THÈSE

présentée

DEVANT L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

pour obtenir

le grade de : Docteur de L'Université de Bretagne Occidentale

Mention : Informatique

PAR

Alix Lubain Poungou

Équipe d'accueil :	LESTER, FRE CNRS 2734. Université de Bretagne Occidentale
École Doctorale :	SMIS
Composante universitaire :	Sciences et Techniques

TITRE DE LA THÈSE :

Nanotechnologies et Architectures reconfigurables

SOUTENUE LE 26 Avril 2007 devant la Commission d'Examen

COMPOSITION DU JURY :

M.	Lionel	torres	Président
M.	Dominique	lavenier	Rapporteurs
M.	Lionel	torres	
M.	Fabrice	huret	Examinateurs
M.	Bernard	pottier	
M.	Loïc	lagadec	

Remerciements

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Architectures et Systèmes (LESTER FRE 2734), Université de Bretagne Occidentale de Brest. Ce travail s'est inscrit dans le cadre d'une coopération entre l'Université de Bretagne Occidentale et l'Université du Massachusett. Il a été cofinancé par le gouvernement gabonais et le Laboratoire d'Architectures et Systèmes.

Je remercie vivement mon directeur de thèse, Monsieur le professeur Fabrice Huret qui a dirigé ces travaux de recherche.

Je remercie sincèrement Monsieur Lionel TORRES, Professeur à l'Université de Montpellier II (LIRM), qui m'a fait l'honneur d'accepter la présidence de ce jury.

Je remercie vivement Monsieur Dominique LAVENIER, Directeur de recherche CNRS (IRISA, Rennes) et Monsieur Lionel TORRES, Professeur à l'Université de Montpellier II pour l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de ce travail.

Je remercie particulièrment Monsieur Bernard Pottier, directeur de l'équipe d'Architectures et Systèmes, de m'avoir accueilli au sein de son laboratoire et de m'avoir fourni les moyens et permis de mener à bien cette thèse. J'ai beaucoup apprécié la qualité de ses interventions, la pertinence de ses remarques et surtout sa disponibilité et sa gentillesse.

Je tiens à remercier tout particulièrement Monsieur Loïc Lagadec, Maître de Conférences à l'Université de Bretagne Occidentale, pour le soutien qu'il m'a témoigné tout au long de cette thèse placée également sous son encadrement. Je tiens à lui exprimer ma profonde gratitude pour ses conseils avisés et sa rigueur scientifique.

J'adresse mes remerciements à Catherine Dezan, Erwan Fabiani, Christophe Gouyen, Samar Yazdani et tous les autres membres du laboratoire qui ont contribué, par leur bonne humeur et leur aide, à créer un cadre de travail agréable. Je tiens aussi à remercier Damien Picard pour son aide technique.

J'exprime mes pensées chaleureusement à l'ensemble des personnes qui ne sont pas citées par manque de place ou par mégarde et qui ont permis, d'une quelconque manière, l'aboutissement de ce travail.

Je remercie ma famille pour son soutien et pour tout le reste.

Table des matières

Table des figures

Liste des tableaux

Liste des sigles et abréviations

$\mathcal{A}$

ALU Arithmetic Logic Unit

API Application Programming Interface

$\mathcal{B}$

BLIF Berkeley Logic Interchange Format

$\mathcal{C}$

CAO Conception Assistée par Ordinateur

CLB Cell Logic Block

CMOS Complementary Metal Oxide Semiconductor

CNT Carbon NanoTubes

CNTFET Carbon NanoTube FET

$\mathcal{D}$

DEC Décodeur

DPGA Dynamically Programmable Gate Array

$\mathcal{F}$

FET Field Effect Transistor : NFET (ou Nfet) et PFET (ou Pfet)

FPGA Field Programmable Gate Arrays

$\mathcal{I}$

ITRS International Technology Roadmap for Semiconductors

$\mathcal{L}$

LUT Look Up Table

$\mathcal{M}$

Madeo Outil de CAO : Madeo-Bet (couche de synthèse physique) et Madeo-Fet (couche de synthèse applicative)

MADMACS Un outil de placement et routage pour dessiner les masques de réseaux réguliers

MIMD Multiple Instruction Multiple Data

MLA Molecular Logic Array

MOSFET Metal-Oxide-Semiconductor FET

MPMD Multiple Program Multiple Data

MV Majority Voter

MWCNT Multiple-Walled Carbon Nanotube

$\mathcal{N}$

NASIC Nanoscale Application Specific Integrated Circuit

NFPGA Nanoscale FPGA

$\mathcal{P}$

PC Compteur de Programme(Counter Program)

PIP Programmable Interconnect Point

PLA Programmable Logic Array

PROM Programmable Read Only Memory

$\mathcal{Q}$

QCA Quantum Cellular Automata

$\mathcal{R}$

RAM Read Acces Memory

ROM Read Only Memory

$\mathcal{S}$

SET Single Electron Transistor

SIMD Single Instruction Multiple Data

SIS Sequential Interactive Synthesis [84] : outil pour synthétiser et optimiser les circuits séquentiels

SPM Scanning Probe Microscopes

SPMD Single Program Multiple Data

SWCNT Single-Walled Carbon Nanotube

$\mathcal{V}$

VPR Versatile Place and Route [5] : environnement de programmation des FPGAs

$\mathcal{W}$

WISP WIre Streaming Processor

$\mathcal{X}$

XC6200 La famille XC6200 de FPGA de Xilinx

Introduction générale

Problématique

L'exploration submicronique est une alternative envisagée pour faire évoluer les performances des machines. Les performances liées à cette échelle permettent d'espérer repousser les limites de la loi de Moore qui stipule que les performances des calculateurs doublent tous les 18 mois. La réduction de la taille des dispositifs peut continuer à préserver cette loi empirique. En effet, la taille des dispositifs (transistors) détermine le dégré d'intégration et la vitesse de transport des électrons. Cette course à la miniaturisation entraîne des limitations pour les dispositifs classiques. Plusieurs obstacles pourraient s'opposer à celle-ci, notamment les phénomènes de la physique quantique et l'augmentation de la dissipation thermique dans les contacts. Cette course favorise en même temps l'émergence de nouvelles voies de recherche, axées sur les nanotechnologies. Par définition, les nanotechnologies désignent un spectre de domaines manipulant les objets d'une taille de l'ordre du nanomètre. A taille constante, les performances des calculateurs futurs pourraient être plus élevées d'un facteur 10^9 [54]. Une nouvelle gamme des dispositifs nanoscopiques apparait ; il faut désormais les intégrer pour construire des architectures de traitement. Cette évolution technologique ouvre des perspectives prometteuses dans le domaine de traitement de l'information.

Pour suivre cette évolution technologique, les architectures de traitement doivent s'appuyer sur des supports technologiques à base de nano-objets. Cela suppose une chaîne de conception différente, qui imposerait de définir des nouvelles options dans la structuration de ces architectures et qui permettrait d'exploiter le potentiel de l'infiniment petit. Richard Feynman a écrit ≪*There is Plenty of Room at the Bottom*≫ [30]. Mais l'exploitation de cet espace soulève de nombreuses questions : comment profiter de cet espace pour construire les nanostructures nécessaires à la réalisation des applications ? Comment connecter ces nanostructures entre elles ainsi qu'aux fils électriques pour former un circuit ? Passer des composants à l'échelle nanométrique, à un circuit fonctionnel est non trivial ; transposer l'ingénierie classique (CMOS) aux nanotechnologies ne suffit pas : il faut proposer de nouvelles architectures de traitement.

Les nouvelles applications demandent l'élaboration de nouvelles architectures de calcul. Face à la complexité de ces applications, le reconfigurable est souvent une option alternative pour prototyper les architectures.

Pour explorer les nouvelles architectures, nous pouvons nous inspirer des modèles existants dans le domaine du reconfigurable. Au contact de nanotechnologies, le reconfigurable

peut apporter une réponse à certaines contraintes liées à l'échelle nanoscopique des nano-objets à incorporer dans les architectures. En effet, les architectures reconfigurables pour les technologies émergentes doivent posséder une structure très régulière (les nanostructures denses et régulières peuvent être programmées pour mettre en œuvre les circuits complexes) et semblent tolérer les défauts. Dans ce cas, la reconfiguration représente une approche évidente pour vérifier les défauts de fabrication et les éviter éventuellement [8, 44].

Contexte de l'étude

Les concepteurs de circuits intégrés utilisent des outils avancés de CAO pour concevoir, tester et réaliser les produits manufacturables. D'une part, les outils de CAO physique permettent de générer le dessin d'un circuit intégré à partir d'une description de sa fonction. D'autre part, les outils de CAO applicative permettent de coupler une application et un dessin pour fournir une application implémentée. Mais ces outils de synthèse n'existent pas pour les supports nanotechnologiques. De plus, les outils classiques n'arrivent plus à suivre les nouvelles orientations technologiques : les nouveaux dispositifs imposent des nouvelles contraintes à prendre en compte dans ces outils.

Avec l'évolution technologique, la couche basse permettant de combiner les opérateurs et la couche architecturale doivent changer. Nous avons donc besoin d'outils pour ramener le niveau d'abstraction des dispositifs. Les outils peuvent permettre de restructurer les couches technologiques pour maîtriser la complexité de traitement lors de la synthèse (physique, architecturale) d'autant plus que l'augmentation de la capacité d'intégration accroît la complexité d'intégration. Les outils doivent arriver à contrôler cette complexité d'intégration.

Les méthodologies exprimées dans les outils sous forme de règles (ou contraintes) reposent sur la technologie CMOS. Prévoir les performances des technologies futures de nature différente de la technologie CMOS en utilisant les outils actuels peut donner lieu à un constat mitigé de retour d'information. Une telle situation conduit à envisager deux alternatives pour la mise en œuvre des outils d'exploitation des nanotechnologies. La première option consiste à refaire les outils propres aux concepts de chaque technologie. Cette option rendrait obsolète les outils existants et son coût serait exorbitant vue la diversité des technologies. La seconde solution consiste à faire évoluer les outils existants par rapport aux nouveaux concepts technologiques. Cette dernière solution a pour but d'intégrer dans les outils existants des nouvelles méthodologies qui rendent compte des contraintes spatiales, fonctionnelles et structurelles des nouvelles technologies (architectures et dispositifs).

A titre d'exemple, un processeur " stream " [68] est développé sur des supports nanotechnologique basés sur un nouveau modèle de conception. Le modèle de conception basé sur le PLA impose la présence d'un signal et son complémentaire à l'entrée et à la sortie d'un circuit. Le routage interne dans un circuit est assuré par les nanodispositifs (nanofils et aiguilleurs moléculaires). Cet aspect de routage implique de différencier les dispositifs de routage de ceux de la logique dans le circuit. Par conséquent, les outils

pour un tel modèle de conception n'existent pas, encore moins pour vérifier une large conception du processeur dont la version initiale supporte un jeu d'instructions réduit et contient deux registres de 2 bits. On est bien loin de ce qui se fait aujourd'hui avec la technologie classique. La conception de circuits intégrés basée sur les nanodispositifs reste particulièrement difficile du fait d'un manque d'outils.

Madeo (outils de CAO) peut subir une adaptation pour formaliser de nouveaux modèles de conception spécifiques à une technologie. Une transposition vers les architectures pour les nanotechnologies des modèles et des méthodologies qui ont fait le succès des architectures reconfigurables (validation de l'outil sur plusieurs architectures commerciales : XC6200, Virtex, Atmel,...) doit être effectuée. Différents points fondamentaux doivent être abordés, en particulier l'automatisation du tracé des supports architecturaux (Génération, Routage, Placement, ...). La nature configurable des nanodispositifs actifs impose des étapes supplémentaires dans la mise en œuvre pour obtenir des supports technologiques reconfigurables.

Cette conception assistée représente une étape importante pour réaliser les circuits et les systèmes intégrant les nano-objets.

Contribution de cette thèse

La prochaine révolution industrielle repose potentiellement sur l'exploitation des nanotechnologies. Les débats se développent autour des outils de programmation, d'implémentation et d'exploitation des architectures de calcul pour ces technologies. Ces travaux contribuent à l'évolution technologique dans les outils. Dans ce cadre, divers axes d'étude sont examinés : mesurer l'impact de ces technologies sur les outils existants au moyen des architectures proposées pour ces dernières, modéliser les supports technologiques, examiner les règles de dessin dictées par les nouveaux aspects technologiques. La spécificité de ces travaux réside dans le niveau de détails de description d'une architecture. Les mécanismes et les aspects spécifiques à une technologie donnée sont décrits à un niveau proche du tracé physique.

Les travaux présentés proposent une définition d'architectures reconfigurables basée sur des supports nanotechnologiques configurables sur un mode PROM. La problématique centrale s'articule autour de trois axes : les règles de dessin, le calcul de coûts et le balayage d'un espace d'architectures. Les règles de dessin reposent sur la nature configurable des dispositifs actifs et sur la conception technologique. Le calcul du coût en surface permet d'avoir un large choix sur la représentation analytique des modèles des architectures reconfigurables. Cette option se base sur une paramétrisation, un choix du modèle et un nombre de contextes de l'architecture.

La démarche finale est de définir conjointement une architecture pour une nanotechnologie et son environnement de programmation, d'implémentation et d'exploitation.

Plan du mémoire

Le travail présenté dans ce mémoire s'articule comme suit :

Le chapitre 1

Le chapitre donne un aperçu du domaine du reconfigurable, des architectures aux outils de programmation : architectures reconfigurables de type FPGAs et ses constituants de base.

Il présente également une chaîne de CAO pour les architectures reconfigurables. C'est un environnement de modélisation et de programmation qui fournit un ensemble d'outils génériques pour manipuler les architectures. Cet environnement peut se composer en deux couches : une couche basse pour la mise en œuvre et le développement des architectures reconfigurables et une couche haute pour la description des applications à implémenter sur ces architectures. L'environnement est ouvert à d'autres types d'architectures.

Le chapitre 2

Le chapitre fournit un inventaire des technologies émergentes, au travers de leurs caractéristiques et de leur usage potentiel. Il présente aussi quelques pistes de techniques de fabrication pour ces technologies nanoscopiques. Des architectures pour ces technologies sont également introduites. Elles possèdent une structure calquée sur celle des architectures classiques avec une reconsidération d'échelle due à la nature nanoscopique des dispositifs de base. Le chapitre se termine sur une conception des supports de nanofabrique basés sur une technologie spécifique, choisie pour la suite de l'investigation.

Le chapitre 3

Le chapitre introduit un environnement à même d'adresser les architectures prospectives, en particulier les architectures à base de nanofabriques. Cet environnement est obtenu sur la base de Madeo, à travers une extension des outils existants. La validation de l'extension s'appuie sur un test de mise en œuvre automatique d'une architecture applicative, réalisée sur les supports de nanofabriques. Un examen de différents types de tracés topologiques permet de garantir l'efficacité des résultats obtenus. Ces résultats montrent par ailleurs la fiabilité d'une démarche accompagnée de calcul d'impacts.

La singularité de l'architecture applicative repose sur des supports de mise en œuvre configurables.

Le chapitre 4

Le chapitre examine la possibilité de conception des architectures reconfigurable avec des supports configurables dans une technologie émergente. Une architecture de FPGA connue sert de référence. Elle est reformulée pour respecter les règles de dessin imposées par la dite technologie. Les briques de base composant la cellule d'une telle architecture sont conçues dans cette technologie. Ces éléments sont alors combinés pour former une tuile qui représente une cellule entière de l'architecture. Un facteur de performances est introduit. Il s'appuie sur l'évaluation du coût spatial d'une implémentation. Une prospection

permet d'envisager, au moyen de paramètres variables, différents modèles représentatifs de l'architecture suivant une application donnée.

Les étapes pour la production automatique de l'architecture sont expliquées. Elles s'appuient sur l'adaptation de l'environnement Madeo. Une implémentation automatique des composants de la cellule reformulée est montrée. La dernière section du chapitre aborde le multicontexte dans les architectures. Les implémentions multicontextes d'une architecture sont proposées et leurs coûts en surface sont caractérisés.

Conclusion

Elle adresse un bilan de la contribution qui couvre une étude bibliographe axée sur les nanotechnologies et les pistes architecturales, une synthèse de supports nanotechnologiques et une définition d'architectures reconfigurables. Elle souligne aussi les perspectives qui s'articulent autour de la finalisation d'un démonstrateur architectural et de l'intégration des supports nanotechnologiques dans un SOC.

Chapitre 1

Madeo et la synthèse portable pour les architectures reconfigurables

1.1 Architectures reconfigurables : FPGAs

Les technologies reconfigurables, depuis l'apparition des FPGAs (Field Programmable Gate Arrays) au milieu des années 1980, sont devenues incontournables. Elles proposent une alternative aux architectures classiques en permettant une optimisation post fabrication pour s'adapter au mieux aux besoins des applications (à implémenter). Cette grande souplesse des solutions programmables engendre un coût non nul, puisque les architectures reconfigurables souffrent d'un fort déficit en terme de densité par rapport aux architectures spécifiques, ce qui a longtemps limité leur usage à des fins de prototypage, de glue logique ou à la production des petites séries. En revanche, dans un cadre de système sur puce, l'intégration de portions reconfigurables (FPGAs) trouve tout son sens pour peu qu'une vision système permette une pleine exploitation des ressources hétérogènes.

Les architectures reconfigurables sont la cible de nombreux travaux (dans la communauté scientifique). Il s'avère donc essentiel de commencer par un chapitre qui définit clairement une architecture de FPGA et les différentes ressources reconfigurables contenues dans cette dernière.

1.1.1 Structures de FPGAs

Les FPGAs constituent un type de circuit appartenant à la famille des circuits programmables. Leur architecture est composée des cellules logiques identiques (CLBs), des connexions entre les cellules et des blocs d'aiguilleurs. La figure 1.1 illustre une vue schématique et simplifiée de l'architecture. Le réseau d'interconnexion entourant les cellules logiques est constitué d'un ensemble de lignes connectées par des points d'interconnexion programmables. Ces lignes bénéficient de deux types de connexions : les lignes connectées directement aux cellules logiques et les lignes connectées entre elles au travers du réseau d'interconnexion.

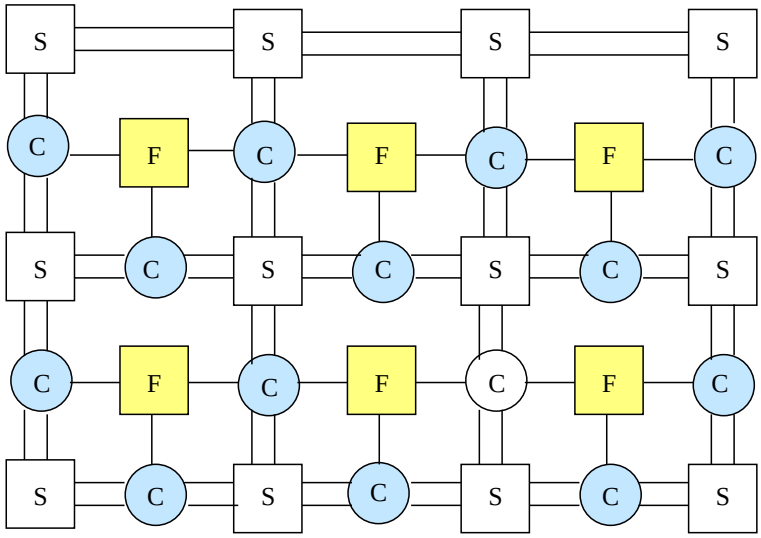


FIG. 1.1 – Vue simplifiée d'un circuit programmable de type FPGA.

Le réseau d'interconnexion garantie l'aiguillage des signaux provenant des différentes ressources de routage à l'aide d'un système d'interconnexions programmables.

Les interconnexions liées à la cellule logique permettent de joindre les entrées du bloc logique avec les lignes d'interconnexions couvrant le FPGA.

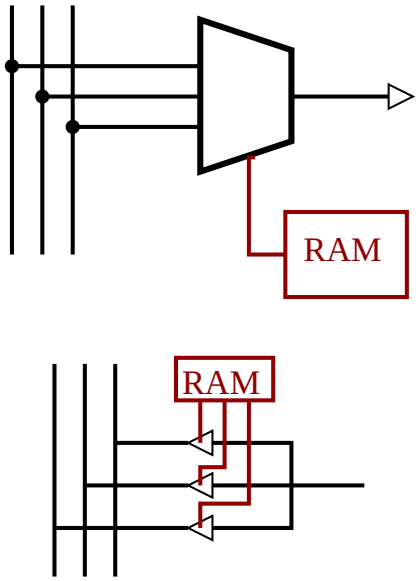
Les aiguilleurs (S) sont connectés entre eux pour permettre de relier les cellules logiques sur les longues distances au travers du FPGA. Ils assurent ainsi la connexion entre les interconnexions verticales et horizontales. Toutes les lignes de connexions situées aux quatre points cardinaux sont configurables au moyen d'une mémoire RAM. La figure 1.2(b) montre les différents bits de configuration entre les ressources de routage disponibles dans un bloc d'aiguilleurs.

Les cellules logiques constituent les éléments de base. Ce sont des blocs logiques configurables. La figure 1.3 montre trois cas de CLBs du plus simple au plus compliqué. La représentation schématique dans la figure 1.3(a) symbolise une LUT qui spécifie la logique combinatoire. La LUT peut être configurée comme un élément de la mémoire ou comme un registre à décalage.

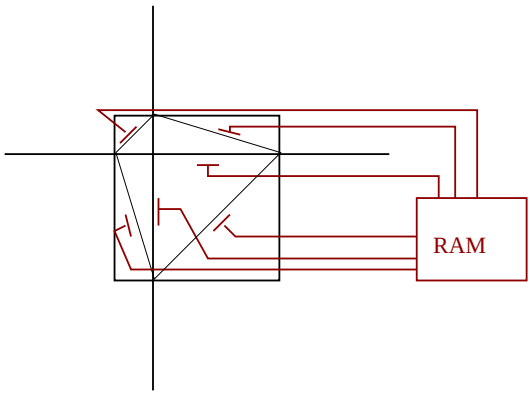
1.1.2 Quelques FPGAs

Afin d'avoir une idée des classes des FPGAs, nous présentons deux d'entre elles. Cette introduction permet de comprendre aisément la réalisation d'une architecture à travers les générations et d'avoir une idée de la structuration des éléments. De plus, une connaissance des briques de base utilisées est alors indispensable pour comprendre les choix architecturaux liés à chaque famille de circuits FPGA.

FPGA en îlots de calcul C'est une architecture dont les éléments de base sont contenus dans une matrice plane. Ces éléments représentent les ressources logiques, un



(a) Blocs de connexions programmables.



(b) Aiguilleurs programmables

FIG. 1.2 – Nature programmable des connexions.

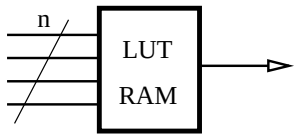
réseau d'aiguillage et les ressources de routages programmables du FPGA. La figure 1.2 montre les deux types de blocs de connexions programmables. De même, l'illustration représentée sur la figure 1.2(a) met en valeur les possibilités de connexions qu'offre un bloc de connexion grâce à sa programmation.

