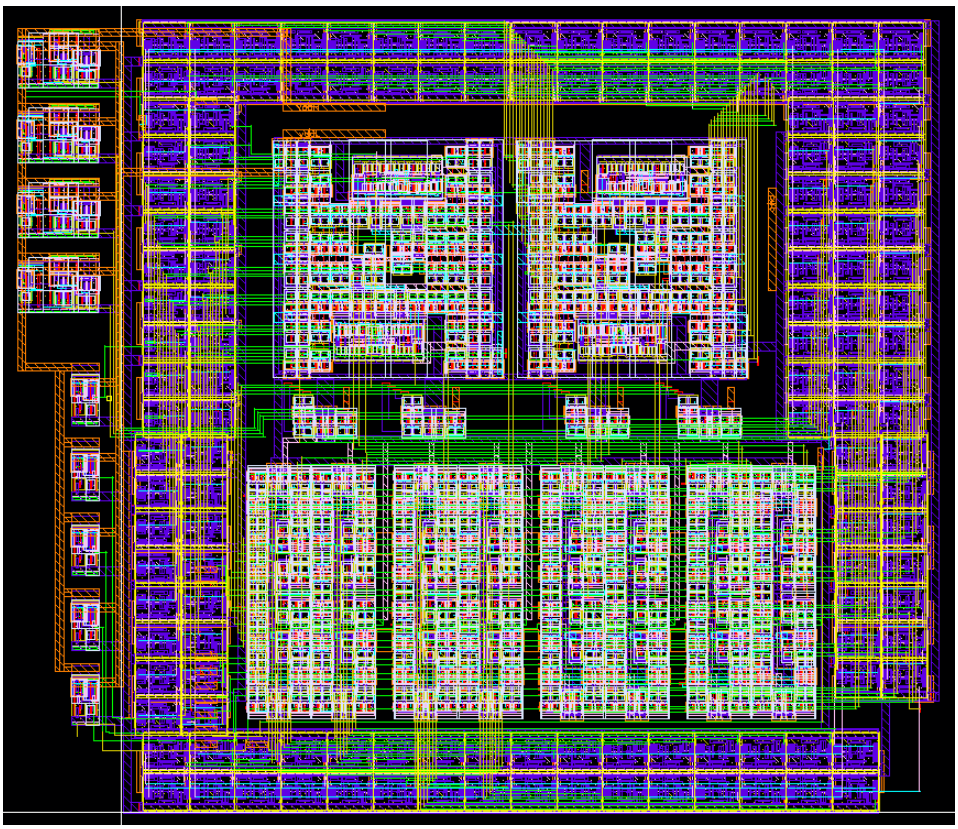
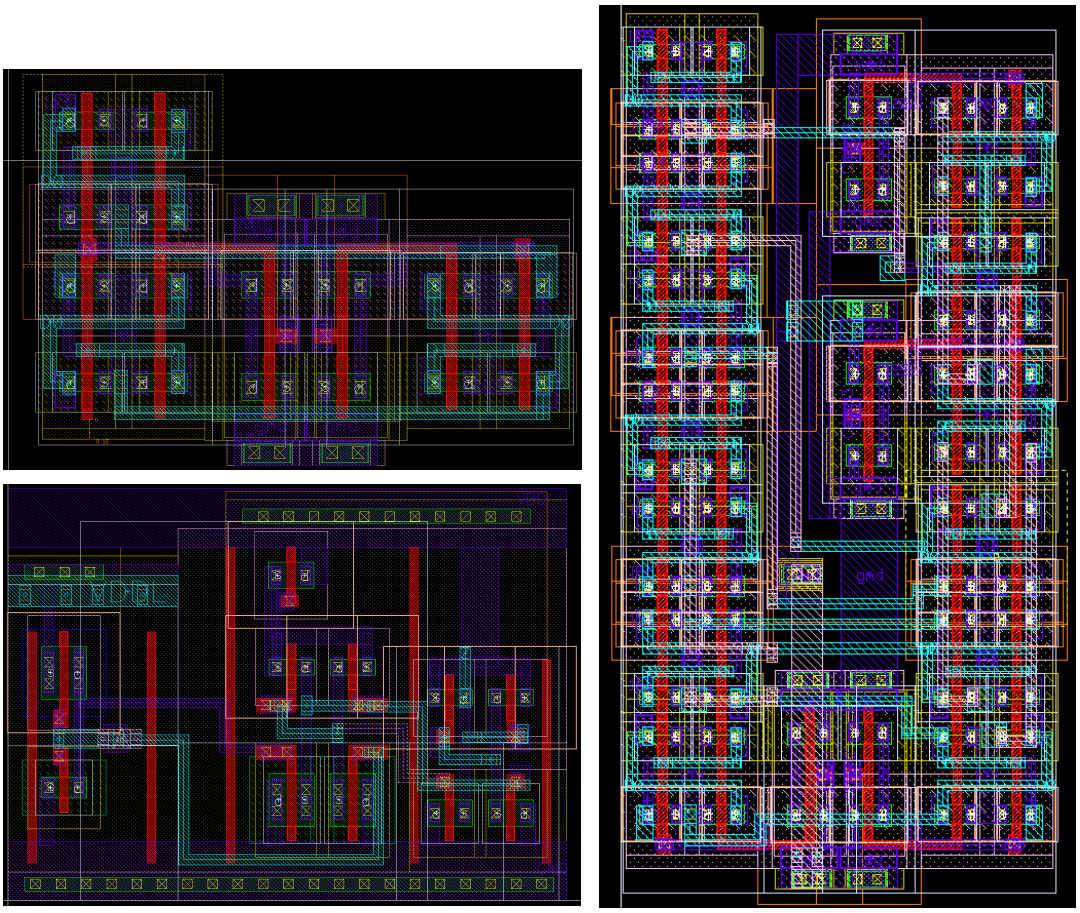