Un bloc logique est composé de tables d'allocation (LUTs), de bascules et de multiplexeurs. Les LUTs contiennent la table de vérité d'une fonction logique ou un ensemble des valeurs stockées grâce leur configuration. Les bascules sont des éléments de mémorisation. Elles permettent de construire les blocs mémoires qui sont configurables selon la taille de mots stockés.

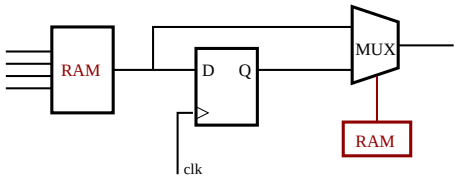
Famille XC6200 de Xilinx (mer de portes). C'est une architecture composée des cellules simplement configurables. La structure est simple, hiérarchique, symétriques et régulière. Chacune des cellules est individuellement programmable et permet d'implémenter une fonction logique.

Les cellules voisines sont regroupées en blocs de 4 par 4 cellules. Ces blocs sont également regroupés en blocs de 4 par 4 pour former des blocs de 256 cellules. La même organisation est répliquée à tous les niveaux de l'architecture.

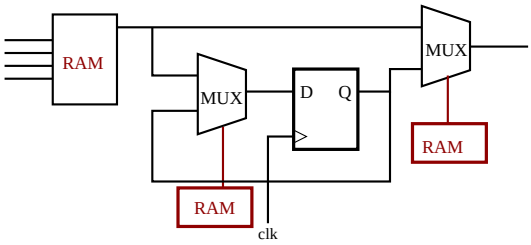
Dans cette famille, les fonctions logiques réalisables sont des simples LUTs. L'unité fonctionnelle se compose essentiellement de multiplexeurs. La figure 1.4(a) symbolise une n-LUT (n est le nombre des entrées). La figure présente des multiplexeurs $4 \rightarrow 1$ utilisés dans chaque cellule logique. Le multiplexeur isolé des autres, contrôlé par les bits de configuration sert à router les signaux vers l'unité fonctionnelle. Les 4 autres multiplexeurs permettent de router les signaux vers l'extérieur de la cellule. La structure du routage est



(a) Unité fonctionnelle réduite à 1 LUT.

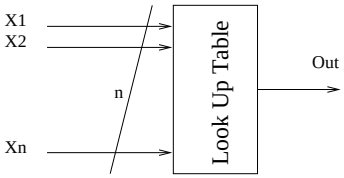


(b) Unité fonctionnelle avec un flip-flop.

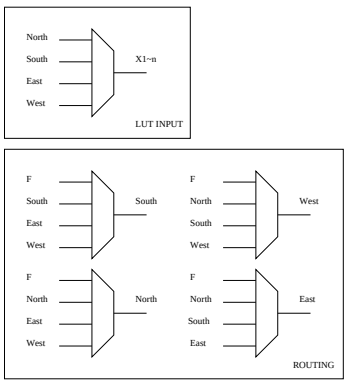


(c) Unité fonctionnelle avec un flip-flop réentrant.

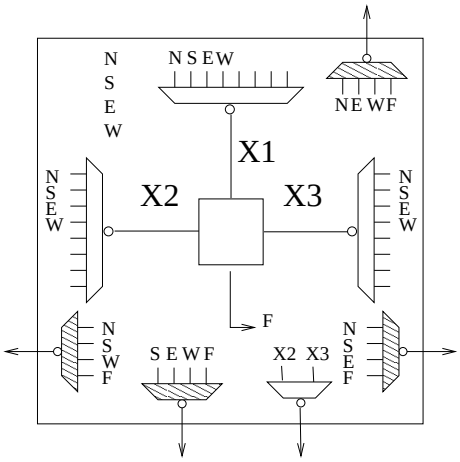
FIG. 1.3 – Les unités fonctionnelles programmables.



(a) Vue schématique d'une n-LUT.



(b) Les multiplexeurs de routage d'une cellule de XC6200.



(c) Une cellule de base XC6200.

FIG. 1.4 – Les constituants de la cellule XC6200.

orientée et logarithmique.

La figure 1.5(a) symbolise les directions possibles de connexion dans une cellule et avec les cellules adjacentes. Les cellules sont munies de ressources de routage de même longueur dans les quatre directions. Une cellule utilisée pour le routage demeure exploitable pour réaliser la logique. La sortie de la LUT est connectée au bloc de routage pour être dirigée dans une direction précise selon la configuration du bloc. La figure 1.5(b) indique la mise en œuvre des ressources dans la cellule de XC6200. Une fois le réseau d'interconnexion configuré, le décodage des signaux de la LUT est effectué. La LUT génère ainsi un signal qui est orienté dans une direction grâce une configuration du réseau d'interconnexion.

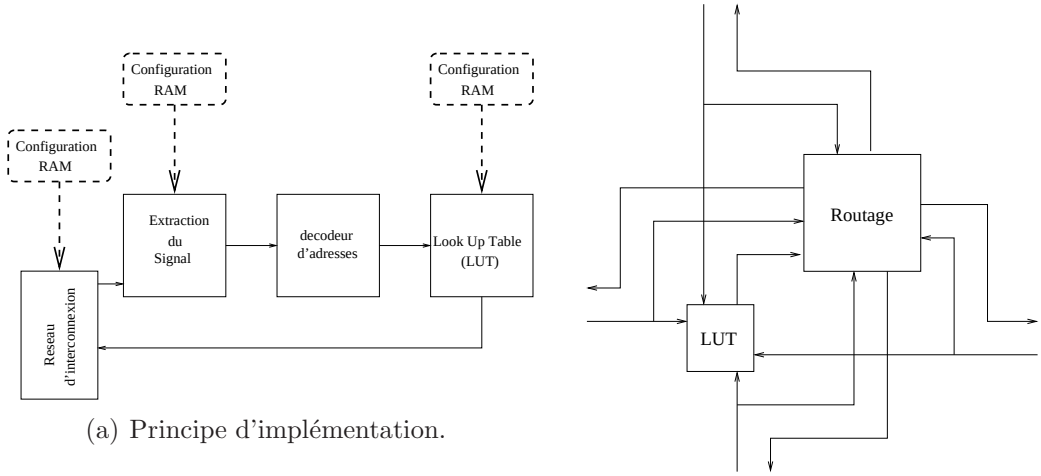
La nature structurelle de la famille XC6200 offre la possibilité de faire varier plusieurs paramètres comme le nombre des entrées/sorties et la largeur des canaux de routage qui peuvent influencer sur la qualité des performances. En effet, les blocs de n par n cellules

contiennent des lignes de longueur n qui autorisent la traversée des blocs sans consommer les ressources de niveaux inférieurs.

1.1.3 Conclusion

La définition d'une architecture implique l'étude des entités qui la composent. Ces entités programmables et leur structuration, préférée en matrice, ont été présentées. Deux types d'architectures ont été montrés, ainsi que les liens qui les unissent aux briques de base.

Dans la section 1.2.2, nous présentons un environnement de CAO qui modélise les architectures reconfigurables. Il permet une prospection architecturale des supports d'exécution (FPGAs).



(a) Principe d'implémentation.

(b) Vue Schématique des ressources.

FIG. 1.5 – Directions des ressources dans une cellule de XC6200.

1.2 Présentation de Madeo

1.2.1 Environnement portable pour FPGAs

Les environnements de programmation constituent aujourd'hui un verrou à l'exploitation efficace des technologies reconfigurables. Les outils doivent être produits de plus en plus rapidement et intégrer de nouvelles fonctionnalités (reconfiguration partielle, etc.) [6]. La production d'outils ne doit pas simplement accompagner les évolutions matérielles, mais doit les devancer afin d'orienter les choix des architectes lors de la conception d'une nouvelle architecture. Seul l'environnement logiciel de manipulation de l'architecture (placeur-routeur, etc.) fournit des critères quantitatifs. L'expérience du projet PROTEUS [28], durant lequel les phases de développement matériel et logiciel ont été menées conjointement, illustre le besoin d'outils stables pour réaliser une prospection architecturale [87].

La disponibilité d'une base algorithmique commune autorisant la comparaison objective des architectures constitue une contrainte supplémentaire. Les architectures doivent être manipulées par les mêmes outils, ce qui impose soit de réimplémenter les algorithmes pour chaque architecture soit de disjoindre les algorithmes de l'architecture. La seconde solution offre, outre une réduction des coûts de développement logiciel, l'assurance d'obtenir un code facilement maintenable. Chaque évolution profite à l'ensemble des architectures manipulables. Une auto-adaptation de ces outils à des architectures dégradées constitue également un enjeu important. En effet, dans un cadre nanotechnologique le zéro défaut ne peut pas être garanti ; chaque exemplaire du matériel diffère de son modèle "idéal" par des défauts détectables après fabrication. L'exploitation de ces architectures requiert une prise en considération automatique des défauts de l'architecture par les outils.

1.2.2 Madeo

Dans sa thèse, Loïc Lagadec décrit Madeo [55], un cadre générique de développement pour modéliser et programmer les architectures reconfigurables, qui adresse les points précédents. Les objectifs visés sont de favoriser la réutilisation des outils et des applications (portabilité), et de fournir des outils adaptés pour la mise en œuvre de nouvelles architectures. Ils se décomposent en trois parties :

1. une couche basse (Madeo-Bet) :

   Elle regroupe un ensemble d'outils de bas niveau pour manipuler les circuits reconfigurables. La chaîne de synthèse est générique. Elle comporte un modèle abstrait d'architecture servant de canevas pour instancier les architectures concrètes décrites grammaticalement. Cette couche garantie un jeu d'outils adapté à une technologie ou à une architecture cible.

2. une couche intermédiaire (Madeo-Fet) :

   Elle assure la synthèse des applications vers de la logique utilisable par les outils de bas niveau. Un compilateur permet à partir d'un graphe de flot de données, d'une inférence de types et d'optimisations, de produire de la logique de BLIF en se servant du code *Smalltalk* de haut niveau.

3. une couche système :

   C'est une couche supérieure qui s'appuie sur le compilateur pour gérer le système synthétisable sur FPGA. Dans cette couche, l'architecture de calcul est décrite dans ses aspects statiques et dynamiques.

1.2.3 Modélisation des architectures dans Madeo

Un modèle est une structure de données partagée par les différents outils. Chaque élément du modèle décrit son comportement : un modèle est une structure de données active. Le comportement régit l'accès aux données et leur confère une sémantique. La figure 1.6 la communication entre les outils à travers un échange de données.

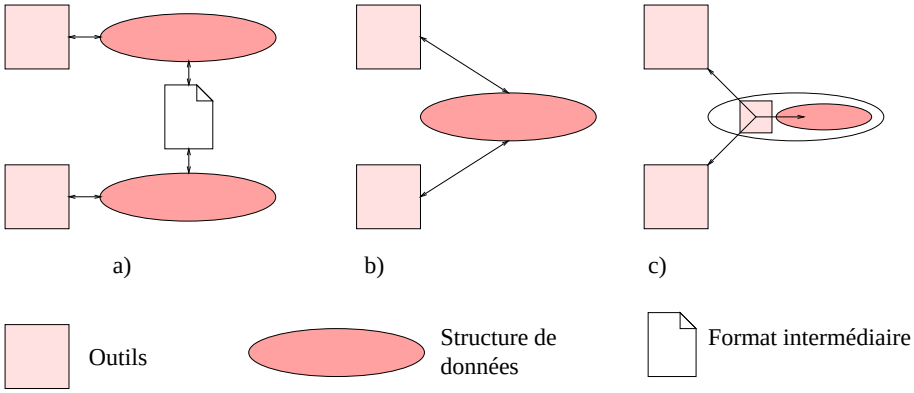


FIG. 1.6 – Trois types d'organisation de données dans une chaîne logicielle. a) Structure de données commune aux outils ; chaque outil définit ces opérations. b) Structure de données partagée. c) Structure de données active.

Chacune des notions explicitées lors de la phase d'abstraction trouve un écho dans le modèle d'architecture reconfigurable. Cette modélisation est constituée d'un ensemble de classes Smalltalk-80, correspondant à ces notions. La table 1.1 illustre la structuration hiérarchique des classes du modèle et leur composition.

Une modélisation des architectures reconfigurables est proposée : elle repose sur l'abstraction des caractéristiques représentatives des principales architectures existantes. De l'analyse comparative précédente, il ressort que les architectures étudiées peuvent être représentées de manière uniforme. La représentation intègre des éléments atomiques et des mécanismes de structuration.

1.2.3.1 Eléments atomiques

Les éléments atomiques peuvent se regrouper au sein de trois catégories principales : les ressources de calcul, les ressources d'interconnexion et les ressources de stockage. Les ressources de calcul sont définies par leur nombre d'entrées et de sorties et leur partie opératoire. La partie opératoire peut être une LUT réalisant toute fonction de k entrées, ou une fonction parmi un ensemble de fonctions possibles. Les ressources de routage interconnectent les ressources de calcul ou de stockage. Elles sont spécifiées par leur nature directionnelle ou non, leur segmentation, leur population et la largeur du canal de routage. La segmentation décrit la portée des différentes ressources. La population correspond à l'énumération des points connectables parmi ceux visités. Les tampons sont placés aux entrées/sorties des fonctions ou dans le réseau de ressources de routage. Ils sont utiles pour stocker des données et réduire le temps de cycle d'un circuit.

1.2.3.2 Eléments composites

Les architectures peuvent être structurées de manière hiérarchique. Cette structuration simplifie la description architecturale par composition et réplication régulière d'éléments.

La prise en considération de la nature hiérarchique des architectures constitue un gage d'extensibilité de la représentation [80] et doit bénéficier aux outils, en particulier pour la description d'architecture, le partitionnement [51] et l'obtention de circuits réguliers.

1.2.4 Description des architectures

1.2.5 Introduction

Le modèle d'architecture décrit l'ensemble des ressources modélisable au sein d'une architecture reconfigurable. Ce modèle est qualifié de méta modèle *abstrait*.

Il sert de patron de conception pour le modèle représentant une architecture. Ce dernier sert à son tour à produire des exemplaires de l'architecture, manipulables par les outils présentés dans la section 1.3.

La figure 1.7 illustre cette chaîne de conception, en y introduisant la notion de description grammaticale.

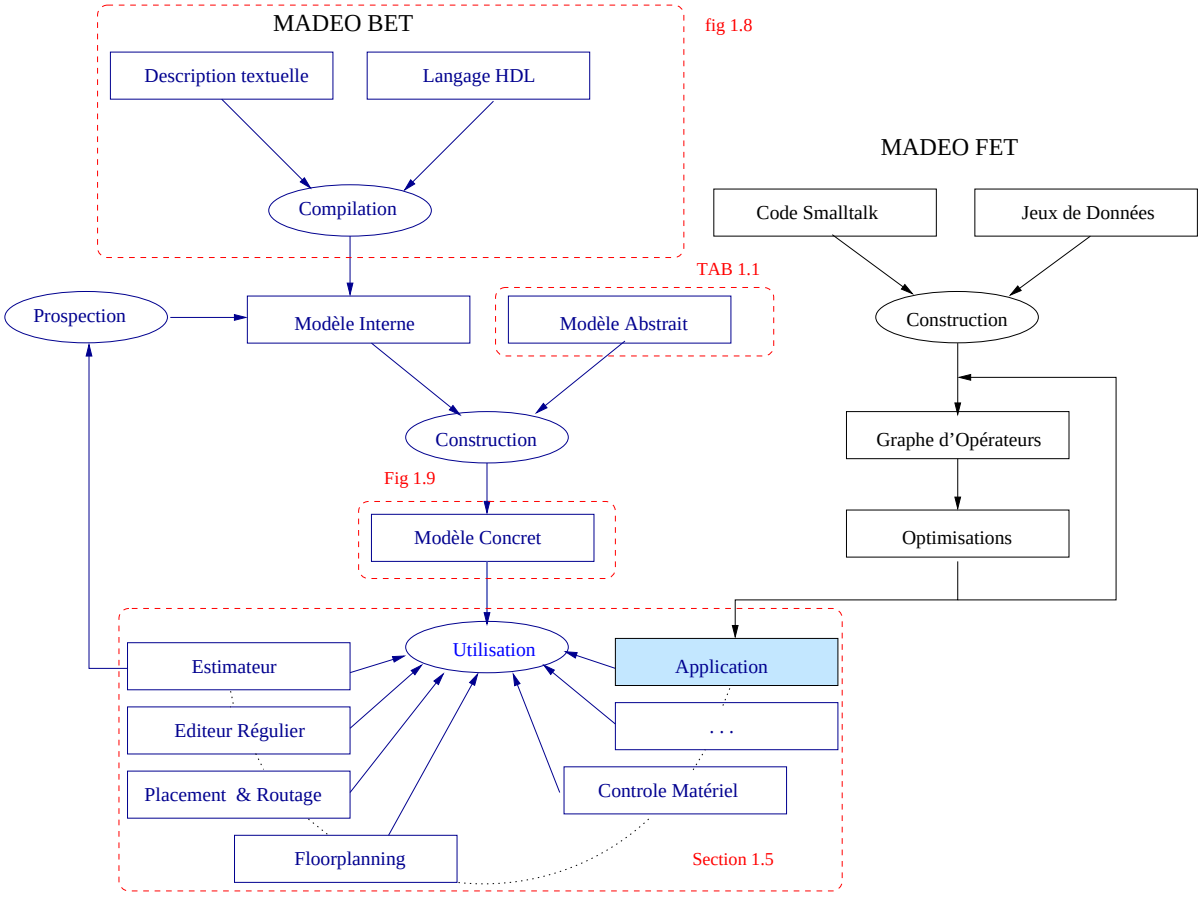


FIG. 1.7 – Production de modèles d'architecture sous Madeo. A gauche la chaîne de description de l'architecture et à droite celle de l'application.

Un modèle *concret* peut être obtenu par compilation d'une description textuelle. La phase de compilation génère un modèle technique qui produit un modèle *concret*, en sousclassant et en instanciant les classes du modèle *abstrait*.

Chaque élément de l'architecture est décrit : soit comme une référence à une classe déjà existante, soit de façon à produire une nouvelle classe.

La description de l'architecture inclut l'énumération des fonctions réalisables par chaque nœud de calcul. Cette caractéristique est particulièrement importante lors de la modélisation d'architectures dont le jeu de fonctions logiques est réduit, par exemple les architectures basse consommation. La description des fonctions réalisables est exprimée dans le langage Genlib, utilisé en entrée de SIS [84] ; la même description peut donc s'appliquer aux phases de synthèse logique et de définition d'architecture. Un objectif est d'automatiser la phase de conversion technologique sous le contrôle de l'architecture.

La description d'une architecture concrète est un mécanisme rapide. Les outils s'appliquant sans modification à toute architecture concrète, l'obtention d'outils opérationnels pour une architecture est uniquement conditionnée par la description de cette architecture. En particulier, l'éditeur de circuits réguliers instancie pour toute architecture les éléments de la bibliothèque de méta-circuits. Un méta-circuit est une description structurée de circuits indépendants de la technologie. Il est alors possible de bénéficier immédiatement de bibliothèques d'opérateurs arithmétiques sans effort.

Mise en œuvre : atomiques Les ressources de calcul sont spécifiées par l'énumération de leurs entrées et de leurs sorties. La partie opératoire est typiquement une table de vérité (LUT) mais peut correspondre à une série de fonctions réalisables.

Les ressources d'interconnexion sont décrites par l'énumération des signaux possibles en entrée puis des points atteignables (segmentation et population des segments). A ces caractéristiques sont associées des valeurs de coût servant au calcul des fonctions d'estimation de la qualité d'une solution.

Les ressources de stockage apparaissent comme des FIFOs de bascules ; elles permettent de reséquencer les circuits en pipelinant les calculs et les ressources de routage, et de stocker des valeurs internes au circuit.

Mise en œuvre : composites Les éléments composites sont de deux natures : pavage ou agglomérat. La notion de pavage introduit une information géométrique. Les pavages peuvent comporter des ressources hétérogènes. La notion d'agglomérat assure une cohésion hiérarchique du modèle. Chaque objet référence son conteneur. Chaque conteneur référence les objets qu'il contient. Si ces objets sont munis de coordonnées géométriques, le conteneur est un pavage. Si les objets sont référencés de manière symbolique, le conteneur est un agglomérat.

Méta modèle Le modèle est constitué d'un ensemble de classes, énumérées dans la table 1.1. Chaque élément est muni d'une structure interne définie par ses variables d'instance. La structure interne est conservée par héritage (entre parenthèse).

Chaque type d'élément est également décrit par son comportement. Le comportement est transmis par héritage, mais se voit généralement étendu ou modifié (surcharge de méthode).

Techniquement, les classes du modèles sont toutes chapeautées par une classe abstraite **RaObject**.

1.2.6 Extensibilité de la modélisation

L'extension de la modélisation est réalisée dans deux directions. La première est quantitative et consiste à augmenter le nombre de classes concrètes, c'est-à-dire à accroître le volume de la bibliothèque d'éléments d'architecture disponibles. La seconde direction est qualitative et consiste à définir de nouveaux éléments de l'architecture abstraite. Cette extension permet de bénéficier des nouveaux motifs prospectifs (supports architecturaux) pour accompagner et suivre l'évolution technologique. Parmi les ajouts envisagés, notons les opérateurs *nbits*, les portions câblées de circuits, les mémoires, les capteurs, etc. L'extension de l'architecture abstraite intervient sans perturber les outils sous réserve du respect de l'interface logicielle commune. De façon similaire, le jeu d'outils peut être étendu pour répondre à de nouveaux besoins. Des outils spécifiques peuvent également être substitués aux outils génériques pour une architecture donnée.

1.3 Outils de synthèse physique

La manipulation des architectures reconfigurables requiert des outils logiciels. Ces outils se situent à différents niveaux. L'objectif étant d'appliquer l'ensemble des algorithmes à toute architecture, ceux-ci sont génériques.

Placeur routeur : le placeur routeur convertit une description de niveau transfert de registres (RTL) en une configuration de l'architecture reconfigurable. La logique manipulée est aléatoire ; la qualité de la configuration produite est quantifiée au moyen de divers estimateurs. L'algorithme de placement routage employé est de type PATHFINDER[69] et repose sur une négociation entre les différents signaux à router lors d'un conflit sur une ressource. Le routage point-à-point d'un signal repose sur l'algorithme de routage en labyrinthe présenté par Lee [60].

Editeur régulier : l'éditeur régulier assure la réplication de façon régulière des motifs de configuration (modules) dans l'architecture. Il est possible de la sorte de réaliser des circuits systoliques ou des opérateurs arithmétiques. La composition des motifs peut être paramétrée par la taille des modules de façon à autoriser des adressages relatifs. La production de circuits réguliers réduit le temps de synthèse par factorisation de la phase de placement-routage et génère des instances identiques de modules. Cette identité assure une homogénéité des performances des sous-circutis.

Floorplanner : le floorplanner applique des critères d'optimisation sur l'agencement de modules qu'il produit grâce à l'éditeur régulier. Il opère à l'aide d'un recuit simulé, mais pourrait bénéficier d'algorithmes déterministes.

RaObject (container position selected additionalParameters)
- RaTriState (input output command)
- RaNilObject
- RaFunctionDescription (inputs outputs list)
- RaConnect (pins)
  - RaProgrammableConnect (c r)
    - RaBuffer (input output)
      - RaInvertBuffer
    - RaPassTransistor (currentState)
    - RaPip (resources)
    - RaSwitch (x y width)
      - RaConcreteSwitch (hResources vResources pips)
      - RaFunctionalSwitch (resources east west north south)
  - RaHardConnection
- RaNode (view size baseCost occupancy presentCongestion historicalCongestion pFac)
  - RaNodeWithoutNammedPins
    - RaWire (pins expanded)
      - RaOutputPin
      - RaInputPin
      - RaPin
    - RaExpandedWire (tracks)
      - RaExpandedPin
  - RaNodeWithNammedPins
    - RaMemory (adress data rw memory)
    - RaMultiplexer (inputs output inputsName outputName selection)
    - RaFunction (inputs outputs function possibleFunctions)
    - RaRegister (input output inputName outputName clock type)
    - RaMultipleSelection (inputs output)
    - RaMultipleOutputs (input outputs)
- RaCompositeObject (interface objects view)
  - RaArrayedObject (x y functions functionsPosition)

Tab. 1.1 – Hiérarchie de classes du méta modèle abstrait

Estimateurs : les estimateurs s'appuient sur les facteurs de coût des différents éléments de l'architecture pour calculer des fonctions de coût global. Ces fonctions évaluent la qualité du circuit produit suivant divers critères : surface englobante, coût du routage, chemin critique, etc.,

Contrôle matériel : sous réserve de disposer d'une définition publique de l'architecture ou d'une interface de programmation (API) publique sur laquelle on peut s'appuyer, le contrôle du matériel peut être réalisé. Deux démonstrateurs existent : un pour l'architecture XC6216 et un second pour les architectures Virtex.

Interface utilisateur : une interface utilisateur interactive offre au programmeur une représentation visuelle, et l'autorise à invoquer les autres outils. L'automatisation des traitements est possible via une zone de texte éditable. La figure 1.9 représente une capture d'écran de cette interface utilisateur.

1.3.1 Lien entre les outils et le modèle

Principe Tous les éléments de l'architecture sont isomorphes. Cela signifie qu'ils possèdent une interface logicielle commune. En revanche, la réaction aux messages est propre à chaque objet. Cette caractéristique disjoint l'algorithmique et le modèle.

Les algorithmes sont décrits à un niveau abstrait (envoi de messages), sans autre hypothèse sur la nature des objets sollicités que le respect de l'interface logicielle. Ce mécanisme d'articulation entre le modèle et les algorithmes garantit que tout objet du modèle est manipulable par les algorithmes. Tout assemblage de ces éléments est lui même manipulable. La seconde caractéristique est le contrôle des objets sur la façon dont ils sont manipulés (au moyen de leurs méthodes). Les méthodes décrivent la mise en œuvre des traitements.

Illustration : le routage L'algorithme de routage point-à-point repose sur un mécanisme d'expansion de vague. L'origine de la vague est la source du signal à router. Chaque point atteint est inséré dans la liste d'expansion. A chaque étape, la tête de liste reçoit un message lui demandant de fournir la liste des points atteignables par connexion directe. Si le point source est une ressource de routage, il retourne une collection de points atteignables et leur coût associé. Chacun des nouveaux points est à son tour inséré dans la liste d'expansion. Si le point source est un agglomérat, la requête est sous-traitée aux objets encapsulés. L'agglomérat a pour charge de collecter les résultats intermédiaires.

Tous les objets répondent au message par une collection, éventuellement vide si l'opération est vide de sens pour eux. De la sorte, le calcul de la vague d'expansion peut être assuré sans prendre en compte la nature des objets présents dans cette vague.

1.3.2 Compilation du modèle

La description grammaticale est saisie par le programmeur dans l'interface de la figure 1.8.

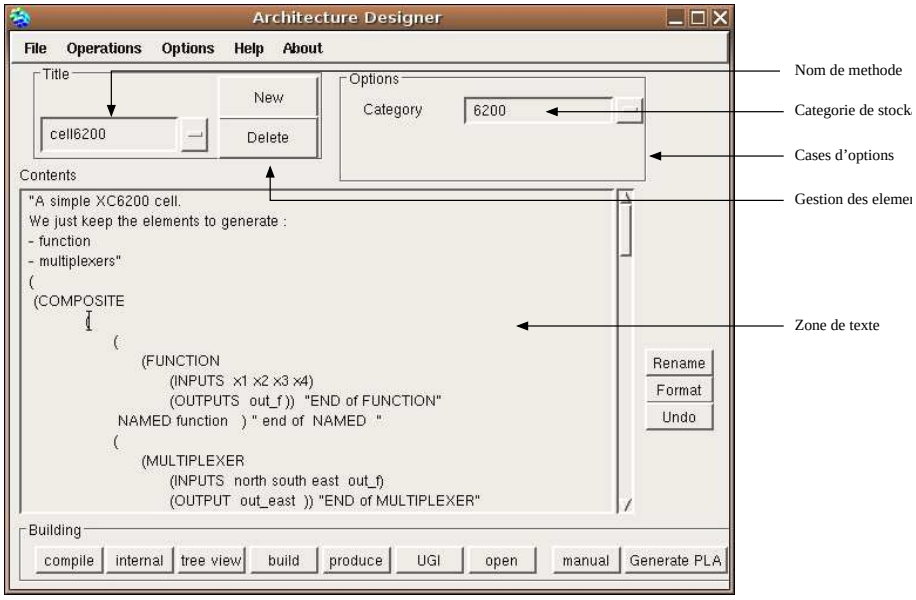


FIG. 1.8 – Vue de l'outil graphique de compilation

Ce dernier dispose d'aides à la programmation (menu *architecture*). Il lui est également possible de formatter son code, de sélectionner une catégorie de méthode, etc...

Le menu *command* permet de contrôler la chaîne de production d'architectures. Il est également possible d'ouvrir une interface utilisateur sur l'architecture produite (cf figure 1.9).

1.4 Intérêt de l'approche

Production rapide d'outils L'approche du projet MADEO tient en trois points. Une modélisation du domaine ciblé est tout d'abord conçue après analyse comparative des différentes architectures. Dans un second temps, un jeu d'outils est mis en œuvre pour manipuler cette modélisation. Celle-ci est ensuite spécialisée pour représenter une architecture donnée. Toute architecture modélisée est manipulable au travers de ces outils, sans intervention sur les outils. La nature arbitraire de la composition des éléments de l'architecture abstraite en vue de produire une architecture concrète, d'une part, et la nature extensible de l'architecture abstraite, d'autre part, assurent que toute architecture peut être représentée. Un corrolaire des deux points précédents est que les outils sont applicables à toute architecture. Par ailleurs, la définition grammaticale d'une architecture concrète étant un mécanisme rapide, la production d'outils pour une architecture dans son ensemble est aussi un mécanisme rapide.

Pérennité des outils Un bénéfice direct de l'approche présentée est la pérennité des outils. Les mêmes outils s'appliquant à toute architecture, leur durée dans le temps n'est

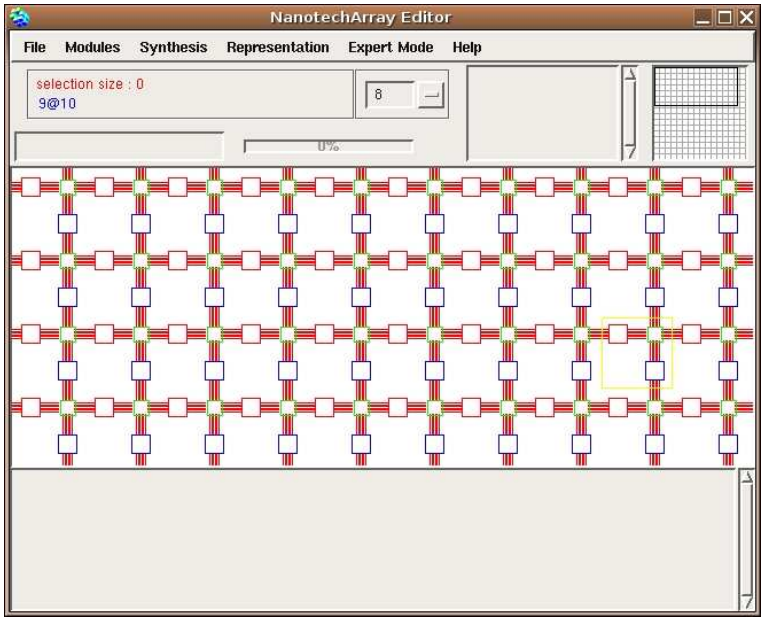


FIG. 1.9 – Capture d'écran de l'interface utilisateur montrant une approche précoce [75].

pas liée à celle d'une famille architecturale. Cette caractéristique est un double gage de qualité. La réutilisation des outils dans différents cadres augmente la généralité des traitements. Les hypothèses implicites dans la modélisation (ou les outils) sont détectées puis supprimées. Un second facteur de qualité est la diffusion immédiate à toutes les architectures modélisées des modifications apportées aux algorithmes. Toute évolution algorithmique théorique est transposée à moindre coût à l'ensemble des architectures.

Exploration architecturale L'exploration architecturale vise à isoler des architectures ou des critères architecturaux assurant de bonnes performances pour certains types de traitement. Le cadre logiciel se prête bien à l'exploration architecturale, les mêmes traitements sont applicables sur toute architecture. Un exemple d'exploration architecturale (impact du grain logique et de la largeur des canaux de routage sur la routabilité) peut être trouvé dans [57] .

Exploitation transparente des ressources Une même description de circuit, y compris structurée, peut être appliquée à toute architecture modélisée : la compilation matérielle est transparente et les ressources virtualisées. Par exemple, l'architecture hétérogène datapath ST [7], conçue pour les applications de traitement numérique (image et signal), présente les ressources qui ne sont visibles du coté programmeur pour mieux exploiter la puissance de l'architecture matérielle.

1.5 Conclusion

Ce chapitre a présenté une introduction de l'environnement Madeo développé pour modéliser et qualifier les architectures reconfigurables. La modélisation met en œuvre une description des éléments atomiques et composites des architectures existantes, fournissant ainsi des modèles représentatifs. Les outils fournis par cet environnement sont génériques, et adaptables aux nouvelles architectures.

A titre de comparaison, cette approche se retrouve dans des outils tels que SIS [84], qui articule des jeux d'algorithmes autour d'un ensemble de formats textuels, permettant l'échange de données. Chaque outil ayant un champ d'applications bien défini, possède une représentation interne de ses informations et des mécanismes d'import/export vers les formats d'échange, et permet de déclencher l'exécution des autres outils. Cette organisation garantit l'extensibilité du cadre logiciel (un ajout d'outil se traduit au besoin par la mise en place de nouvelles passerelles vers les formats d'échange).

Appliqué au domaine des architectures reconfigurables, une telle approche se retrouve également dans VPR [5]. Un modèle d'architecture est décrit et manipulé par des outils en charge de différents étages de la synthèse d'application (packer, placeur, routeur, estimateurs).

Madeo se distingue principalement de VPR par sa capacité à adresser des spectres d'architectures plus larges : chemins de données reconfigurables, architectures irrégulières. Par ailleurs Madeo inclut une couche haute, servant à décrire les applications sous forme de code comportemental objet comparable à [15].

Au niveau bas, c'est-à-dire en sortie de Madeo, la configuration des architectures reconfigurables apparait classiquement sous la forme d'une liste de ressources à allouer, ce qui s'apparente aux descriptions JBits [41] permettant de programmer les ressources des FPGAs Virtex.

Enfin, des travaux ont porté sur la génération de code visant à la synthèse technologique des architectures qui sont d'avantage à comparer à des outils comme MADMACS [34], en ce sens qu'une description dans un langage de haut niveau, interprétable, produit l'équivalent d'un masque.

Madeo apparait comme un environnement puissant, couplant de façon originale des niveaux de synthèse d'application, de synthèse physique et potentiellement de synthèse technologique, tout en garantissant une bonne capacité d'adaptation à de nouveaux cadres technologiques.

En particulier, le chapitre suivant introduit un certain nombre de technologies en émergence ; le chapitre 3 démontre la capacité de Madeo à adresser ces technologies.

Chapitre 2

Nanotechnologies

2.1 Technologies futures

L'intérêt porté aux nouvelles technologies s'avère grandissant et s'explique par les bouleversements en termes de performances et coûts qu'elles peuvent offrir dans tous les domaines de la science. La maîtrise de l'infiniment petit conduira par exemple dans le domaine de l'informatique à élaborer de nouveaux et puissants calculateurs qui traiteront efficacement l'information. Pour atteindre une telle efficacité de calcul, il faut une analyse et une étude complète de ces technologies, lesquelles donneront satisfaction par rapport aux attentes. Ces technologies futures devraient apporter certains besoins fondamentaux et avoir certains attributs attractifs pour justifier l'investissement nécessitant de construire une nouvelle infrastructure. Elles devraient satisfaire les conditions suivantes :

- la mise à l échelle fonctionnelle devrait être largement supérieure à celle connue en technologie CMOS ;

- une énergie dissipée minimale par opération fonctionnelle ;

- un coût minimal extensible par fonction ;

- un ratio élevé d'entrées/sorties pour le traitement de l'information.

Récemment, plusieurs groupes de recherches ont proposé un certain nombre de dispositifs nanoélectroniques qui exploitent les effets mécaniques à l'échelle nanoscopique. Parmi ces technologies, il y a les composants à un seul électron fondés sur le transfert contrôlable d'électrons isolés entre les îlots conducteurs (les SETs [48]), les dispositifs à base de nanotubes et de nanofils qui proviennent de l'électronique moléculaire [21, 66, 49], les automates cellulaires quantiques (les QCAs [90]), les dispositifs résonant tunneling (Diodes, FETs [14]), etc. Dans cette section, nous présentons le principe de conception et de fonctionnement de ces technologies. Tous ces dispositifs possèdent des propriétés fondamentales utilisables pour la création des nouvelles fonctions à la base des composants électroniques de la nouvelle génération. Nous répertorions leurs avantages qui suscitent autant la convoitise, et leurs inconvénients qui restent autant de défis à surmonter pour les rendre réalisables. Nous donnons quelques applications primaires ou composants électroniques qui peuvent contenir ces dispositifs (dans leur réalisation).

2.1.1 SETs

Ce sont des dispositifs qui résultent de l'électronique à un électron dont le principe physique repose sur le phénomène de blocage de Coulomb qui provient de la quantification d'une charge élémentaire dans un nœud isolé à double jonctions.

2.1.1.1 Principe de fonctionnement : transistor SET

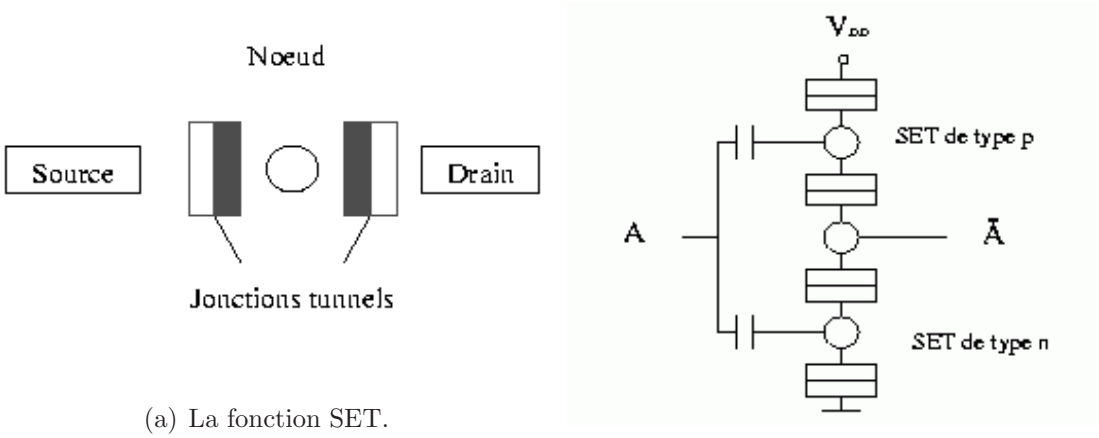
Un dispositif SET est constitué d'un nœud séparé des électrodes par deux jonctions tunnels comme le montre la figure 2.1(a). Son fonctionnement est conditionné par la valeur de la capacitance : si la capacitance du nœud est très faible de l'ordre de 1 à 200 aF, le potentiel du nœud doit changer considérablement quand un électron passe à travers le nœud. Les jonctions correspondent à une barrière énergétique séparant les deux conducteurs dont les propriétés géométriques et physiques sont telles qu'il est possible pour une charge de la traverser par effet tunnel. Le principe de fonctionnement est le suivant : quand la jonction tunnel résultante est alimentée par un courant de source constante, si la charge Q à la surface de la jonction est plus grande que $+e/2$, un électron peut traverser la jonction par effet tunnel dans une direction particulière, diminuant ainsi la charge Q de e. De même si la charge Q est inférieure à $-e/2$, un électron peut traverser la jonction dans la direction opposée, augmentant la charge Q de e. Si la charge Q est comprise entre $-e/2$ et $e/2$, il n'y a pas d'effet tunnel dans aucune direction : c'est le blocage de Coulomb.

Le transistor SET est un dispositif logique permettant de transférer un par un les électrons de la source au drain : l'électron représente l'état logique du système. La disponibilité d'un tel dispositif permet de modéliser les circuits logiques (ALU, multiplexeur, ...) et les mémoires pour le traitement de l'information : la figure 2.1(b) illustre une conception représentative dans un inverseur SET. Il possède une structure semblable à celle de l'inverseur classique. La grande différence vient du fait que le canal est isolé de la source et du drain par les jonctions tunnels. Le rôle du canal est joué par un puit quantique (nœud). Le transistor SET présente deux caractéristiques majeures :

- une oscillation de Coulomb en tension, produite lorsque la tension d'alimentation est inférieure à la tension de seuil, permet d'envisager le fonctionnement de certaines portes logiques ;
- la tension de seuil est une fonction périodique de la charge Q dans le nœud de Coulomb. Certaines portes logiques comme l'inverseur sont basées sur cette caractéristique.

2.1.1.2 Propriétés

Les SETs possèdent des propriétés séduisantes pour implémenter des systèmes ultra-denses et complexes de traitements d'information. Cela est dû à leur petite taille et leur faible consommation d'énergie : la technologie de fabrication classique permet d'intégrer 10^{11}transistors SETs par cm^2 et une puissance de consommation de 1nW par transistor [9].



(a) La fonction SET.

(b) La jonction SET inverseur.

FIG. 2.1 – Les circuits élémentaires de la technologie SET.

L'objectif principal est de concevoir des circuits logiques fonctionnant à la température ambiante et compatibles à la technologie CMOS en utilisant les méthodes de fabrication CMOS pour surmonter les difficultés intrinsèques imposées par leur taille et l'absence de stabilité dans le fonctionnement en température ambiante.

2.1.1.3 Applications logiques

Les portes logiques L'approche adoptée pour modéliser les circuits SETs est de coupler capacitivement les jonctions SETs car ils offrent une complémentarité dans la famille SET pour permettre d'établir facilement la logique. Ainsi l'élément de base de la logique dans cette famille est l'inverseur constitué de quatre jonctions tunnels et trois nœuds comme le montre la figure 2.1(b). Les deux SETs qui le composent opèrent de manière complémentaire. En effet, si le SET de type p conduit, celui de type n est bloqué et vice versa. Chaque fois que la capacité est chargée par le SET de type p, elle est déchargée par le SET de type n.

La simulation de ce dispositif a été faite par le logiciel SIMON [61] : la courbe comportementale présente trois régimes qui correspondent au fonctionnement d'un condensateur (chargement, transition et déchargement). La taille du régime de transition est limitée par le rapport entre les capacitances de la grille et de la jonction tunnel. La tension de sortie change plus rapidement que celle d'entrée dans le régime transitoire quand la sortie passe du niveau logique haut au niveau logique bas. Le gain de tension est possible à une température de l'ordre de 30 K (1.2 à 27 K), au delà de cette température, il n'est plus possible, encore moins en température ambiante. Les plus grands gains de tensions sont observés pour des faibles températures (5.2 à 200 mK).

Malgré la limitation imposée par la température au fonctionnement des circuits SETs (difficulté à marcher eb température ambiante), certaines portes logiques sont simulées. En particulier, les portes NOR/NAND sont implémentées en appliquant à une tension de contrôle une valeur appropriée. Comme pour la logique CMOS, la technologie SET peut

être mis en cascade pour former un réseau logique pouvant exécuter toutes les opérations logiques. Les fonctions AND et OR peuvent être réalisées avec deux dispositifs SET. XOR nécessite cinq dispositifs [26].

Mémoires Des dispositifs mémoire peuvent être conçus à base de SET. Si une mémoire à un seul électron (SEM) peut être réalisée, la capacité mémoire peut atteindre sa plus grande limite. Le SET peut être utilisé comme une cellule mémoire puisque l'état du nœud de Coulomb peut changer en présence ou en absence d'un électron. Un groupe de chercheurs du laboratoire Hitachi à Cambridge a développé une cellule mémoire hybride SET/CMOS dans laquelle l'information est stockée sur une nœud de la mémoire similaire aux transistors [39]. La différence de tension entre les état binaires 0 et 1 est environ 0.14 V et correspond à la présence ou l'absence des électrons sur le nœud. D'autres chercheurs Chou [43] et Chan [89] ont démontré la possibilité de fabriquer une mémoire basée sur les SETs. Chaque SET est réalisé en imbriquant une ou plusieurs nanoparticules de Silicium dans une mince couche isolante de SiO_2 et en arrangeant le drain et la source autour du nœud de Coulomb. Le temps de lecture/écriture de la structure est de l'ordre de 20 ns, la durée de vie est au dessus de 10^9 cycles, et le temps de présence de l'électron dans ce nœud est de plusieurs jours à plusieurs semaines.

Circuits hybrides La substitution des dispositifs classiques par les circuits SETs ne produit pas de bons résultats pour plusieurs raisons. Le gain maximal disponible d'un circuit SET est très petit, souvent inférieur à 1. La résistance de sortie est faible et la tension applicable au drain est limitée à une petite valeur pour satisfaire la condition de blocage de Coulomb (condition de fonctionnement du SET). Pour pallier ces inconvénients, il est préférable de développer des circuits spécialisés pour les SETs en considérant leurs caractéristiques. L'approche utilisée est de construire un circuit hybride contenant des transistors MOSFETs qui ont un gain élevé, une grande résistance en sortie et une grande tension applicable et qui peuvent suppléer les SETs. Cette logique fournit une fonction d'interface et les circuits spécialisés aux SETs.

Cette technologie hybride SET/CMOS qui exploite les avantages particuliers du SET et du CMOS permet de consolider la conception des dispositifs SETs et offre une complémentarité qui fournit de nouvelles fonctionnalités et de hautes performances puisque la faible consommation d'énergie de SET est sollicitée et que les avantages du CMOS tels la grande vitesse, la conduction, le gain de tension et l'impédance d'entrée peuvent compenser certains inconvénients de la technologie SET.