FIGURE B.1: *Layout* du CLB et de ses *level shifter* d'interface avec les I/O *pads*.



(a) En Haut : multiplexeur 4 : 1. En bas : *up shifter*.

(b) Multiplexeur 14 : 1.

FIGURE B.2: *Layout* des multiplexeurs de routage local et du *level shifter* passant de V_{DDL} à V_{DDH}.

Bibliographie

[1] I. Kuon and al., "FPGA Architecture : Survey and Challenges", in *Foundations and Trends in Electronic Design Automation*, pp. 135-253, 2007.

[2] M. Zhang and al., "Performance Comparison of SRAM Cells Implemented in 6,7 and 8-Transistor Cell Topologies", in *proc. JSSC*, 2006.

[3] K. Agarwal, S. Nassif, "Statistical Analysis of SRAM Cell Stability", in *Design Automation Conference*, pp. 57-62, 2006.

[4] T. Speers and al., "0.25 μm FLASH Memory Based FPGA for Space Applications", in *International Conference on Military and Aerospace Programmable Logic Devices*, pp. 6-9, 1999.

[5] C. Hu, "Interconnect Devices for Field Programmable Gate Array", in *Electron Devices Meeting*, pp. 591-594, 1992.

[6] S. Chiang and al., "Antifuse Structure Comparison for Field Programmable Gate Arrays", in *International Electron Devices Meeting 1992 Technical Digest*, pp. 611-614, December 1992.

[7] S. Chih-Ching and al., "Characterization and Modeling of a Highly Reliable Metal-to-Metal Antifuse for High-Performance and High-Density Field-Programmable Gate Arrays", in *Reliability Physics Symposium*, pp. 25-33, 1997.

[8] A. S. Sedra, K. C. Smith, "Microelectronic Circuits Revised Edition", *Oxford Series in Electrical and Computer Engineering*, 2007.

[9] Actel Corporation, "SX-A family FPGAs v5.3", `http://www.actel.com/documents/SXA_DS.pdf`, February 2007.

[10] A. El Gamal and al., "An Architecture for Electrically Configurable Gate Arrays", in *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 394-398, April 1989.