2.1.1.4 Conclusion

La famille SET fournit des avantages utiles à l'amélioration des performances aux outils de traitement de l'information. Elle présente un style de conception similaire à celui du CMOS et une faible consommation d'énergie de l'ordre de 0.01 à 10 nW par porte [9](pour une tension appliquée de quelques millivolts) qui est purement statique car la puissance dynamique est quasi négligeable puisque les caractéristiques de l'inverseur se

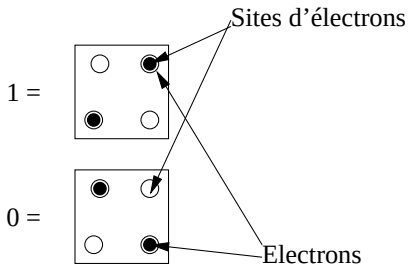
dégradent avec la température, ce qui est contraire à la famille CMOS. Elle offre une densité d'intégration importante et une large fonctionnalité. Leur réalisation physique reste en partie freinée par leur difficulté à fonctionner à température ambiante. Ces dispositifs sont également perturbés par un bruit dû aux charges de fond. La performance de cette famille à hautes fréquences s'avère limitée par la capacité réduite de conduire les charges de hautes capacitances.

2.1.2 QCAs

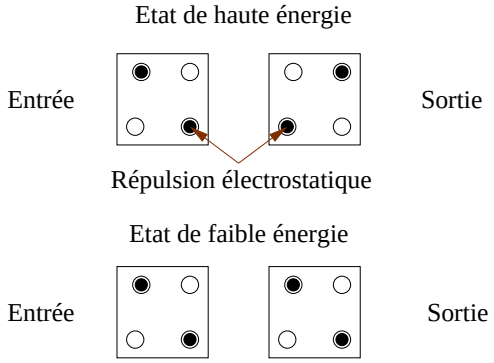
Les automates cellulaires quantiques sont des remplaçants potentiels pour les technologies classiques en ce sens qu'ils présentent des propriétés intéressantes pour résoudre le problème des limitations des technologies classiques. Ces nanostructures offrent des possibilités de calcul en utilisant un ensemble de cellules dotées pour implémenter les fonctions logiques digitales. Les éléments caractérisant la logique sont revisités pour comprendre le fonctionnement d'une telle technologie.

2.1.2.1 Concept de QCA

Inventé par Lent et Porod de l'université de Notre Dame aux Etats Unis, le concept de QCA présente des dispositifs nanotechnologiques capables de jouer le rôle des semiconducteurs au silicium qui serviraient à améliorer le mécanisme du routage, ce qui est rendu possible en utilisant des buffers et des registres à base de QCAs. Une cellule de QCA contient généralement quatre sites dont deux sites sont habités par deux électrons qui permettront à la cellule d'interagir avec les cellules voisines par la force coulombienne. La distribution de charge permet la configuration de la cellule en deux états binaires qui sont utilisés pour transmettre l'information. C'est donc l'effet interne de la cellule qui est à l'origine de la représentation digitale de l'information (figure 2.2).



(a) Etats de configuration.



(b) Interaction coulombienne entre deux cellules.

FIG. 2.2 – Codage d'information.

La topologie de QCA est constituée d'un pavage de cellules. Ces cellules sont combinées

de façon à modéliser les éléments nécessaires pour l'interconnexion et former la logique. Les circuits QCA peuvent donc être implémentés mais souffrent de contraintes qui permettent de les actionner sous un régime d'horloge à quatre phases. Pour éviter la perte de cohérence des circuits, un champ électrique est appliqué dans les cellules et permet d'élever ou de baisser les barrières d'énergies tunnel de sites, ce qui empêche ou permet aux électrons de changer de position (donc influencer les cellules voisines). Les cellules peuvent être localisées dans des zones de sorte que le champ influençant toutes les cellules soit le même. Chaque zone fait un cycle de quatre phases (figure 2.3) :

- au cours de la phase Switch, les barrières d'énergie dans la zone sont élevées. Les électrons internes à la cellule peuvent être influencés par des charges coulombiennes de zones voisines ;
- la phase Hold est caractérisée par une haute barrière d'énergie et par l'absence de changement d'état. Une zone est influencée par les zones adjacentes ;
- les autres phases (Release et Relax) diminuent les barrières d'énergie et la zone n'influence pas les autres zones.

Les zones peuvent être de forme irrégulière mais leur taille devrait être imposée par les aspects de fabrication et de dissipation [38]. Les cellules de QCA ont été expérimentalement implantées en utilisant des sites métalliques mais la meilleure option est l'implantation moléculaire qui offre une densité d'intégration à l'échelle nanométrique plus importante [4]. Les dispositifs à base de QCA qui permettent de former la fonctionnalité logique sont essentiellement formés de lignes binaires, de l'inverseur, et de la porte majorité (MV) qui sont définis dans les parties suivantes.

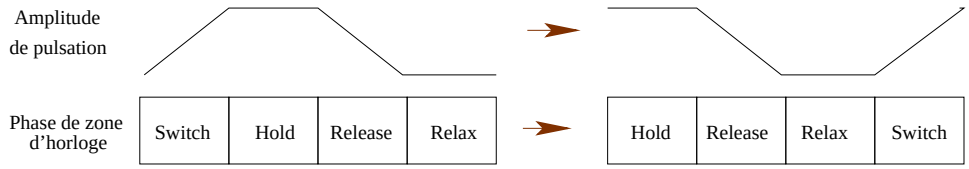


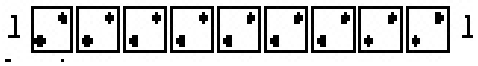
FIG. 2.3 – Déplacement de données à travers une horloge de QCA.

2.1.2.2 Conducteurs

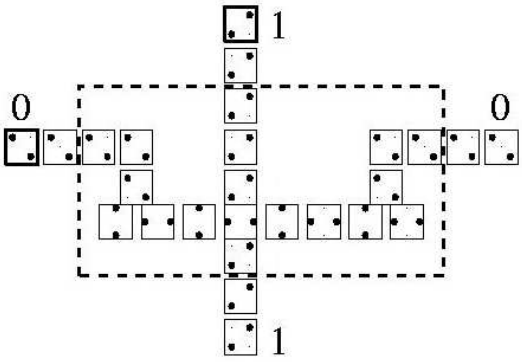
Ligne binaire Une ligne binaire est un pavage linéaire de cellules dans différents états de polarisation. Chaque état d'une cellule influence l'état de la cellule voisine. Cette combinaison linéaire est utilisée pour transmettre l'information binaire d'un point à un autre. Une fois que la polarisation de la première cellule de la ligne est fixée, les autres cellules libres s'alignent dans la même direction (figure 2.4(a)). Dans ce cas, l'information contenue dans le conducteur est transmise à l'autre bout du fil. La ligne de QCA est robuste contre les variations des paramètres d'une cellule à une autre, c'est à dire lorsque la cellule conductrice ou les cellules intermédiaires sont faiblement polarisées, la ligne transmet toujours la valeur binaire attendue. Une cellule pivotée à 45 $^{\circ}$ est identique à la

cellule standard. Cependant, les polarisations de deux cellules voisines font qu'elles sont alignées d'une manière opposée. Cet anti-alignement entraîne des polarisations alternées des cellules dans un fil conducteur. Une telle ligne est appelée 'chaîne inverseuse'. Si la longueur de la chaîne est connue, l'information envoyée est simplement déterminée.

Lignes croisées A cause de la nature des cellules QCA, le croisement de deux lignes se fait en utilisant un arrangement coplanaire des cellules. La figure 2.4(b) montre une manière de croiser deux lignes de cellules QCA. La ligne horizontale transmet un 0 et la verticale un 1. Certaines cellules contenues dans la ligne horizontale sont pivotées et se comportent comme les cellules d'une chaîne inverseuse. Le signal sur cette ligne est transmis aux cellules pivotées par un arrangement spécial de trois cellules : la conversion nécessite que la cellule standard soit au dessus de deux cellules pivotées car l'état d'une cellule standard n'a pas directement d'effet de commutation sur une cellule pivotée dans une ligne et vice versa. La ligne verticale contient des cellules normales sans interférence. Ainsi, le système transmet correctement toutes les combinaisons possibles de deux signaux sans interférence.



(a) Fil polarisé : transmission du binaire 1 à travers le pavage de cellules.



(b) Fils croisés : une transmission verticale du binaire 1 et une autre horizontale du binaire 0.

FIG. 2.4 – Configuration de lignes conductrices en technologie QCA.

2.1.2.3 Applications de base

2.1.2.4 Inverseur

Les cellules orientées diagonalement sont similaires aux cellules pivotées horizontalement. Elles s'alignent dans des directions opposées comme la chaîne inverseuse. Cet anti-alignement peut être utilisé comme un inverseur QCA. Dans ce cas, le pavage linéaire doit contenir un nombre pair de cellules. Toutefois, un autre arrangement de cellules présente une conception qui agit comme un inverseur (figure 2.5). Le signal entre par la gauche d'une ligne binaire. Au bout de celle-ci, il est scindé en deux lignes parallèles excentrées et inversé au point de convergence. Cette conception est géométriquement symétrique.

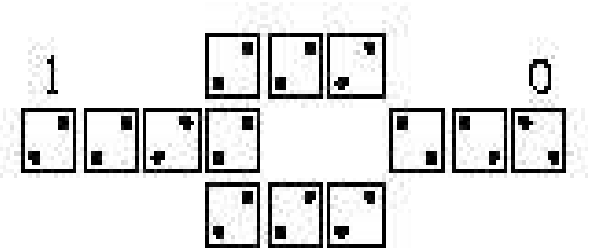


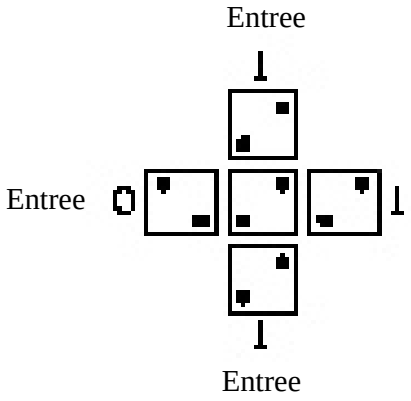
FIG. 2.5 – Configuration d'un inverseur.

2.1.2.5 Porte majorité

La porte de base dans une structure QCA est symboliquement l'électeur de majorité qui peut être réalisé au moins avec 5 cellules comme le montre la figure 2.6(a). Les états d'entrée des cellules sont fixés pendant que la cellule centrale et celle de sortie réagissent librement à la charge fixée. Cet élément logique est symbolisé par un bloc MV ayant trois entrées et une sortie (figure 2.6(b)). Sa fonction logique s'exprime par :

$$MV(A, B, C) = AB + BC + AC \tag{2.1}$$

Les portes programmables AND/OR peuvent être implémentées à partir de cette porte majorité en fixant de manière permanente une entrée. Cette entrée fixée est appelée *program*. Une porte programmable AND/OR peut être réduite pour implémenter une porte OR dédiée en fixant une fois pour toute la cellule *program* à l'état 1. De même, la porte AND dédiée peut aussi être implémentée en fixant la cellule à 0.



(a) Porte majoritaire.

(b) Symbole de la porte majorité.

FIG. 2.6 – Configuration d'une porte majorité.

2.1.2.6 Applications complexes

Un certain nombre de dispositifs peuvent être modélisés à partir des portes MV et des inverseurs. La conception d'un additionneur complet est proposée dans [90]. Il est

implémenté en utilisant 3 portes MV et 2 inverseurs. Le dispositif contient en tout 145 cellules QCA. Les mémoires peuvent être implémentées à base des cellules de QCA en adaptant leur conception de façon à éviter les problèmes liés à la nature des QCAs. La conception de la mémoire doit éviter une conception bidimensionnelle et une cellule ne doit pas contenir un seul bit car la densité de la mémoire QCA peut être important en stockant plusieurs bits dans une cellule mémoire (multiples cycles d'horloge rentrent en jeu rendant la génération et la propagation des signaux complexes). La structure basique proposée est une structure en H récursif qui est constituée de routeurs formant les nœuds intérieurs et de macro mémoires formant les feuilles [32]. C'est un système hautement compact et extensible avec des temps d'accès uniformes. Pour accéder à une cellule mémoire, un paquet de données est construit et séquentiellement envoyé vers une adresse ou une ligne de données. A la tête du paquet se trouve un bit d'adresse de la cellule mémoire. Un autre bit de code d'opération indique une opération d'écriture ou de lecture.

Pour conclure la technologie QCA offre un certain nombre d'avantages qui la rendent attractive par rapport aux technologies classiques. Elle possède une haute densité fonctionnelle d'intégration (50 Gbits/cm^2 possible [4]), une dissipation très basse d'énergie, l'absence d'interconnexions physiques entre les conducteurs et des vitesses de calcul élevées. La structure de la mémoire en H offre des latences réduites et une flexibilité qui permet d'envisager des optimisations. Toutefois, cette technologie présente un certain nombre de défis liés à sa nature : sensibilité à la charge de fond (sensible aux bruits extérieurs) [31], fonctionnement à faible température.

2.1.3 L'électronique moléculaire

L'électronique moléculaire s'intègre aussi à la nanoélectronique, lorsqu'elle s'intéresse à assembler des molécules dont les dimensions se situent à l'échelle nanométrique. Elle se présente comme le premier choix de la liste des technologies post silicium. Elle jouit d'une attraction pour la construction des systèmes de traitement : les dispositifs dérivés conservent la même abstraction que dans les dispositifs classiques comme les transistors, les diodes, etc. Ce champ émergent utilise des molécules agissant comme des conducteurs, des diodes et des résistances. Elle offre alors une capacité de faire de la logique et des mémoires moléculaires en garantissant une nouvelle plateforme pour remplacer la plateforme classique. Nous répertorions les dispositifs existants et leurs caractéristiques pour mieux envisager la conception des architectures de calcul. Dans la section 2.1.3.1, nous présentons les dispositifs filiformes pouvant jouer le rôle de conducteur et leurs propriétés de transport électrique. Nous introduisons dans la section 2.1.3.2 les dispositifs bidimensionnels construits à partir de ces conducteurs.

2.1.3.1 Dispositifs filiformes

Le progrès incessant dans le domaine des matériaux à l'échelle moléculaire nous amène à concevoir des systèmes unidimensionnels tels les nanofils ou les nanotubes d'épaisseur de quelques nanomètres en diamètre et de plusieurs microns en longueur. Ces systèmes

montrent des propriétés électroniques, optiques et mécaniques remarquables pour construire des dispositifs plus élaborés qui sont utiles pour fabriquer les grands circuits électroniques : leur impact potentiel dans les aiguillages et les contacts moléculaires.

Nanotubes de carbone Les nanotubes de carbone (en anglais Carbon Nanotubes ou CNT) constituent un sous ensemble spécial et important des matériaux électroniques moléculaires. Leur structure moléculaire se présente sous la forme de cylindre creux obtenu en roulant sur elle-même une feuille de graphite. Elle est composée d'atomes de carbone organisés suivant un maillage hexagonal : au départ, les atomes sont groupés dans un ensemble d'hexagones qui forment une couche planaire semblable à une feuille atomique de graphite et ensuite la couche est enroulée pour former un tube (figure 2.7). Le diamètre du tube varie entre 1 à 20 nm et la longueur peut atteindre plusieurs microns. Chaque atome de la structure est lié à trois atomes voisins. C'est cette liaison covalente qui détermine les propriétés du tube. Les nanotubes appartiennent à la famille de fullerène (molécule en forme de cage comportant $2(10 + n)$ atomes de carbone formant 12 pentagones et n hexagones [59]).

Suivant les conditions de synthèse, ces structures cylindriques se divisent en deux catégories : les nanotubes monofeuillets (SWCNT pour Single-Walled Carbon Nanotube) et les nanotubes multifeuillets (MWCNT pour Multi-Walled Carbon Nanotube).

1. Nanotube monofeuillet : un réseau des atomes de carbone enroulé sur lui-même sous forme cylindrique. Les nanotubes ont des diamètres uniformes formant ainsi des faisceaux à la manière des cordes. Dans chaque faisceau, les tubes s'empilent et forment un arrangement périodique de symétrie triangulaire. Le nombre de tubes peut atteindre plusieurs dizaines dans un faisceau et leur diamètre peut varier selon les conditions de synthèse de 0.6 à $1.5nm$.
2. Nanotube multifeuillet : les tubes s'emboîtent les uns dans les autres formant un arrangement coaxial de plusieurs feuillets d'atomes de carbone enroulés sur eux-mêmes. Le nombre de feuillets et leur diamètre sont très variables.

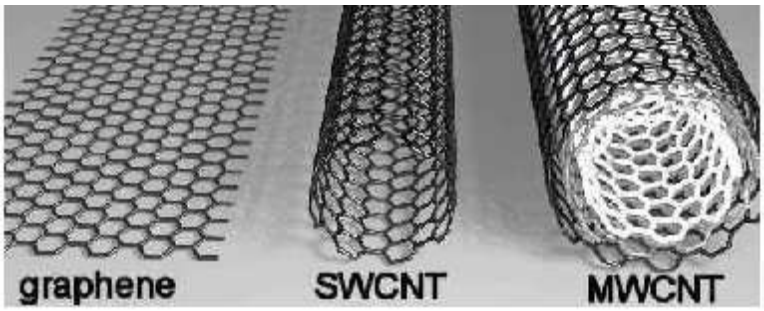


FIG. 2.7 – De la feuille de graphite aux nanotubes monofeuillet et multifeuillet.

Le caractère unidimensionnel et la structure forte des liaisons carbone-carbone donnent aux nanotubes des propriétés remarquables . En effet, les nanotubes présentent des pro-

priétés mécaniques, électriques, thermiques et chimiques hors du commun qui dépendent de la structure géométrique du graphène déterminant la structure en bande du tube [95, 13] : 100 fois plus résistant que l'acier, émission à meilleur émetteur à effet de champ connu conduit de forte densité de courant ($10^9 A/cm^2$ [23]), conductivité thermique comparable à celle du diamant.

Les propriétés de conduction électrique découlent de la structure électronique de bande du nanotube. Les atomes de carbone qui composent le tube possèdent quatre électrons de valence dont un seul détermine les propriétés de transport. Ces électrons se déplacent librement dans la direction axiale du tube. Les propriétés électriques proviennent donc du confinement des électrons libres dans cette direction transversale. C'est pour cette raison que les nanotubes de carbone agissent comme des fils conducteurs. Le comportement électrique d'un nanotube est fortement dépendant du diamètre de son tube et de la nature de son enroulement (chiralité) [82, 65] : il peut agir comme un sémiconducteur, un conducteur métallique ou un isolant.

De telles propriétés permettent d'envisager plusieurs applications telles que des dispositifs à émission à effet de champ, des dispositifs électroniques, le renforcement de matériaux composites, les polymères conducteurs, des senseurs, etc.

Ils définissent bien les structures électroniques unidimensionnelles utiles pour construire les structures bidimensionnelles mais leur dopage comme semiconducteurs est difficile. Ils sont utilisés dans une configuration de transistor comme canaux. Cela est réalisé en déposant les électrodes de source et de drain à chaque extrémité du tube.

Depuis la découverte des nanotubes, plusieurs méthodes de synthèse sont explorées pour les produire à grande échelle et de manière contrôlée. Deux catégories de méthodes de synthèse se dégagent et se distinguent par le niveau de température mise en œuvre : les méthodes de synthèse à haute température ($T > 3000\ ^\circ$C) et les méthodes de synthèse à moyenne température ($500\ ^\circ\text{C} < T < 1200\ ^\circ$C).

1. Synthèse à haute température : le principe des méthodes de cette classe consiste à évaporer du graphite et à le condenser dans une enceinte où règnent un fort gradient de température et une pression partielle d'un gaz inerte. Les différentes méthodes se distinguent par le procédé de vaporisation du graphite mis en œuvre. Les méthodes actuellement utilisées sont les suivantes : l'arc électrique, l'ablation laser et la méthode solaire.

2. Synthèse à moyenne température : le principe consiste à décomposer un gaz carboné à la surface des particules d'un catalyseur métallique dans un four porté à une température comprise entre 500 $^\circ$C et 1200 $^\circ$C selon la nature du gaz. Le carbone libéré par la décomposition du gaz précipite ensuite à la surface de la particule et cette condensation aboutit à la croissance de structures tubulaires graphitisées. La méthode CVD (Chemical Vapor Deposition) est la plus développée et prometteuse.

Nanofils de silicium D'autres dispositifs filiformes ont étés fabriqués. Ce sont des fils de silicium ayant des diamètres de quelques nanomètres : les nanofils de silicium [27, 18]. Ces dispositifs sont actuellement produits par l'arc électrique, l'ablation laser, le dépôt

en phase vapeur. Les nanofils de silicium ont une compatibilité idéale d'interfaçage avec les technologies classiques. La mesure de variation de leur conductance les présage à plusieurs usages : fils conducteurs, capteurs dans les environnement chimiques, etc. La composition d'un nanofil peut varier le long de son axe et en fonction de son rayon. Cela donne l'opportunité d'avoir les structures hétérogènes qui peuvent fournir des dispositifs contrôlables et des interconnexions imbriquées dans une seule structure.

Contrairement aux nanotubes dont l'accroissement et le dopage contrôlé des tubes semiconducteurs semblent difficiles, les nanofils peuvent être dopés pour contrôler leurs propriétés électriques [16] : le changement de la conductance en fonction de la tension appliquée peut être utilisé pour distinguer la nature des nanofils. De plus, le dopage contrôlé des nanofils avec le phosphore et le bore produit des semiconducteurs de type p et de type n. Une telle aubaine permet de les combiner pour former les jonctions p-n puisque la conduction à travers les nanofils hautement dopés peut être contrôlée par un champ électrique. La tension de seuil pour un nanofil peut être commandé par les propriétés de matériaux utilisés pour sa composition ou son dopage et par les facteurs géométriques[42].

2.1.3.2 Dispositifs complexes

Les nanotubes et les nanofils sont développés pour une variété d'applications pour les dispositifs électroniques. Ils peuvent être aussi utiles pour leur interconnexion que pour la construction des dispositifs actifs. En effet, les nanotubes et les nanofils peuvent être assemblés pour agir comme des diodes, des résistances ou des transistors [17, 46] : deux nanofils de type différent peuvent former une jonction de diode à leur intersection. De même, trois nanofils dont deux sont perpendiculaires au troisième peuvent forment un transistor. Le comportement électronique de ces deux dispositifs sont observés grâce aux mesures de courant/tension [81, 72]. Pour des tels dispositifs, les propriétés de la mécanique quantique sont exploitées pour contrôler les niveaux de courant et de tension afin d'améliorer leur comportement à l'échelle nanométrique.

Plusieurs portes logiques peuvent être construites à l'image de celles offertes par la famille CMOS. Des groupes de recherches ont démontré et fabriqué un ensemble des FETs en nanotube de carbone (CNTFETs figure 2.8). Des circuits utilisant de tels CNTFETs ont été également démontré et fabriqué [46] : la réalisation d'une porte OR nécessite au moins deux CNTFETs de type p en parallèle utilisés comme entrées qui croisent un autre CNTFET de type n servant de sortie. Les gains de tensions obtenus sont largement au dessus de 5 permettant ainsi l'interconnexion des portes entre elles sans restaurer le signal à chaque étage. Le même principe est utilisé pour implémenter toutes les portes élémentaires en associant les jonctions des nanotubes ou nanofils.

2.1.4 Technologies de programmation émergentes

Les technologies classiques de programmation utilisées pour mettre en œuvre les aiguilleurs programmables présentent des atouts et des inconvénients, ce qui explique

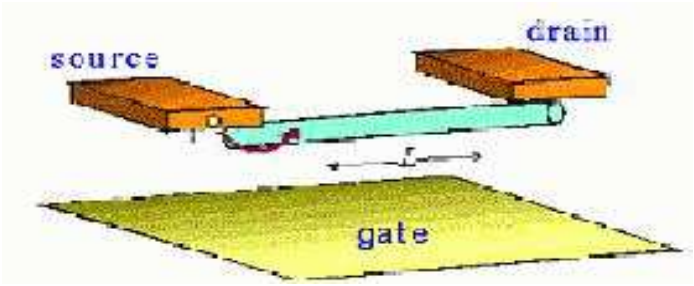


FIG. 2.8 – Un FET en nanotubes.

qu'aucune n'ait pris l'ascendant décisif par rapport aux autres ; la SRAM présente des caractéristiques de taille et de performance inférieures à celles des technologies concurrentes. Son attrait provient donc des possibilités de configurations multiples et rapides. L'antifusible est largement compact et introduit moins d'effets parasites. Il n'est pas reprogrammable. La technologie à grille flottante (EPROM) permet une densité comparable à celle d'une RAM dynamique mais elle consomme beaucoup. La capacité utile et les performances des circuits programmables résident dans les caractéristiques technologiques élémentaires des dispositifs programmables. Toute amélioration à ce niveau contribue à réduire l'écart opérationnel qui sépare les circuits programmables des circuits spécifiques.

Les nouvelles méthodes de synthèse ont permis aux chercheurs de mettre au point des systèmes moléculaires de plus en plus complexes. Les composés accessibles constituent la base des machines nanoélectroniques. Ils semblent parfaitement adaptés à l'élaboration des calculateurs puisque leur fonctionnement illustre un véritable comportement bistable qui permet de coder une information binaire. De tels dispositifs sont donc des interrupteurs qui contiennent leurs propres états et peuvent être programmés en utilisant les fils des signaux avec lesquels ils sont en contact. A l'échelle nanoscopique, ces aiguilleurs peuvent exhiber un comportement de commutation réversible [63, 79, 10, 47], caractérisant ainsi la nature reprogrammable de certains dispositifs. Dans les technologies classiques, le coût (spatial ou énergétique) d'une jonction programmable est largement important mais ces dispositifs émergents peuvent réduire considérablement cette charge, d'autant qu'un seul dispositifs peut posséder l'ensemble des performances des technologies actuelles. La figure 2.9 présente les technologies de programmation et leur nature programmable ou reprogrammable possible.

2.1.4.1 Aiguilleurs mécaniques

Certains nanomatériaux possèdent des propriétés de commutation lorsqu'ils sont suspendus. Ils se comportent ainsi comme des interrupteurs. C'est le cas de deux nanotubes de carbone suspendus en croix qui exhibent un interrupteur non-volatile bistable [52] : lorsque une tension appropriée est appliquée à travers la paire des tubes, ils se mettent en contact sous l'effet de la force de Van der Waals. Ils reviennent à leur position initiale lorsque une tension opposée est appliquée.

2.1.4.2 Aiguilleurs moléculaires

Les recherches les plus actives sont probablement orientées vers l'utilisation de molécules à plusieurs constituants, comportant des anneaux entrelacés (caténanes [47, 62, 10]) ou un anneau traversé par un axe (rotaxane [47, 62]). Par exemple, la structure de rotaxane est faite d'une tige et d'un anneau. Lorsqu'une source de tension est appliquée à la tige, la position de l'anneau change en fournissant une différence de résistance utile pour fixer les états d'ouverture et de fermeture. Cette fixation est répétée plusieurs fois, ce qui lui donne un caractère reprogrammable. C'est une molécule qui peut fonctionner en milieu ambiant avec des tensions d'ouverture de $2V$ et de fermeture de $-2V$. Son état peut être détecté avec une tension variant entre 0.1 et $0.3V$, ce qui correspond à une opération de lecture dans une cellule de stockage classique.

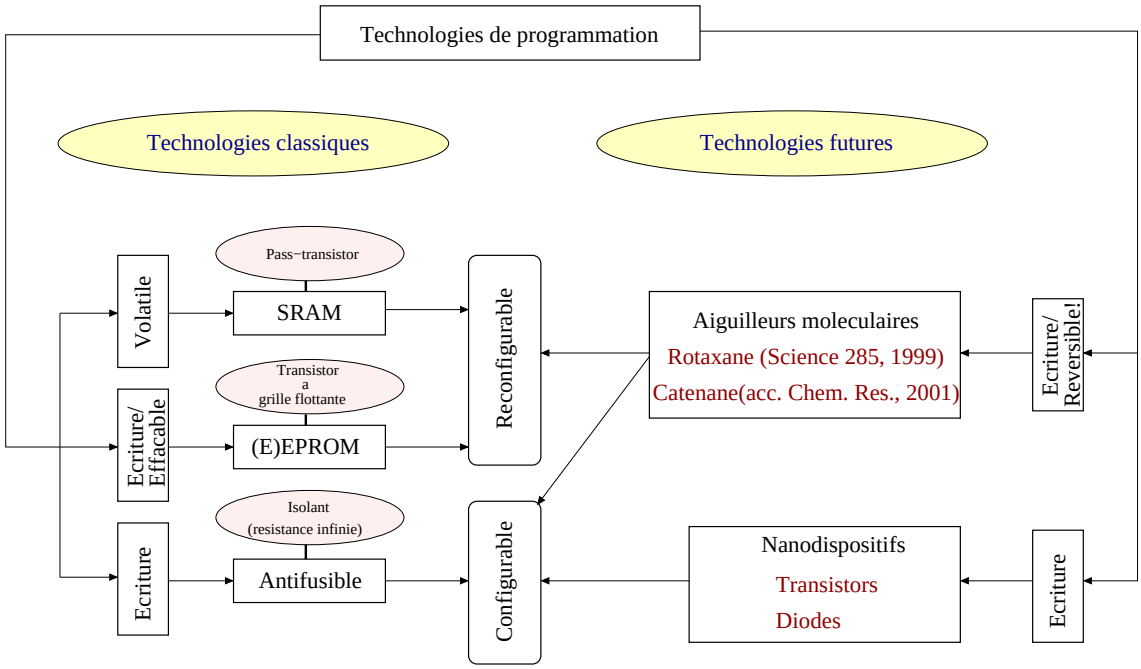


FIG. 2.9 – Différentes technologies de programmation classiques et émergentes.

En conclusion, les dispositifs issus des électroniques moléculaires constituent le premier choix de la liste en ce sens qu'ils sont réalisés à l'échelle moléculaire, donc plus de capacité de calcul et de mémoire. Leur fabrication est peu coûteuse puisque la chimie de synthèse est le moyen utilisé pour les faire. Ils favorisent une organisation structurée des circuits ou des dispositifs importants à l'échelle nanométrique (nanostructures) mais ces structures moléculaires contiennent beaucoup d'erreurs d'assemblage.

2.1.5 Techniques de fabrication

Nous nous intéressons ici aux techniques constructives que l'on peut qualifier "*approche bottom-up*". Par opposition, l'approche top-down, relevant de technologies de précision

telle que la lithographie, implique la fabrication de matériaux de *grande taille*, de l'ordre de quelques microns, tels les micropuces. Il résulte de cette seconde approche des difficultés à produire en grande quantité des systèmes complexes, car cela impliquerait une augmentation exponentielle des coûts de fabrication.

2.1.5.1 Auto-assemblage

Les techniques d'auto-organisation apparaissent comme le moyen le plus convainquant pour construire les nanostructures et l'assemblage des circuits en ce sens qu'elles permettent de produire des quantités énormes des dispositifs identiques à bon marché [37, 97, 86]. Les dispositifs ainsi fabriqués ont des tailles de quelques nanomètres. Les transferts d'électrons dans les jonctions sont opérés selon les principes de la mécanique quantique, à l'échelle moléculaire, pour contrôler les niveaux de tension et de courant à travers les terminaux de dispositifs.

On connaît deux procédés d'auto-organisation, l'auto-assemblage chimique et l'auto-assemblage physique. Ces deux procédés se distinguent des techniques lithographiques, en ce qu'elles mettent en action des réactions opérées progressivement entre plusieurs couches moléculaires. Le procédé chimique s'avère en général plus rapide que le procédé physique.

2.1.5.2 Auto-assemblage chimique

C'est donc une technique de production *bottom-up* dans laquelle les atomes ou les molécules s'organisent dans une structure nanométrique ordonnée par les interactions chimiques entre unités moléculaires absolument identiques. Au niveau expérimental actuel, on part de substances chimiques, que l'on fait réagir entre elles pour obtenir une géométrie particulière (les nanotubes, les nanofils, puis connexions de ces fils, ...) [3, 12]. Les chercheurs de l'Université de Colombia en partenariat avec IBM sont ainsi parvenus à réaliser l'assemblage des nanoparticules magnétiques et semiconductrices [70]. La méthode utilisée se déroule en deux étapes : la première étape consiste à produire des particules identiques selon la taille de la structure souhaitée (particules magnétiques d'oxyde de fer de diamètre 11 nm et les particules de séléniure de plomb de diamètre 6 nm). Les deux matériaux ont des propriétés dissemblables et complémentaires. Ils permettent de produire un composite AB_{13} ayant des propriétés magnéto-optiques. La seconde étape est l'auto-assemblage des particules sous certaines conditions expérimentales.

Une caractéristique notable de l'auto-assemblage chimique est que le principe de la synthèse chimique contrôlée fournit graduellement et expérimentalement des connaissances permettant d'optimiser la structure et les propriétés du système résultant [97, 73].

Les dispositifs chimiquement auto-assemblés ont les avantages suivants :

- ils sont faciles à préparer et à former à partir de solutions de molécules assemblantes. La fabrication massive sera peu coûteuse ;
- ils sont moléculairement ordonnés et robustes sous certaines conditions d'utilisation ;
- ils reflètent une stabilité thermodynamique, se forment spontanément et tendent à rejeter les défauts (les réactions réussies au niveau moléculaire) ;

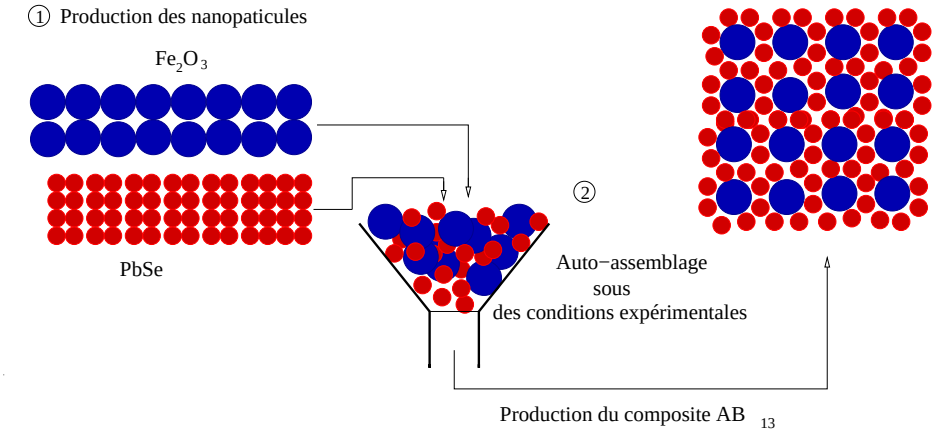


FIG. 2.10 – Etapes de la fabrication d'un composite AB_{13}.

– ils permettent de contrôler l'épaisseur d'une couche.

2.1.5.3 Auto-assemblage physique

En microélectronique, l'auto-assemblage physique se réfère à l'assemblage positionnel par lequel les atomes ou les molécules sont délibérément manipulés et positionnés un par un pour former un dispositifs. Cette opération est extrêmement laborieuse et n'est pas couramment disponible comme processus industriel à l'échelle atomique : les dépôts non uniformes créent les imperfections à l'échelle moléculaire qui peuvent exhiber de nouvelles propriétés pour les applications nanoélectroniques. Cette méthode englobe un ensemble de techniques tels SPM (Scanning Probe Microscopes) traitant les surfaces, dont l'utilité et la force dans l'industrie se situent dans leur capacité de caractériser et de mesurer les surfaces [86].

2.1.5.4 Contraintes

L'utilisation de l'auto-organisation impose une limitation forte sur l'architecture à l'échelle moléculaire : l'alignement précis des dispositifs et leur manipulation pour construire des circuits complexes semblent être difficiles à réaliser. Le problème principal qui se pose est celui de l'interconnexion des nanostructures compte tenu d'éventuels défauts d'alignement : relier les différents dispositifs pour en faire des circuits complexes est dur parce que l'utilisation des fils conventionnels à l'échelle nanométrique entraîne la perte de fiabilité de ces fils. Ils se cassent facilement lorsque la température est forte. Les solutions provisoires envisagées pour ces contraintes au niveau physique sont :

1. encapsuler l'interconnexion de deux dispositifs (nanotubes, nanofils,...) avec un autre dispositif moléculaire [37]
2. déposer une fine couche de supramolécule entre les fils [11].

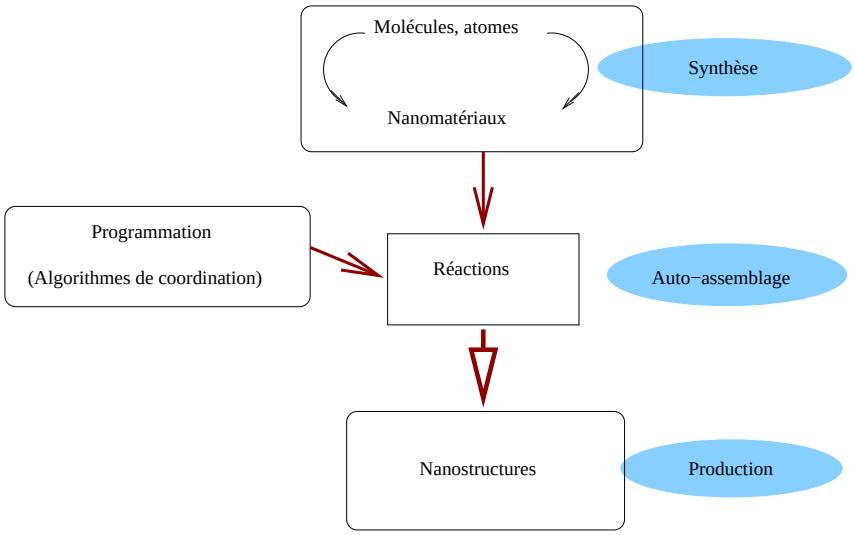


FIG. 2.11 – Principe de l'auto-assemblage.

2.2 Architectures

Le développement des nouveaux dispositifs technologiques nécessitent des nouveaux modèles d'architectures capables de faire face aux contraintes imposées par chaque technologie et d'exploiter leurs avantages au détriment de leurs inconvénients. Ces architectures devraient être géométriquement simples et régulières, et doivent permettre d'utiliser la reconfigurabilité pour implémenter les systèmes d'applications complexes. Dans cette section, nous présentons des nouvelles méthodologies architecturales proposées par plusieurs groupes de recherche. Toutes ces conceptions partagent pratiquement les mêmes objectifs et stratégies à haut niveau. Elles dérivent d'une structure crossbar qui est une approche générale pour les circuits moléculaires.

Une architecture crossbar se compose de deux plans parallèles de matrices de nanofils séparées par une fine couche ayant les propriétés électrochimiques particulières. Chaque plan contient un nombre de nanofils parallèles généralement de même type. Les nanofils de deux plans se croisent en formant un angle droit et leurs intersections constituent les jonctions disposées contenir les nanodispositifs configurables (diode, résistance ou transistor). Cette approche est appréciée par sa simplicité et son moindre coût à la fabrication en utilisant les techniques de fabrication (cf. 2.1.5.

2.2.1 Nanofabriques : travaux de Goldstein

Ce sont des architectures pour les électroniques moléculaires conçues pour surmonter les contraintes associées à l'assemblage des composants à l'échelle moléculaire et pour exploiter les avantages offerts par la nanoélectronique. Les dispositifs nanoélectroniques utilisés se restreignent aux nanofils, aux diodes et aux résistances. L'architecture profite

ainsi de la nature reprogrammable des aiguilleurs moléculaires (diodes).

2.2.1.1 Description

Seth C. Goldstein propose une architecture de matrices à l'instar d'un FPGA constituant les blocs logiques à l'échelle nanométrique nommés nanoblocs [36]. Les nanoblocs sont organisé en clusters (nanoclusters) qui sont connectés entre eux par les longs fils et placés sur un substrat de silicium (figure 2.12). Le substrat contient une circuit classique CMOS pour fournir la puissance, les signaux de contrôle, l'interface d'entrées/sorties et une logique.

Structure d'un nanobloc Le nanobloc est l'unité de base reconfigurable qui joue le même rôle qu'un CLB pour un FPGA. Il est nécessaire d'analyser la structure interne du bloc pour comprendre la granularité de la logique et justifier le type de fonctionnalités à implémenter. Le nanobloc peut implémenter une fonction booléenne à 3 entrées. Chaque signal logique étant apparié avec son complémentaire, un nanobloc possède au total 6 entrées et 6 sorties. Le nanobloc est composé de trois parties :

1. Molecular Logic Array (MLA) : il est composé de deux plans orthogonaux de fils de manière à former une matrice de matériaux d'aiguillages. Chaque intersection de fils abrite une diode moléculaire configurable. Cela donne la possibilité d'avoir des fonctionnalités. La nanofabrique entière pourrait être programmée à la manière d'un FPGA ou un autre composant reconfigurable. Chaque coté d'un bloc contient soit 3 entrées soit 3 sorties.

   L'utilisation des techniques pour mapper la logique à un crossbar par l'intermédiaire d'un décodeur programmé permet d'implémenter les PLAs dans des telles structures.
2. Interface E/S : il s'agit du domaine des points de connexion servant à relier un bloc aux blocs voisins. Les entrées/sorties du bloc sont rangées parallèlement de façon à implémenter les PLAs parce que la logique dans certains outils de synthèse des FPGAs est réalisée grâce à l'utilisation des LUTs, ce qui donne une simplicité dans l'usage des techniques de synthèse. Les sorties d'un bloc se connectent aux entrées d'un autre bloc à travers un bloc d'aiguillage reconfigurable.
3. Verrou (latch) : la structure du MLA présente une baisse des niveaux de tension et de courant lorsque les aiguilleurs reconfigurables sont parcourus par un courant ou alimentés par une tension. Pour la restauration du signal et les circuits séquentiels, les verrous NDR (Negative Diode-Resistor) sont utilisés à l'interface des nanoblocs. Ces dispositifs offrent non seulement des propriétés importantes pour la restauration du signal mais aussi l'isolement des entrées/sorties et l'immunité contre le bruit.

Structure d'un nanocluster Un nanocluster constitue un regroupement homogène ou quasi-homogène de nanoblocs qui forment une grille régulière. Les blocs logiques sont organisés de manière à ce que chacun d'entre eux soit connecté à ses quatre voisins les plus proches. La zone où les fils d'entrée et ceux de sortie se chevauchent constitue le

bloc de commutation de l'information d'un nanobloc à l'autre. Les nanoblocs situés à la périphérie d'un cluster sont connectés aux longues lignes qui entourent les nanoclusters. Les longues lignes sont utilisées pour la communication distante entre les nanoblocs ou les nanoclusters.

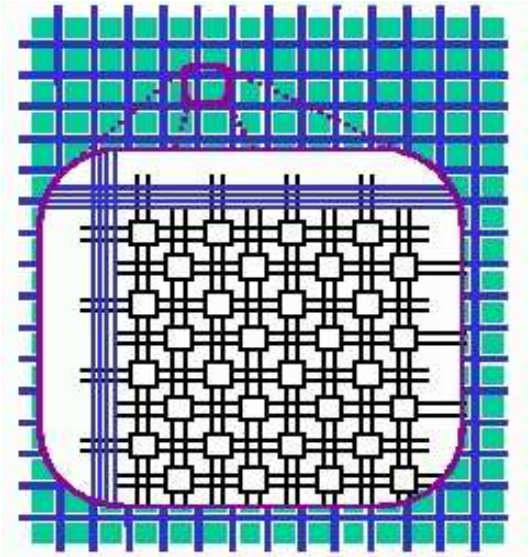


FIG. 2.12 – Représentation schématique de nanofabriques [36].

L'organisation des nanoclusters à travers les longues lignes se rapproche d'une conception connue : celle de l'architecture des FPGAs exploités pour leur flexibilité, l'implémentation des circuits et le prototypage rapide des circuits. C'est une structure qui favorise la mise en échelle de plusieurs manières : le nombre des composants augmente proportionnellement avec le nombre des longues lignes.

Cette architecture est tolérante aux fautes de fabrication de part sa régularité et sa reconfigurabilité. Lorsqu'une zone défectueuse est localisée sur la nanofabrique, un mapping de défauts est fait (une carte représentant l'architecture avec les zones défectueuses est établie au moyen de simulations électriques post fabrication) et la nanofabrique peut utiliser les régions intactes. La configuration d'une telle architecture est essentiellement locale pour chaque nanocluster.

2.2.2 Architecture stochastique : travaux de DeHon

DeHon propose une architecture primaire pour l'électronique moléculaire basée sur les nanodispositifs comme les nanotubes, les nanofils et les aiguilleurs moléculaires. Cette architecture peut garantir une fonctionnalité logique classique avec toute la logique et le mécanisme de restauration des signaux à l'échelle nanoscopique. Elle est tolérante aux fautes et compatible aux techniques de synthèse matérielle bottom-up émergentes [22]. Dans cette approche architecturale, les dispositifs nanoélectroniques actifs utilisés sont les

nanotransistors FET comme aiguilleurs logiques (dispositifs à trois bornes constitués de nanofils dopés ou nanotubes), ce qui confère à l'architecture une restauration des signaux due à la technologie FET.

2.2.2.1 Description

Le principe est de construire des architectures avec toutes les ressources nécessaires implémentant les calculs à l'échelle nanoscopique. L'assemblage des nanofils en matrices permet de former les nanotuiles programmables. Ces nanotuiles peuvent agir comme des mémoires ou circuits logiques (PLAs pour réaliser la logique). Elles peuvent permettre de réaliser les interconnexions pour router les signaux. La figure 2.13 illustre une telle conception de pavage des nanotuiles. Autour des nanotuiles, se trouvent les microfils supportés par une infrastructure lithographique fiable qui assure une mise sous tension de l'architecture, une alimentation en puissance des nanotuiles, une programmation et une évaluation de la logique de contrôle des grilles.