[11] H.-C. Hsieh and al., "A 9000-Gate User-Programmable Gate Array", in *Custom Integrated Circuits Conference*, pp. 15.3/1-15.3/7, May 1988.

[12] J. Rose and al., "Architecture of Field-Programmable Gate Arrays : The Effect of Logic Block Functionality on Area Efficiency", in *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1217-1225, October 1990.

[13] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288-298, March 2004.

[14] V. Betz and J. Rose, "How Much Logic Should Go in an FPGA Logic Block ?", in *IEEE Design and Test of Computers*, vol. 15, no. 1, pp. 10-15, January-March 1998.

[15] V. Betz and J. Rose, "Cluster-Based Logic Blocks for FPGAs : Area-Efficiency vs. Input Sharing and Size", in *Custom Integrated Circuits Conference*, pp. 551-554, May 1997.

[16] A. Marquardt and al., "Speed and Area Tradeoffs in Cluster-Based FPGA Architectures", in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 1, pp. 84-93, Feb. 2000.

[17] J. Luu et al., "VPR 5.0 : FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling", in *FPGA'09*, Feb. 2008.

[18] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0", Tech. Report, Microelectronics Center of North Carolina, 1991.

[19] G. de Streel and al., "Multi-V_T Ultra-Low-Power FPGA Implementation in 65nm CMOS Technology", submitted to FTFC 2012 conference.

[20] F. Li and al., "Power Modeling and Characteristics of Field Programmable Gate Arrays", in *IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712-1724.

[21] V. Betz and J. Rose, "FPGA Routing Architecture : Segmentation and Buffering to Optimize Speed and Density", in *Proceeding : ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 140-149, February 1999.

[22] G. Lemieux and al., "Directional and Single-Driver Wires in FPGA Interconnect", in *Proceedings : International Conference on Field- Programmable Technology*, pp. 41-48, December 2004.

[23] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays", in *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277-282, 1991.

[24] S. Wilton, "Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories", PhD thesis, University of Toronto, Department of Electrical and Computer Engineering, 1997.

[25] Y.-W. Chang and al., "Universal Switch-Module Design for Symmetric-Array-Based FPGAs", in *FPGA '96. Proceedings of the 1996 ACM Fourth International Symposium on Field-Programmable Gate Arrays*, pp. 80-86.

[26] M. Imran and al., "A New Switch Block for Segmented FPGAs", in *Lectures Notes in Computer Science : Field Programmable Logic and Applications*, pp. 274-281, 1999.

[27] D. Lewis and al., "The *Stratix*^{TM} Routing and Logic Architecture", in *Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays*s, pp. 12-20, ACM Press, 2003.

[28] V. Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs". Kluwer Academic Publishers, 1999.

[29] Xilinx Corporation, "Virtex-5 Family Overview", `http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf`, February 2009.

[30] Altera Corporation, "Stratix III Device Family Overview", `http://www.altera.com/literature/hb/stx3/stx3_siii51001.pdf`, March 2010.

[31] Altera Corporation, "MAX II Device Handbook", `http://www.altera.com/literature/hb/max2/max2_mii5v1.pdf`, March 2010.

[32] D. Bol, "Pushing Ultra-Low-Power Digital Circuits into the Nanometer Era", Ph. D Thesis, Université catholique de Louvain, 2008.

[33] D. Flandre, "ELEC2 Dispositifs avancés", syllabus de cours, Université catholique de Louvain, 2010.

[34] B. H. Calhoun and al., "Flexible Circuits and Architectures for Ultralow Power", in *Proceedings of the IEEE*, vol. 98, No. 2, Feb 2010.

[35] Y. Pu and al., "Statistical Noise Margin Estimation for Sub-Threshold Combinational Circuits", in *Design Automation Conference*, pp. 176-179, March 2008.

[36] D. Bol, "Robust and Energy-Efficient Ultra-Low-Voltage Circuit Design under Timing Contraints in 65/45 nm CMOS", in *Journal of Low Power Electronics and Applications*, 2011.

[37] M. Seok and al., "Sleep Mode Analysis and Optimization with Minimal-Sized Power Gating Switch for Ultra-low Vdd Operations", in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.20, pp.605-615, 2012.