Logique La proposition suggère que les transistors FET peuvent être construits à l'échelle nanométrique. Ces transistors FET peuvent être utilisés pour concevoir les inverseurs aussi bien que les portes NAND et NOR. En effet, les PLAs peuvent être construits en utilisant deux plans logiques. Chaque plan logique se compose d'une matrice des dispositifs programmables (transistors FET) accompagnée d'une restauration des signaux. Chaque plan logique peut servir de plan NOR lorsque ses nanofils sont inversés. Ainsi le couplage de deux plans NOR donne une structure de PLA qui permet de réaliser les fonctions logiques. Le circuit de restauration des signaux est réalisé avec les transistors FET. L'architecture peut aussi utiliser des diodes pour stocker un état et fournir les blocs logiques reconfigurables.

L'architecture contient un système de décodeurs FET qui permet de conduire les dispositifs localisés aux intersections des nanofils dans les matrices. Un couple de décodeurs (un décodeur pour chaque dimension) est placé au sommet de deux des quatre côtés de la matrice de nanofils. Ils permettent à un grand nombre de nanofils d'être sélectionnés par un petit nombre de microfils d'adresses : un microfil peut contenir plusieurs zones dopées sur lesquelles les nanofils doivent êtres fixés. Les nanodispositifs fixent les intersections entre les nanofils et les microfils d'adresses, créant ainsi des systèmes d'adressage de ET câblés. L'adressage d'un nanofil consiste à conduire les microfils sur lesquels il est fixé. Les nanofils adressés minimisent l'effet d'une faute sur une ligne d'adresse : lorsque tous les microfils d'adresses connectés à un nanofil particulier sont alimentés, ce nanofil est sélectionné pour recevoir une source de tension. C'est l'interface de conduction nano-micro.

Interconnexion La taille des PLAs semble limitée par la longueur de nanofils. En effet, plus le nanofil est long plus il devient sensible à la cassure et à la courbure entraînant ainsi le non alignement et l'accroissement du taux de défauts. Les blocs de PLAs doivent avoir une taille modeste. Ils sont connectés de sorte que chaque bloc reçoit les entrées de

différents blocs de PLAs. Le bloc de PLA implémente la logique mais sert aussi de bloc d'aiguillage pour router les signaux. Les points de croisement programmables permettent de sélectionner les entrées qui participent à la logique ou au routage interne du bloc.

Entrées/sorties CMOS A l'image de FPGA symétrique, l'architecture présente des entrées/sorties de bord connectées aux nanofils des canaux de routage. Ces blocs d'entrées/-sorties à l'échelle lithographique sont fournis pour connecter la logique à l'échelle nanométrique aux microfils. Les nanofils servant d'entrées peuvent être conduits par les microfils. Un seul microfil peut connecter plusieurs nanofils car les microfils sont largement espacés.

Le principal défi se repose sur la conception d'une telle architecture qui peut être réalisée en utilisant les techniques d'assemblage bottom-up présentées dans la section 2.1.5. Ces techniques essentiellement stochastiques associées aux défauts d'alignement des nanofils engendrent un pourcentage des dispositifs inutilisables. Une stratégique sublithographique primaire pour fabriquer les matrices des PLAs est présentée dans [22]. Elle consiste à utiliser les techniques de Langmuir-Blodgett pour aligner les nanofils dans une direction et les transférer sur une surface de substrat lithographique préalablement préparée où l'espacement entre les nanofils est défini par la gaine d'oxyde [46, 94].

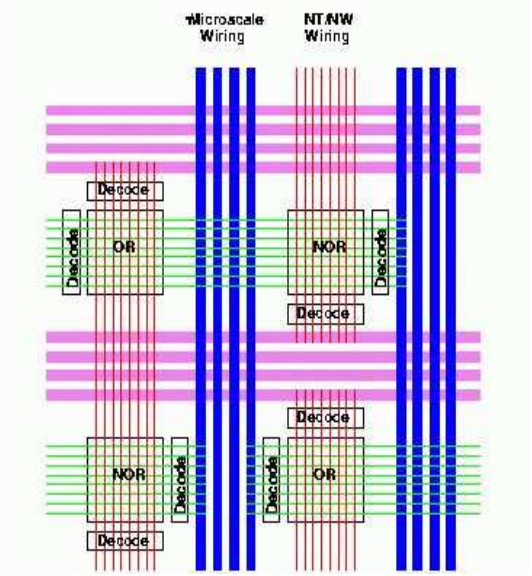


FIG. 2.13 – Interconnexion des nanoblocs fonctionnels [22].

2.2.2.2 Commentaire

C'est une architecture de nature hybride car elle intègre à la fois les dispositifs classiques et nanoélectroniques. Son style de conception semble être simple et quasi-régulier par son organisation topologique. Elle offre une large programmabilité puisque elle est

entièrement construite de matrices des nanofils dont les intersections abritent les jonctions programmables. Elle offre une large fonctionnalité logique et présente un mécanisme de restauration des signaux. Sa tolérance aux fautes est disponible par un supplément des dispositifs prévus dans toutes les matrices logiques. Elle permet leur configuration autour des éléments défectueux. Cependant, plusieurs défis restent en jeu pour fiabiliser complètement la conception finale de l'architecture, en particulier les problèmes d'interfaçage entre les systèmes nanoélectroniques et les systémes classiques, de fabrication des nanodispositifs et de tolérance aux fautes (malgré la taille modeste de tuiles envisageable et la redondance des systèmes).

2.2.3 NASICs : travaux de Moritz

Moritz et al. développent une nanofabrique hiérarchique et programmable basée sur les nanodispositifs [67, 68]. L'unité fondamentale est une grille de nanofils de silicium [27, 18] (ou de nanotubes de carbone [65, 82]) avec des jonctions agissant comme des diodes, des transistors FETs ou pouvant simplement être déconnectés : une nanotuile.

La conception présentée offre une mise à l'échelle pour construire des circuits très denses, complexes et pipelinés. Un mécanisme de stockage temporaire des données sur un nanofil, rythmé par une phase d'horloge, est incorporé. Les tuiles peuvent implémenter toute sorte de circuits logiques comme les additionneurs, les multiplexeurs et les bascules. Ces tuiles sont connectées entre elles au moyen de nanofils ou de microfils pour former une structure plus large. Les signaux d'entrée (respectivement de sortie) et leurs complémentaires sont disponibles dans la tuile car l'inversion des signaux impacte fortement sur la densité de la tuile. L'implémentation de la logique dans une tuile est basée sur la logique de PLA avec l'utilisation de buffers pour coupler les plans ET et OU (circuit logique sur les nanodispositifs destiné à adapter les sorties d'une implémentation logique aux entrées d'une autre implémentation dans une grille de nanofils). Ces buffers forment un plan de couplage qui entraîne une réduction de la densité dans la tuile (observable dans la conception par une absence de dispositifs actifs suivant les directions diagonales lors des couplages logiques). Cet effet constitue ainsi une contrainte à prendre en compte dans la conception d'un circuit complexe, en particulier pour le retour des signaux et le routage des signaux dans différentes directions. Une telle contrainte peut avoir une forte influence dans la conception des architectures lorsque les techniques de tolérance aux fautes ne sont pas basées sur la reconfiguration autour des dispositifs défectueux.

Pour montrer la possibilité de mettre en œuvre une architecture applicative sur une nanofabrique, cette équipe a développé un simple processeur WISP (WIre Streaming Processor) en utilisant les NASICs [68]. Nous donnerons plus de détails de cette conception architecturale à la section 2.3 dans lequel cet exemple pilote de mise en œuvre est illustré.

2.3 Technologie retenue : NASICs

De toutes les technologies présentées, qui ne constituent pas une liste exhaustive mais présentent les grands axes de recherche investigués, nous avons choisi de nous orienter dans l'exploration de la technologie NASIC (Nanoscale Application Specific Integrated circuit). Cette orientation découle essentiellement d'un projet de coopération, plutôt que des caractéristiques favorables d'une technologie par rapport à une autre d'autant plus que chacune fait valoir ses atouts par rapport à une réalisation physique. L'objectif est de mettre en commun les expertises (outil Madeo et technologie NASIC) pour favoriser la réalisation physique des supports nanotechnologiques avec des performances fiables. De plus, les architectures pour ces technologies sont identiques (même structuration). Se restreindre à une technologie n'exclut en rien la généralisation dans le traitement car l'outil à utiliser pour la mise en œuvre des modèles architecturaux est adaptable.

Dans cette section, nous présentons les fondements du concept des NASICs. Nous décrivons les différentes approches de cette technologie d'implémentation. Ensuite nous explorons un exemple de mise en œuvre.

2.3.1 Style d'implémentation logique

Cette section présente deux styles d'implémentation logique à travers un exemple simple de fonction logique ET avec deux entrées. La plus utilisée aujourd'hui pour réaliser les circuits intégrés est la logique CMOS. La logique préchargée, longtemps déclassée, devient un modèle de grille proposé pour la nanoélectronique. La comparaison de ces deux styles de logique permet de comprendre la préférence de chacun au cours de la migration technologique (aujourd'hui et demain). Loin de reportorier les règles de conception qu'on peut trouver dans [64], nous adressons le modèle de dessin physique d'un circuit intégré dans les deux les implémentations. Ce modèle est une grille topologique basée sur ces règles. La table de vérité permet de représenter le dessin physique.

2.3.1.1 Logique CMOS

La logique basée sur la technologique CMOS repose le couplage de deux structures complémentaires (pullup et pulldown) avec des transistors comme interrupteurs. La structure pullup est un réseaux de transistors de type p qui connecte la sortie d'un circuit à la source de tension lorsque celle-ci est au niveau haut. De même, la structure pulldown, formée de transistors de type n, connecte la sortie du circuit à la masse quand celle-ci est au niveau bas. Tout circuit intégré peut être réalisé en reliant les transistors en série dans une structure pulldown et en parallèle dans une structure pullup (vice versa). La figure 2.14 montre la mis en œuvre de la porte ET. La sortie de la porte est au niveau bas si les deux entrées a et b sont au niveau bas ou si l'une de ces deux entrées est au niveau bas ; cela est mis en évidence dans la figure par la connexion de deux étages de pulldown à la masse (partie gauche du schéma). Cette sortie est fixée au niveau haut si les deux entrées sont au niveau haut, ce qui correspond au fait que le second étage de pullup soit lié à la source d'alimentation.

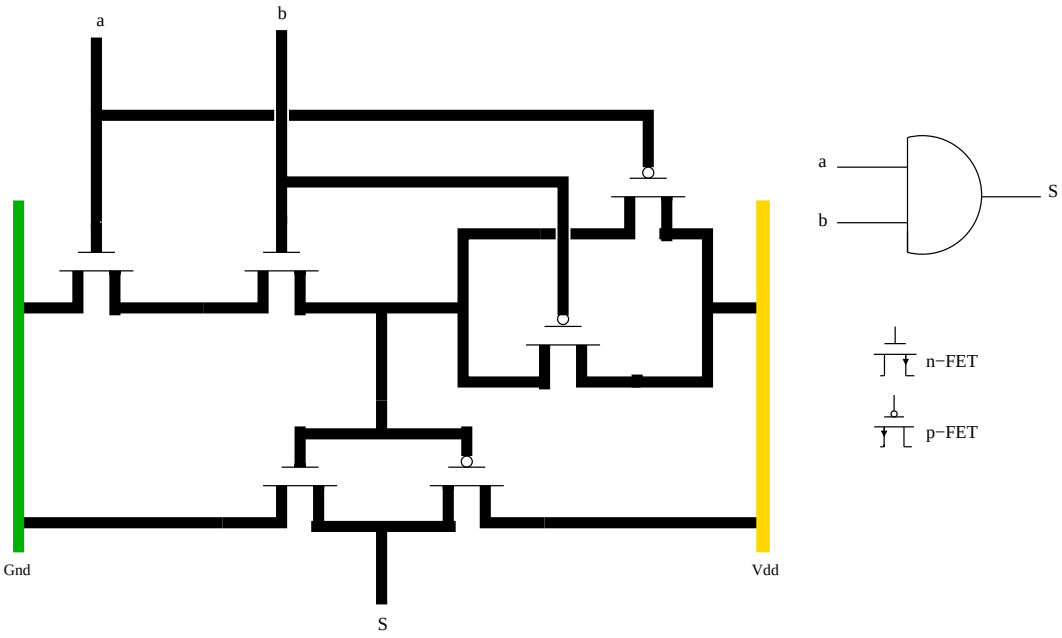


FIG. 2.14 – Implémentation basée sur la logique CMOS.

Cette technologie est à l'heure actuelle celle qui présente l'une des grandes densités d'intégration disponibles. Deux avancées majeures ont permis d'atteindre les performances nécessaires et suffisantes à la réalisation des puces fiables. La première est la réduction de la consommation des circuits intégrés CMOS. La seconde est de concevoir et de réaliser des circuits mixtes monolithiques en technologie CMOS bon marché aussi bien à haute fréquence (13 MHz) qu'a ultra haute fréquence (2 GHz). Le choix universelle de cette logique CMOS est aussi motivé par la connaissance de la technologie, les avantages de performance par rapport aux technologies concurrentes, la maîtrise de la chaîne de fabrication, etc.

2.3.1.2 Logique préchargée

C'est une logique d'implémentation où la capacité d'un fil est utilisée pour remplacer le pullup dans un sens. Elle utilise deux phases d'horloge pour fonctionner. La première phase autorise l'envoi d'une charge dans le fil sans se soucier de l'endroit où elle sera dirigée. La seconde phase permet créer un chemin de la masse au fil (localisé entre la masse et le fil) qui résulte de la valeur de la structure pulldown. Cette technique d'implémentation logique accélère la propagation des signaux le long d'un fil. En réalité, l'établissement d'un chemin entre la masse et le fil favorise la décharge rapide de celui-ci puisque le courant doit passer à travers la faible résistance de la structure pullup. Il apparait que le temps de préchargement des signaux dans les fils est très court. Ce temps est fonction de dimensions de transistors de la structure pulldwon (largeur et longueur du canal d'un transistor). Par ailleur, un autre bénéfice est que l'usage du courant est limité à la valeur affectée pour précharger un fil à chaque cycle d'horloge.

La principale raison pour laquelle la logique préchargée n'est pas souvent utilisée est que la technique se fonde sur la présence d'une grande capacité pour bien fonctionner. En effet lorsque la fil de stockage est petit, le chemin peut ne pas établi et la charge stockée peut se disperser dans les canaux des transistors voisins de la capacité. Cette perte de charge n'est pas recompensée. La tension sur le fil s'avère considérable réduit au point de rendre difficile la lecture de la valeur stockée dans la capacité [91]. Par conséquent, l'usage de cette implémentation logique implique le besoin d'avoir des grands fils, coûteux en surface comme condensateurs pour transporter les signaux dans toutes les directions d'une puce.

Pour illustrer cette technique d'implémentation logique, nous préférons la faire valoir dans le cadre de la conception NASIC où le modèle de grille reste le même mais les fils lithograghiques sont remplacés par les nanofils. Nous considérons toujours l'exemple de la porte ET. Pour obtenir son dessin physique, nous nous servons de la table de vérité. De cette table, tous les termes produits de la fonction sont réalisés avec les dispositifs de type p (nanofils et transistors de type p). Les produits fournissant une même valeur de la fonction sont regroupés ensemble pour former une sortie à l'aide des dispositifs de type n (nanofils et transistors de type n). Par exemple, le terme ab est le seul qui fixe la fonction S à 1. Il est donc isoler des autres qui représentent le complémentaire de la fonction.

La structure étant grille, elle se constitue de deux plans parallèles de nanofils (ou nanotubes). Chaque plan contient les nanofils parallèles de même type. Dans la figure 2.15, Les nanofils de couleur rouge et bleue distinguent les deux plans et marquent la différence de type (p-Nanofil et n-Nanofil). Les nanofils (rouges) d'un plan croisent ceux (bleus) de l'autre. La region de croisement peut contenir un nanodispositif (transistor). Chaque entrée d'un circuit est représentée avec deux nanofils (un fil pour la valeur 1 de l'entrée et l'autre pour son complémentaiire). Les signaux produits sont aussi présentés dans les deux formes (une des différence avec l'abstraction classique). Dans l'exemple de la porte ET, les termes produits sont réalisés avec les dispositifs de type p (p-FET localisés à gauche de la grille). Les termes produisant la même valeur de la fonction couplés avec les dispositifs de type n pour former les sorties du circuit (n-FET localisés à droite). Autour de la logique de transistors (termes produits et termes sommes) se localisent les pullups et les pulldowns qui ramenent une tension à la sortie d'un fil. Chaque nanofil rouge est connecté à une source de tension Vdd et à la masse Gnd. Les nanofils d'entrée (respectivement ceux de sortie) sont connectés à la masse (respectivement à la source d'alimentation).

2.3.1.3 Implémentation logique pour la nanoélectronique

La complexité d'un circuit intégré est un facteur important pour la qualité de la conception du circuit. La nécessité de gérer cette complexité nous pousse à préferer les méthodes logiques qui permettent d'exploiter les structures régulières. Autrement, la répétition d'une structure simple peut considérablement réduire la complexité d'un circuit. Par exemple, l'implémentation de la porte ET avec la technologie CMOS montre que un réseau de 2 transistors (inverseur) représente la structure plus simple. Cette struc-

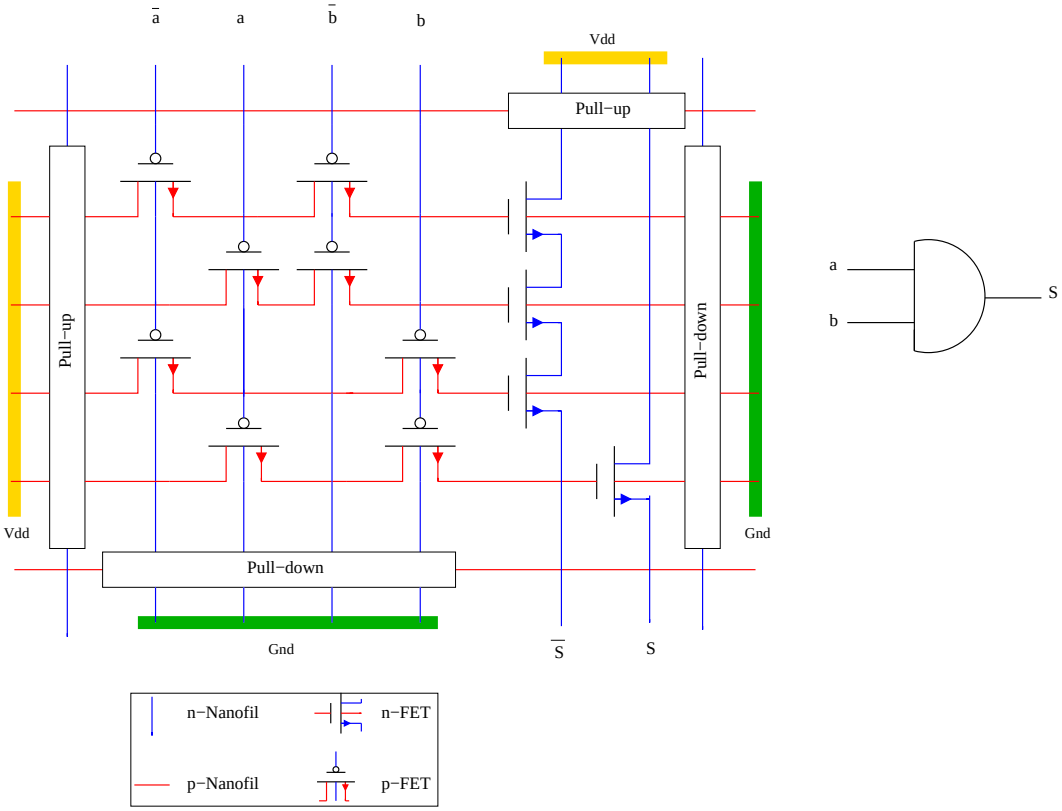


FIG. 2.15 – Implémentation de la fonction S = ab.

ture simple est répétée 3 fois dans pour construire la porte dans la disposition présentée. L'implémentation en technologie CMOS repose sur la structure de l'inverseur.

Dans le cadre de l'implémentation avec une logique préchargée, la structure la plus simple est le transistor qui permet de composer des structures linéaires (transistors placés en série) nécessaires pour réaliser le modèle en grille. La disposition topologique de ces dernières est facile à réaliser par rapport à celle basée sur l'inverseur. Cela explique aussi l'existence des topologies arbitraires d'implémentation pour certains circuits intégrés complexes en logique CMOS. Par ailleurs, la composition d'une conception basée sur des structures simples et identiques facilite la compréhension de la topologie géométrique. En d'autres termes, les modèles de grille réguliers avec une interconnexion structurée offrent un espace réduit entre les ressources de routage. Dans ce cas, les puces deviennent ainsi plus petites et plus rapides.

La logique préchargée favorise la réalisation des structures complexes régulières. Plus une structure est régulière, plus le remplissage de cellules (dépot d'un transistor dans chaque jonction de la grille) est dense. Avec le progrès technologique, plusieurs dispositifs de taille nanométriques seront disponibles. Pour les dimensions minimales de dispositifs envisageables, le choix entre les technologies d'implémentation logique en concurrence doit se faire sur la base des propriétées topologiques de leurs interconnexions, de la richesse fonctionnelle des circuits de base qu'elles proposent, de la simplicité du procédé tech-

nologique, de la performance relative à la consommation, de l'adéquation à l'intégration d'un système complet et de la performance relative à la densité d'intégration. Ainsi, pour les dimensions nanométriques auxquelles les dispositifs doivent être réalisés, la logique CMOS se trouve limitée par l'ensemble de contraintes physiques. La logique préchargé doit permettre de réaliser des conceptions plus denses avec une bonne utilisation de la fabrique.

2.3.2 Circuits NASICs

2.3.2.1 Conception statique

Description La conception des NASICs dérive de la structure crossbar. C'est une structure de nanodispositifs organisés en matrice de nanotubes de carbone ou nanofils de silicium. La figure 2.16 illustre la mise en œuvre d'un additionneur complet 1-bit dans une nanotuile. Au centre de la tuile se trouve une nanogrille dont les jonctions sont programmées pour réaliser la logique. Cette nanogrille présente deux plans NOR couplés par des buffers NFETs utilisés pour orienter les signaux vers la sortie. Le premier plan NOR basé sur les transistors PFETs représente les termes produits (c'est la partie de la figure marquée par les points noires) et le second plan NOR génère la somme des termes produits (partie marquée par les points blancs). Les deux plans forment alors un modèle logique de type PLA. Les fils épais de couleur qui entourent la nanogrille représentent les microfils servant à assurer les communications entre l'interface externe et les nanogrilles. Les parties imprimées en noire dans la nanogrilles représentent les pullups/pulldwons utilisés pour ramener la tension de sortie des nanofils au niveau haut ou bas. Tout signal est représenté par deux nanofils : le signal original et son complémentaire.

Cette nanofabrique présente une limitation topologique : une grande partie du circuit ne peut contenir des dispositifs actifs. C'est l'effet dit diagonal qui occasionne une perte de la densité selon la direction diagonale (surface sans les points blancs ou noires). Une amélioration de la densité dans une fabrique est possible en remplaçant les buffers FETs du second plan NOR (plan représentant la somme des termes produits) par un plan logique des diodes. La figure 2.17 illustre un exemple d'optimisation de la densité de la fabrique.

Circuits séquentiels La conception devient nettement plus complexe quand il s'agit de réaliser les circuits avec un retour d'information ou des chemins de donnés. Elle présente alors des défis en incorporant des circuits séquentiels dans une nanofabrique de NASICs. La figure 2.18 illustre une nanotuile implémentant un flip-flop pour stocker localement un bit dans un plan de calcul. Le circuit révèle une faible utilisation de la surface due au retour de l'information réalisé par 3 blocs non inverseurs qui bouclent certains signaux à l'entrée. Il met en évidence une surface inoccupée par les nanodispositifs. Cela diminue le rendement de la surface pour réaliser des circuits séquentiels complexes.

Un exemple de réalisation d'un simple chemin de données qui combine l'additionneur optimisé et le flip-flop montre la persistance de l'effet diagonale et la réduction de la densité effective de la conception [68].

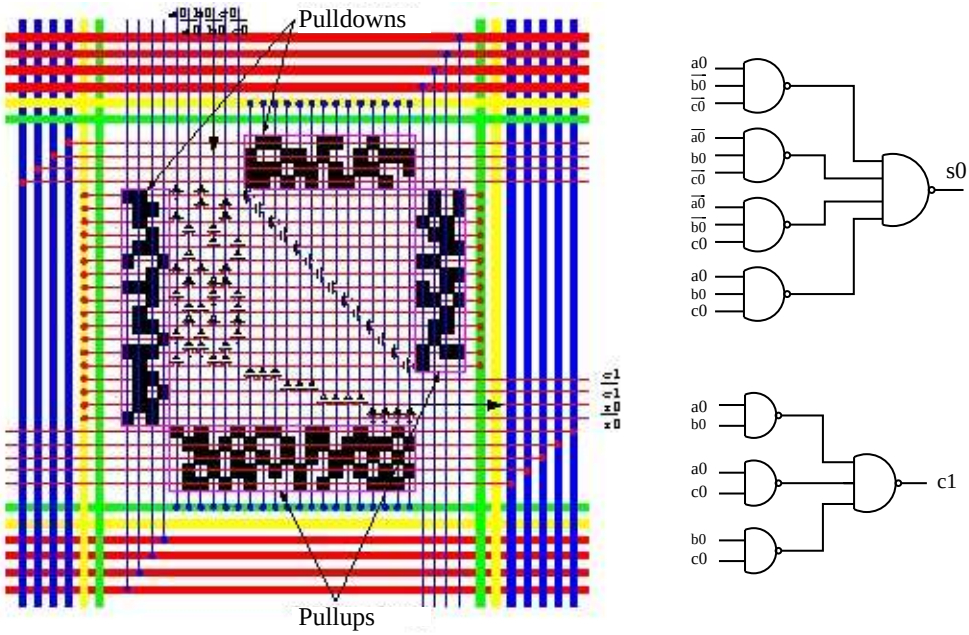


FIG. 2.16 – Nanotuiles réalisant un additionneur 1-bit [67] *(les fils épais sont les microfils et les fils fins sont les nanofils. Perte de densité dans la tuile suivant la diagonale).*

La construction des circuits séquentiels dans une nanogrille est difficile à réaliser à cause de la faible densité effective des bascules. Tous ces circuits statiques présentent plusieurs problèmes. Ils exigent un dimensionnement soigneux des dispositifs pour une logique correcte qui fixe les contraintes supplémentaires sur la fabrication. De plus, la puissance d'énergie due aux courants des chemins de données pourraient être un facteur limitant de la mise en échelle des nanotuiles.

La solution envisagée pour résoudre ou du moins alléger certains inconvénients imposés par la logique statique est d'utiliser les circuits dynamiquement cascadés.

2.3.2.2 Conception dynamique

La conception dynamique des NASICs procure des atouts essentiels. Elle impose d'avoir un seul type de dopage dans une dimension et permet aux nanofils de se comporter comme des registres sans utilisation explicite de bascules. Le mécanisme de stockage temporaire sur un fil (the latching on the nanowire) qui intervient au tour du calcul isole les étages d'exécution d'un circuit. Cette conception logique permet d'obtenir un modèle de circuits pipelinés et impose une marge au bruit pour les performances.

Faisabilité physique Le fonctionnement de la logique est basé sur le stockage d'une charge sur un fil. la charge (ou la décharge) conditionnelle du fil est fonction des entrées. Un nanofil dans une grille en logique dynamique est donc constitué d'un réseau de transistors FETs pour réaliser une fonction logique et de deux transistors , de types opposés et situés

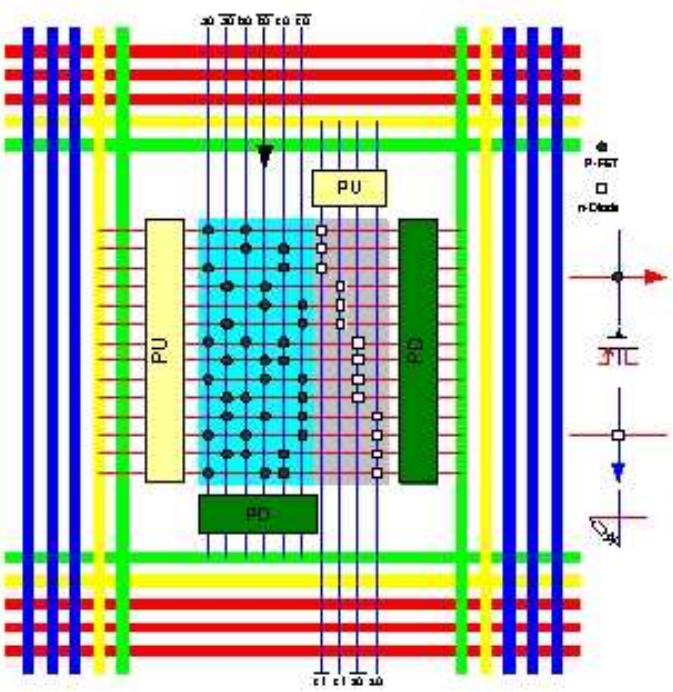


FIG. 2.17 – Optimisation par l'usage de la logique de diodes [67].

à chaque extrémité du fil, pour les phases de précharge et d'évaluation. Ces deux derniers transistors sont commandés par une horloge. Pendant la phase de précharge correspondant à une phase d'horloge au niveau bas, la sortie logique du fil est préchargée. Dans la phase d'évaluation marquée par une phase d'horloge au niveau haut, la sortie du fil est déchargée en fonction des entrées logiques.

La figure 2.19(a) montre la mise en œuvre d'une telle logique dans une tuile, en utilisant la logique OR-NAND. Le plan OR correspond à la partie A et le plan NAND est indiqué par la partie B. Les deux parties utilisent une séquence de précharge/évaluation pour générer la sortie en formant un pipeline de deux étages. Un aspect significatif est la présence d'une troisième phase de maintien qui permet de stocker temporairement les valeurs de sorties entre les circuits.

Le style logique présenté dont l'avantage majeur est d'avoir des faibles capacités au niveau des entrées et des sorties, donc des temps d'évaluation courts, est physiquement faisable. Les critères de fonctionnement logique d'un circuit sur ce principe sont suffisamment réunis pour envisager la réalisation des applications importantes.

La figure 2.19(b) montre le séquencement de deux circuits combinatoires A et B dont les fonctionnement logiques sont gérés par phases dynamiques (précharge, évaluation et maintien). Pour le circuit A, la phase de maintien nécessite la précharge des circuits A et B pour passer au niveau logique haut. Dans la phase de précharge, la valeur de sortie $outA$ est préchargée à la source haute. Dans la phase d'évaluation, le circuit se décharge selon la logique des entrées inA. Au cour de la phase de maintien, la valeur de la sortie

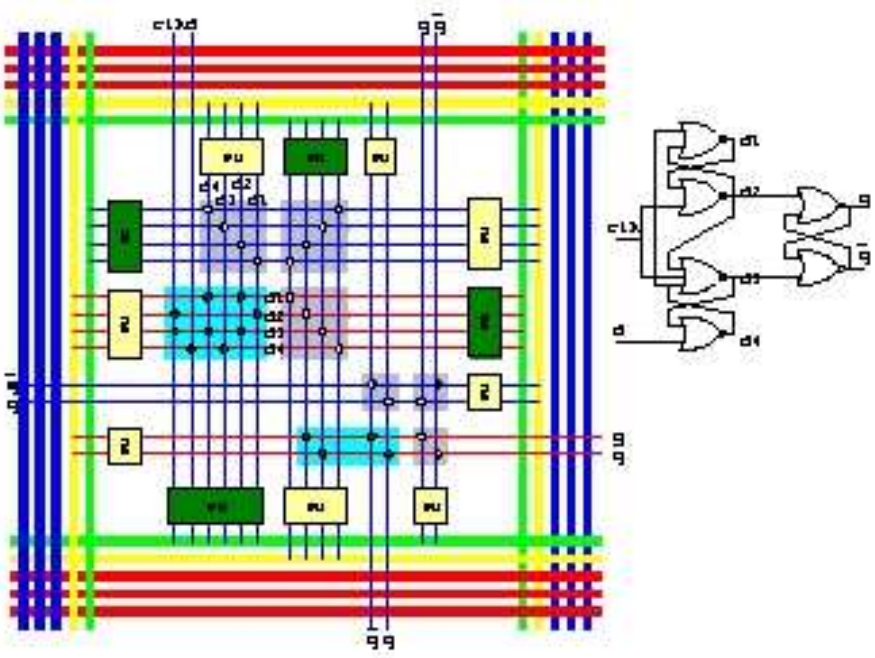


FIG. 2.18 – Conception d'une bascule 1-bit dans une nanotuile [67] *(le circuit fournit localement une petite mémorisation. Dans la nanotuile, le retour d'information est réalisé par 3 blocs NFETs non inverseurs qui renvoient les signaux d_1, d_2, d_3 et d_4 à l'entrée du flip-flop).*

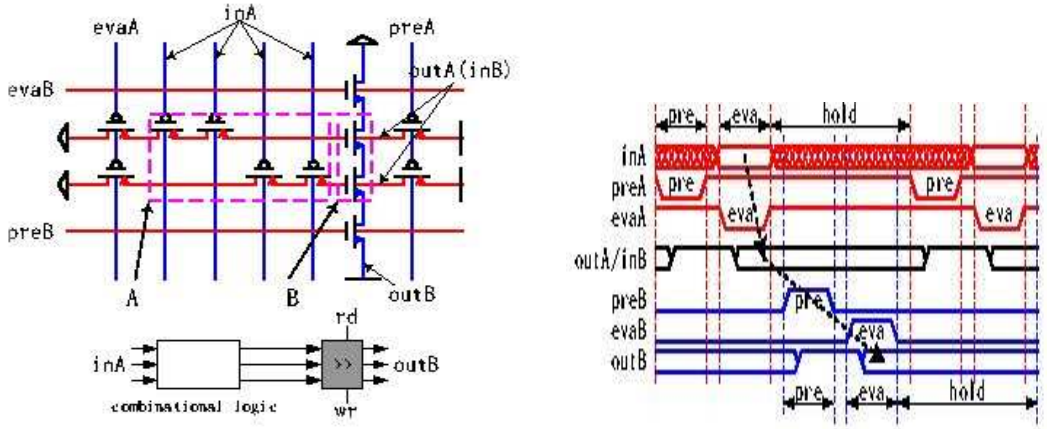
outA est conservée et le *preA* et le *evaA* sont ouverts.

Une possible perte de la valeur stockée dans un nano-verrou due au courant de fuite peut être observée si un circuit reste trop longtemps dans la phase de maintien.

2.3.3 Exemple d'utilisation des NASICs : le WISP

Dans le but d'appliquer les techniques développées dans le cadre des NASICs, l'équipe de Moritz a conçu un processeur de type stream sur une nanofabrique. Ce premier prototype nommé WISP-0 sert d'exemple pilote pour optimiser le modèle des circuits NASIC avec le souci de préserver l'avantage de densité d'intégration offerte par les nanodispositifs. Quelques hypothèses de simplification sont émises dans la conception pour transférer les données à travers la fabrique : un minimum de chemins de contrôle et de retour d'information. Ils induisent des nouveaux mécanismes qui permettent de :

1. mémoriser temporairement les valeurs produites pendant un traitement dans les nanofils de façon à éviter l'utilisation des bascules (source de perte de densité).
2. supprimer les dépendances à l'exécution : cela permet d'augmenter significativement la densité effective en réduisant les chemins de retour de l'information et la logique de contrôle.



(a) Circuit dynamique et nano-verrou. (b) Chronogramme du circuit dynamique.

FIG. 2.19 – Principe de réalisation et de fonctionnement de circuits dynamiques dans une nanotuile [67].

2.3.3.1 Architecture du WISP-0

Le WISP-0 est une architecture pipeline à 5 étages comme l'illustre la figure 2.20(a) : un compteur, une ROM, un décodeur, un banc de registres et une ALU. Chaque étage est représenté par une nanotuile. Les nanotuiles adjacentes sont connectées entre elles par des nanofils. L'ensemble des nanotuiles est ainsi dynamiquement cascadé pour former le pipeline. La figure 2.20(b) montre le floorplan du tracé physique du WISP-0.

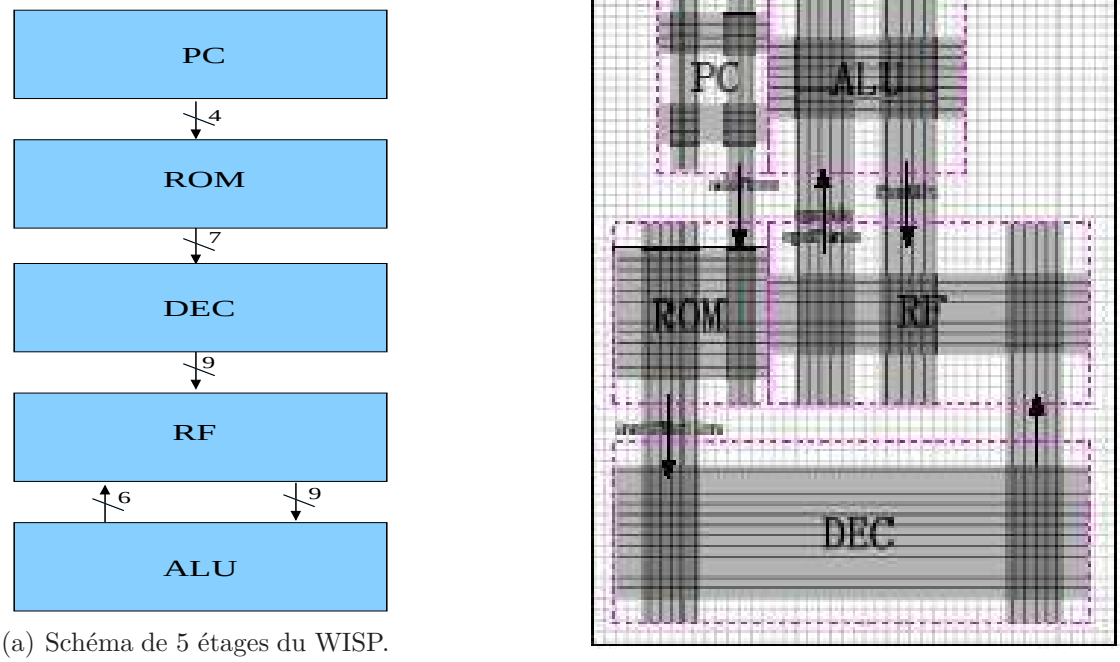
La couche ISA Le jeu d'instructions est composé des 5 instructions suivantes : #(nop mov movi add mult). Elles sont codées sur 3 bits :

- 000 : nop
- 001 : movi
- 010 : mov
- 011 : add
- 100 : mult

Les opérandes ont pour taille 2 bits, on obtient des instructions codées sur 7 bits stockées dans la ROM

opcode[3*bits*]	operand$_1$[*bits*]	operand$_2$[2*bits*]

TAB. 2.1 – Codage des instructions



(a) Schéma de 5 étages du WISP.

(b) Floorplan du WISP.

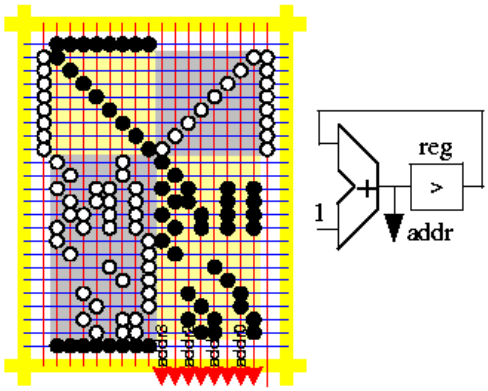
FIG. 2.20 – Pipeline de 5 étages du WISP [68].

2.3.3.2 Composants

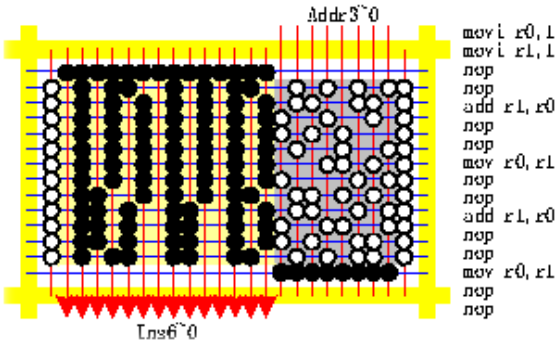
Compteur d'instruction (PC) Le bloc PC implémente un compteur d'instructions. Il génère une adresse codée sur 4 bits. La figure 2.21(a) représente la mise en œuvre du compteur de programme sur une nanotuile. Il a deux modules logiques : un circuit d'incrémentation situé dans la partie inférieure et la bascule dans la partie supérieure (les points blancs représentent les transistors PFET et les noirs les transistors NFET). La sortie du compteur est retardée dans chaque cycle et retournée au compteur au cycle suivant. L'adresse est alors incrémentée dans chaque cycle : on a donc $PC = (PC + 1) Modulo 16$.

ROM Le bloc ROM est une mémoire ROM d'instructions. Cette mémoire contient le programme qui doit être exécuté. Ce bloc vérifie une certaine instruction selon l'adresse provenant du compteur de programme puis l'instruction est acheminée au bloc décodeur. La figure 2.21(b) montre un exemple de ROM d'instructions avec les codes stockés. Cette mémoire ne peut pas être modifiée en cours d'exécution. Le bus d'adresse de la ROM a une largeur de 4 et la mémoire peut donc contenir jusqu'à $2^4 = 16$ instructions.

Décodeur (DEC) Le décodeur reçoit les instructions codées sur 7 bits lues dans la ROM à l'adresse donnée par le registre PC. Il permet de coder une instruction en un code opération et deux opérandes. la figure 2.22(a) illustre son implémentation dans une



(a) Tracé physique d'un compteur de programme.

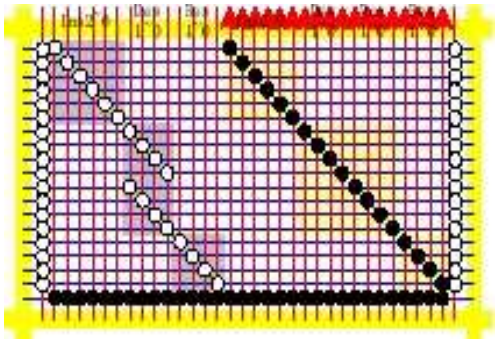


(b) Tracé topologique d'une ROM et exemple de code du programme stocké.

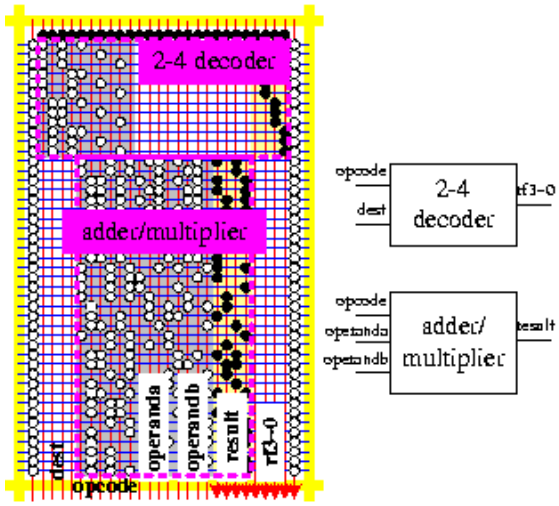
FIG. 2.21 – Implémentations NASICs [68].

nanotuile NASIC.

Unité Arithmétique et Logique (ALU) L'ALU se compose d'un additionneur et d'un multiplieur, ainsi que d'un décodeur $2 \rightarrow 4$. La figure 2.22(b) représente le tracé de l'ALU. Il reçoit les entrées provenant du fichier des registres (opcode, opreanda et operandb) et produit le résultat de retour d'écriture. L'adresse de retour d'écriture est décodée par le décodeur $2 \rightarrow 4$ au même instant . Le résultat et l'adresse sont alors transmis dans la file des registres où l'écriture du résultat sera effectuée au prochain cycle.



(a) Décodeur.



(b) L'unité arithmétique et logique.

FIG. 2.22 – Tracés physiques NASICs [68].

Banc de registres (RF) WISP-0 possède des registres de taille 2 bits. Il utilise deux multiplexeurs de type $4 \rightarrow 1$ et un autre de type $2 \rightarrow 1$. Lorsque le code opération et les opérandes accèdent aux registres, les valeurs des opérandes sont lues et envoyées dans l'ALU. la figure 2.23 correspond au tracé physique des registres. Les données sont mémorisées sur les $4 \times 2 \times 2$ nanofils horizontaux situés en bas de la figure. Les valeurs des registres sont alors sélectionnés par un multiplexeur $4 \rightarrow 1$. un autre multiplexeur $4 \rightarrow 1$ est utilisé pour choisir entre les valeurs immédiates et les valeurs des registres. Au même moment, le code opération et l'adresse des registres de destination sont pipelinés à l'ALU. Lorsque l'ALU souhaite écrire le retour des résultats au fichier des registres, les données et les adresses passent par le coin gauche en haut de la nanotuile.

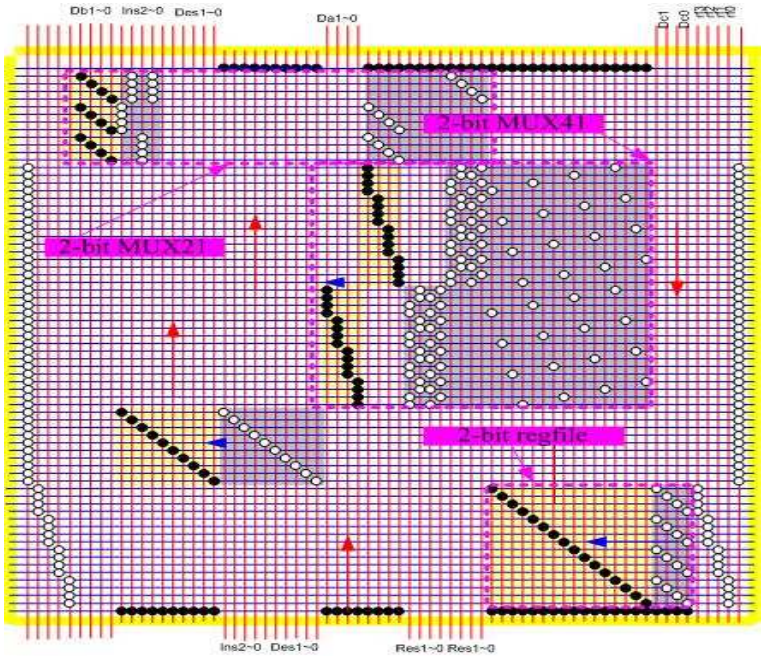


FIG. 2.23 – Registres [68].