[38] D. Bol and al., "Robustness-Aware Sleep Transistor Engineering for Power-Gated Nanometer Subthreshold Circuits", in *IEEE international Symposium on Circuits and Systems (ISCAS)*, pp. 1484-1487, 2010.

[39] F. Li and al., "Low-Power FPGA Using Pre-defined Dual-Vdd/Dual-Vt Fabrics", in *FPGA '04 Proceedings of the ACM/SIGDA*, 2004.

[40] A. Gayasen and al., "A Dual-Vdd Low Power FPGA Architecture", in *Proceedings of International Conference on Field Programmable Logic and Applications*, pp. 145-157, 2004.

[41] F. Botman and al., "Exploring the Opportunity of Operating a COTS FPGA at 0.5V", in *Proc. IEEE Subthreshold Microelectronics Conference*, 2011.

[42] Actel Corporation, "IGLOO nano Low Power Flash FPGAs with Flash*Freeze Technology" `http://www.actel.com/documents/IGLOO_nano_DS.pdf`.

[43] Actel Corporation, "ProASIC3L Low Power Flash FPGAs with Flash*Preeze Technology" `http://www.actel.com/documents/PA3L_DS.pdf`.

[44] V. Aken'Ova, "An Improved "soft" eFPGA Design and Implementation Strategy", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 179-182, 2005.

[45] H. Mrabet, "Automatic Layout of Scalable Embedded Field Programmable Gate Array", in *ICEEC International Conference on Electrical Electronic and Computer Engineering*, pp. 469-472, 2004.

[46] Menta Corporation, "eFPGA Core", `http://www.menta.fr/down/DatasheetBrief_eFPGA_Core.pdf`.

[47] Microchip Corporation, "PIC16(L)F1508/9 Data Sheet", `http://ww1.microchip.com/downloads/en/DeviceDoc/41609A.pdf`.

[48] N. Verma and al., "A High-Density 45nm SRAM Using Small-Singal Non-Strobed Regenerative Sensing", in *Proc. ISSCC*, 2008.

[49] N. Mehta and al., "Limit Study of Energy & Delay Benefits of Component-Specific Routing", in *FPGA'12*, 2012.

[50] J. Cong, Y. Ding, "FlowMap : An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", in *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 1-12, 1994.

[51] Altera Corporation, "Quartus II Handbook v11.1.0" `http://www.altera.com/literature/lit-qts.jsp`.

[52] E.M Sentovich and al., "SIS : A System for Sequential Circuit Synthesis", EECS Department University of California Berkeley, *Technical Report No. UCB/ERL M92/41*, 1992.

[53] P. Jamieson and al., "Odin II : An Open-Source Verilog HDL Synthesis Tool for CAD Research", in *IEEE Symp. on Field-Programmable Custom Computing Machines*, pp. 149-156, 2010.

[54] Berkeley Logic Synthesis and Verification Group, "ABC : A System for Sequential Synthesis and Verification", Release 70930.

[55] University of California Berkley, "Berkley Logic Interchange Format (BLIF)", `http://www.cs.uic.edu/~jlillis/courses/cs594/spring05/blif.pdf`.

[56] V. Betz, J. Rose, "VPR : A New Packing, Placement and Routing Tool for FPGA Research", in *International Workshop on Field Programmable Logic and Applications*, pp. 213-222, 1997

[57] D. Bol and al., "A 25MHz $7\mu W/MHz$ Ultra-Low-Voltage Microcontroller SoC in 65nm LP/GP CMOS for Low-Carbon Wireless Sensor Nodes", in *Proc. ISSCC*, 2012.

[58] SiliconBlue Corporation, "ICE65L08 Ultra Low-Power FPGA Known Good Die", `http://www.siliconbluetech.com/media/downloads/iCE65L08DiCEDatasheet.pdf`

[59] CATRENE Working Group on Energy Autonomous Systems, "Energy Autonomous Systems : Future Trends in Devices, Technology, and Systems", `http://www.catrene.org`.