2.4 Vers les architectures de traitement : un FPGA sur les QCAs

Classiquement, un FPGA est une collection de blocs logiques qui sont arrangés dans un cadre d'interconnexion. Les signaux sont envoyés dans ces blocs logiques à travers les ressources de routages disposés (verticalement et horizontalement) entre ces derniers. La nature programmable des connexions dans la matrice de routage permet d'orienter les signaux dans toutes les directions possibles. L'implémentation d'un FPGA pour les QCAs nécessite de prendre en compte trois éléments de base suivants : les cellules logiques, l'interconnexion et l'organisation structurelle des cellules logiques.

La nature d'horloge de QCA est complexe. L'horloge change de phases quand la barrières potentielles affectant un groupe de cellules QCA passent à travers quatre phases d'horloge(Switch, Relax, Release et Hold), ce qui causerait un problème pour les fils d'interconnexion alimentant les blocs logiques. L'information transmise sur un long fil serait pipelinée et sa transmission ne serait pas spontanée. La solution à ce problème est de considérer la phase Relax d'horloge qui a l'effet d'extraire un groupe de cellules dans une disposition donnée. Dans ce cas l'horloge est utilisée pour arrêter les groupes de cellules QCA afin de créer les commutateurs. C'est ce même mécanisme qui pourrait être utilisé pour programmer les cellules de routage ou la logique. Cette technique permettrait de développer le mécanisme d'interconnexion d'un FPGA en QCA.

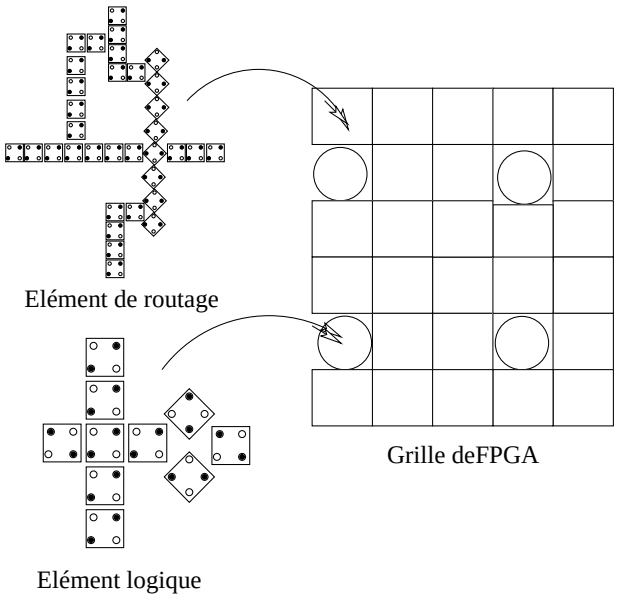


FIG. 2.24 – Routage et blocs logiques combinés pour former un FPGA.

2.4.1 Blocs logiques

Les blocs logiques peuvent être facilement implémentés à l'aide des portes majorités qui peuvent agir comme de portes AND ou OR. Ces deux portes peuvent suffire pour implémenter les équations logiques de somme de produits ou de produit de sommes. Cette logique de AND et OR n'est pas fonctionnellement complète car elle ne peut être programmée pour implémenter un inverseur qui est nécessaire pour les circuits complexes. Un multiplexeur peut être utilisé pour sélectionner une entrée inversée.

2.4.2 Interconnexion

Un réseau de routage de FPGA peut être vu comme un ensemble de connexions. La programmation du réseau de routage par une horloge permet à plusieurs éléments

de routages d'être touchés dans une large zone d'horloge, ce qui permet au signal de se propager à travers ces éléments dans un cycle. Pour router les signaux dans toutes les directions sans interférences, l'élément basique de routage peut être répandu à travers une région élémentaire d'horloge. Cela permet d'offrir une flexibilité dans le routage. Les éléments de routage peuvent être connectés avec les éléments logiques pour former un FPGA complet.

2.4.3 Organisation structurelle

Une fois que les blocs logiques et le réseau d'interconnexion sont établis, l'ensemble peut être programmé pour structurer un FPGA. La partie fondamentale d'un microprocesseur est un ALU qui peut utiliser un Full adder. La logique du Full adder est convertie en logique NAND qui peut être placée dans un FPGA. Dans la figure 2.24, les cercles représentent les blocs logiques et les carrés représentent les ressources de routage dans différentes zones d'horloge. Les ressources de routages sont multiples de quatre carrés pour correspondre aux quatre phases d'une horloge de QCA. Le circuit présenté est simple et régulier.

2.5 Conclusion

Dans ce chapitre, nous avons présenté certaines technologies nanoélectroniques émergentes et leurs procédés de fabrication. Elles offrent beaucoup de possibilités pour améliorer les performances et l'efficacité des systèmes classiques de traitement de l'information. Chacune d'entre elles présente des spécificités dues à sa nature. Pour tirer profit des avantages de ces nanotechnologies, certaines pistes architecturales sont proposées pour ces technologies. La conception de telles propositions architecturales répond aux problèmes spécifiques à la nature nanoscopique des nanodispositifs envisagés.

Nous avons également présenté une conception physique des dispositifs, développée par l'équipe de Moritz. Cette conception est au coeur des travaux qui vont suivre. L'exemple de la mise en œuvre du processeur de type stream montre qu'il est possible d'implémenter des architectures applicatives sur les supports physiques nanotechnologiques. Cela ouvre des nouvelles voies d'investigation : reconstruire les architectures reconfigurables et proposer des outils prospectifs pour valider les supports architecturaux basés sur les nanotechnologies.

Chapitre 3

Madeo pour la synthèse d'architectures sur NASICs

Face à une situation technologique versatile, telle que décrite dans le chapitre précédent, la rapidité de prototypage est capitale. L'objectif est donc d'offrir une capacité de prospection architecturale pour l'exploration rapide des architectures réalisées à base de technologies nouvelles.

La réalisation physique d'architectures résulte de la synthèse VLSI : la manière d'assembler des éléments d'une chaîne dans une topologie donnée. L'émergence technologique permet le développement des nouvelles conceptions architecturales. Disposer des outils de synthèse contribue à maîtriser la réalisation de ces architectures. La synthèse d'architectures est aussi motivée par la gestion de la complexité des circuits NASICs qui peut finalement aider à apprivoiser celle des architectectures reconfigurables basées sur cette technologie NASIC.

Le chapitre s'articule en deux parties. La première adresse l'extension de l'outil Madeo (section 3.1) afin de proposer une mise en œuvre des supports architecturaux sur un socle technologique prospectif. La seconde partie, qui regroupe les deux dernières sections du chapitre, propose un exemple de mise en œuvre automatique d'une architecture applicative. Pour s'assurer de la validité des outils étendus, une implémentation produite du WISP-0 sous Madeo est présentée. Les tracés des nanotuiles générés automatiquement sont alors comparés avec les tracés manuels.

Ce travail vise à démontrer la viabilité d'une approche précoce de CAO.

3.1 Adaptation de Madeo aux nanofabriques

L'importance et la nécessité d'outils génériques pour les nouvelles architectures telles que les NASICs ont été longuement justifiées dans [25, 75] : estimation de performances, simulation des architectures, évaluation des modes de programmation sont les principales raisons pour lesquelles Madeo est utilisé pour modéliser les architectures pour les nouvelles technologies. Dans le chapitre 1, nous avons présenté Madeo comme un environnement de programmation pour les architectures reconfigurables. La modélisation des architectures

reconfigurables et celles des nanofabriques présentent des caractéristiques communes, dues à leurs structures. En effet, dans les deux cas, elles présentent une architecture régulière, organisée hiérarchiquement en matrices de blocs identiques.

Une extension du cadre de Madeo intervient suivant deux directions : au niveau bas pour la prise en charge technologique et au niveau haut pour la description des traitements à implémenter.

3.1.1 Description de l'architecture et outils de synthèse physique

La description d'une nano-architecture doit suivre la même démarche que celle d'une architecture de FPGA classique. La grammaire doit donc être adaptée ainsi que le modèle abstrait d'architecture. L'adaptation des outils pour adresser les contraintes spécifiques aux supports nanotechnologiques et les règles pour implémenter la logique (PLAs structurés) exigent une extension du modèle abstrait. Dans la couche basse, le modèle permettant la représentation du matériel est alors étendu tandis que de nouveaux outils de programmation sont introduits.

3.1.1.1 Extension du modèle initial

Les modèles d'architectures de Madeo s'articulent en deux niveaux. L'un, qualifié de méta modèle abstrait, décrit les éléments architecturaux et les mécanismes de structuration autorisés. L'autre, qualifié de méta modèle concret, représente une architecture donnée. Ce dernier est manipulé par les outils qui découvrent le méta modèle via une API (Application Programming Interface) fournie dans le méta modèle abstrait.

Le méta modèle concret apparaît comme une combinaison et une spécification d'éléments du méta modèle abstrait. En d'autres termes, la modélisation d'une nouvelle architecture se traduit généralement par une extension quantitative du méta modèle abstrait de Madeo. C'est à dire par un ajout d'éléments concrets. En revanche, la prise en compte d'éléments architecturaux, différents en nature, se traduit par une extension qualitative. C'est à dire par ajout de briques de base au méta modèle abstrait. Adresser les ressources NASICs a conduit à une extension qualitative (nanogrille, nanotransistors PFET et NFET, etc) laquelle est validée par des extensions quantitatives (tuiles de différentes tailles).

La conception des objets NASICs se différencie de celle des fabriques classiques sur des aspects essentiellement technologiques. Une cellule dans une nanofabrique s'identifie à une grille de nanofils orthogonaux dont les intersections contiennent des nanodispositifs. Il convient de créer de nouvelles classes qui caractérisent les objets ou les aspects spécifiques d'une telle structure. Cette extension contribue à l'évolution technologique et architecturale. Les objets introduits sont les suivants :

Grid La classe associée est *NanoArray* représentant une nanotuile. C'est une matrice 2-D comportant des connexions de fils horizontaux et verticaux aux intersections desquels sont placés des transistors. Les fonctions sont implémentées dans cette nanotuile.

Physiquement, cette représentation correspond à une superposition de deux couches de nanotubes de carbone (ou nanofils de silicium, cf section 2.1.3.1) séparées par une autre couche de dispositifs actifs. Dans cette représentation, la couche séparatrice est vierge. Cette notion peut être associée à l'aspect interne du nanobloc vu à la section 2.2.1.

Transistors La classe associée est *NanoTransistor* qui comporte deux sous-classes : *NfetNanoTransistor* et *PfetNanoTransistor* correspondant aux deux types de transistors existants. Ce sont des dispositifs placés sur la nanotuile avec une orientation particulière. Dans la représentation, ils permettent de remplir la couche séparatrice qui peut être vue comme un plan contenant des trous ; chaque trou correspondant physiquement à une intersection de deux nanotubes. Les dispositifs qui peuvent combler ces trous sont par exemple les aiguilleurs moléculaires (cf section 2.1.4).

Nano et micro wires :

NanoWires, fils composant la nanotuile. Cette notion virtuelle représente matériellement soit les nanofils de silicium ou les nanotubes de carbone. Ce sont les conducteurs électriques à l'échelle nanoscopique contenus dans tuiles. *MicroWires* (ou wires) indique la notion classique d'un fil conducteur. Dans les architectures présentées, Cette notion se rapporte aux longues lignes (ou simplement aux microfils) qui alimentent les cellules distantes.

NanoCrossOverConnect, connexion entre deux nanotuiles, cadre avec la notion de communication locale. C'est une notion qui regoupe les nanotubes de carbone (ou nanofils de silicium) qui sont directement connectés entre les nanoblocs (ou nanotuiles) voisins (cf section 2.2).

Connections Les composants constituant un NASIC sont représentés par les classes *NanoProgrammableConnect* et *NanoRaConnect*. La première regroupe tous les dispositifs actifs (les transistors). La table 3.1 montre la nouvelle hiérachie des classes dans le meta modèle abstrait. Les classes exprimant les nouveaux aspects architecturaux, qui ont été ajoutées à la structure des classes dans Madeo sont en couleur rouge.

En définitive cette extension offre des bénéfices diversifiés. D'une part, elle contribue à enrichir quantitativement la bibliothèque dans Madeo. Les portions d'architectures déjà définies peuvent être utilisables dans une autre architecture. D'autre part, elle permet d'élargir la nature abstraite des architectures. Certains aspects caractérisant technologiquement les NASICs et initialement non existants dans la représentation le sont déjà grâce à l'ajout de nouvelles notions.

3.1.1.2 Modèle de nanofabrique

Après l'extension du méta modèle, la nanofabrique est décrite grammaticalement. Cette description génère un modèle et l'instancie pour produire une modélisation concrète de l'architecture. Le code de la description de l'architecture est jointe en annexe A.1. La figure 3.1 montre le résultat obtenu dans l'éditeur régulier.

RaObject (container position selected additionalParameters)
- RaTriState (input output command)
- RaNilObject
- RaFunctionDescription (inputs outputs list)
- RaConnect (pins)
  - RaProgrammableConnect (c r)
    - RaBuffer (input output)
      - RaInvertBuffer
    - RaPassTransistor (currentState)
    - RaPip (resources)
    - RaSwitch (x y width)
      - RaConcreteSwitch (hResources vResources pips)
      - RaFunctionalSwitch (resources east west north south)
  - RaHardConnection
- NanoRaConnect (pins)
  - NanoRaProgrammableConnect (c r)
    - Transistor (orientation)
      - NfetTransistor
      - PfetTransistor
- RaNode (view size baseCost occupancy presentCongestion historicalCongestion pFac)
  - RaNodeWithoutNammedPins
    - RaWire ((pins expanded)
      - Microwire
      - RaOutputPin
      - RaInputPin
      - RaPin
      - Nanowire
    - RaExpandedWire (tracks)
      - RaExpandedPin
  - RaNodeWithNammedPins
    - RaMemory (adress data rw memory)
    - RaMultiplexer (inputs output inputsName outputName selection)
    - RaFunction (inputs outputs function possibleFunctions)
    - RaRegister (input output inputName outputName clock type)
    - RaMultipleSelection (inputs output)
    - RaMultipleOutputs (input outputs)
- RaCompositeObject (interface objects view)
  - RaArrayedObject (x y functions functionsPosition)
    - NanoArray (hNanowiresvNanowires connections)

TAB. 3.1 – Nouvelle hiérarchie de classes du méta modèle abstrait : les classes en rouge sont les classes ajoutées.

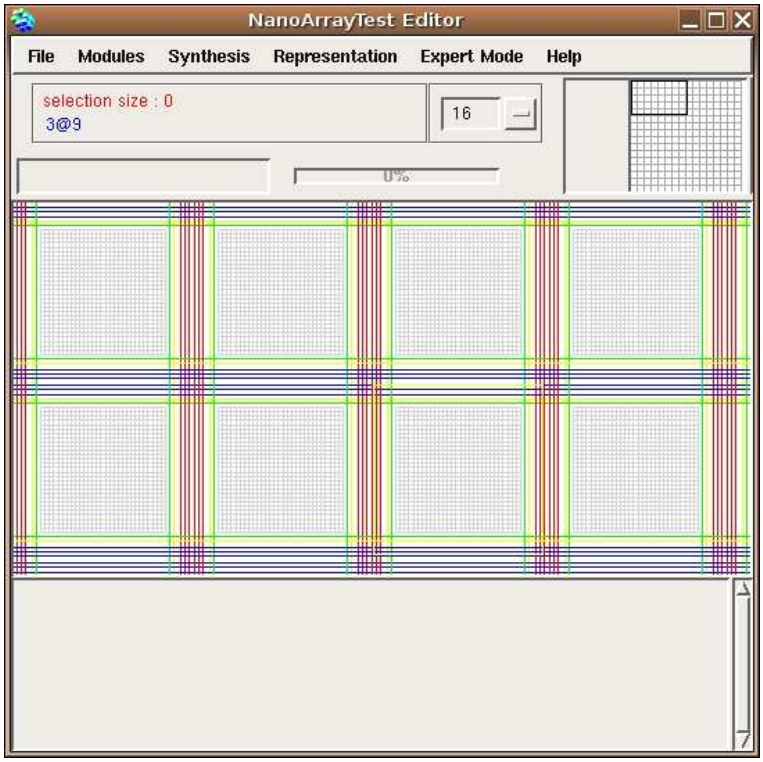
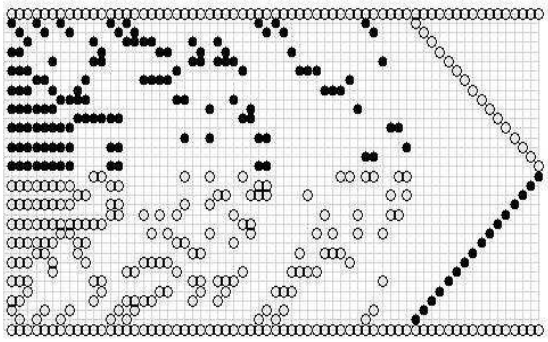


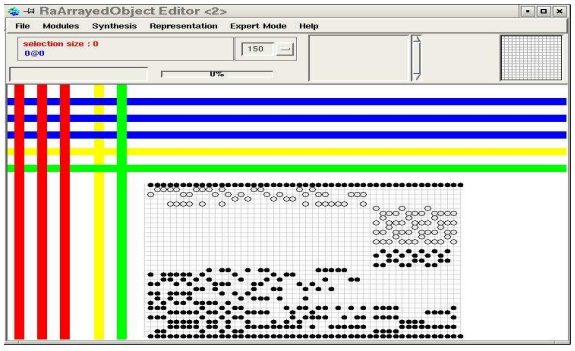
FIG. 3.1 – Capture d'écran de l'éditeur *(un pavage de tuiles de nanofabrique).*

3.1.2 Adaptation des outils

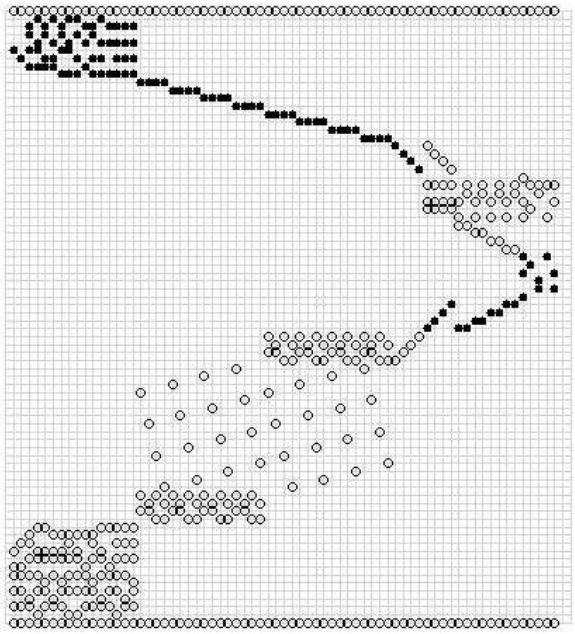
Les architectures décrites pour les circuits nanoélectroniques sont directement basées sur les tuiles logiques dont la structure est un crossbar (matricielle). La programmation des tuiles NASICs repose sur une logique de type PLA. Dans une grille, il n'existe pas de différence entre le calcul et le routage : plusieurs nœuds de calcul peuvent cohabiter au sein d'une même tuile en partageant des signaux. Prendre en entrée d'un nœud un signal issu d'autre nœud implique un routage implicite. Ce type de routage peut être fortement dégradé si le taux de défauts dans une grille est très important (coupures de nanofils, non-fonctionnement des jonctions programmables). Le routage peut aussi intervenir entre les différentes tuiles. Il est alors explicite. Les innovations dans le routage des architectures pour les nanoélectroniques réside essentiellement dans la spécificité du routage dans une structure crossbar (tuile). Pour les grandes classes d'architectures, les algorithmes génériques de routage ne changent pas pour les modèles disponibles dans Madeo mais il faut juste fixer quelques paramètres. Les algorithmes deviennent spécifiques quand il s'agit du routage au sein de la structure crossbar.



(a) Compteur de programme à 8 bits.



(b) ALU complet sur Madeo. Possibilité de varier le nombre de bits.



(c) Fichier des registres sur Madeo.

FIG. 3.2 – Modélisation automatique dans Madeo.

3.2 Conception des applications

3.2.1 Description comportementale

L'application à implémenter est décrite sous forme de méthodes dans Madeo. Ce dernier fournit alors une représentation graphique en créant un graphe de flot d'opérations. A chaque entrée de l'application, est associé un type correspondant à un ensemble de valeurs possibles. La sortie de la compilation est une LUT contenant toutes les combinaisons possibles selon les valeurs d'entrée.

La compilation de la description permet de générer les PLAs implémentables sur la nanotuiles. Deux visions sont possibles : une vision atomique de l'application et une autre structurelle. Pour la première, la méthode se compose de plusieurs nœuds et est appelée par une autre méthode. Il existe un seul appel fonctionnel rendant la vision atomique. Dans le second, les différents appels fonctionnels sont conservés en donnant un graphe de flot de données avec plusieurs nœuds.

3.2.2 Logique deux niveaux et signaux complémentés

La synthèse dans Madeo produit normalement des fichiers de type BLIF (Berkeley Logic Interchange Format) [58]. Pour respecter le paradigme de la logique deux niveaux, le flot de synthèse a été modifié pour produire des fichiers PLAs. La minimisation logique intervient via espresso [84]. Les signaux sont présentés dans l'architecture sous leur forme normale et complémentée. La minimisation est réalisée après un partitionnement des fichiers par couple de 2 bits en sortie. Puis une phase de fusion permet de reconstruire le fichier PLA.

Cette solution peut conduire à des PLAs plus complexes que nécessaires, pour lesquels certaines combinaisons d'entrée ne sont pas détectées comme non significatives. En revanche, la surface est conservée même dans le cas où le PLA est sous optimal.

3.3 Processeur WISP-0 sous Madeo

Pour s'assurer de la validité des outils étendus, une première étape a consisté à produire le processeur WISP-0, en lien avec l'équipe de Moritz. Pour cela, les cinq étages du processeurs (PC, ROM d'instructions, décodeur d'adresses, ALU, banc de registres) ont été décrits dans le formalisme d'entrée de Madeo. Les amplitudes de données ont été fixées conformément aux spécifications de WISP-0. Les descriptions logiques générées ont été confrontées aux fichiers originels : les tracés obtenus sont comparables en surface aux tracés de référence.

Cette implémentation du processeur WISP-0 sous Madeo a permis de dégager des règles de conception et de production pour la génération de tracés de tuiles. D'autres applications peuvent être implémentées grâce à une généralisation de ces règles.

3.3.1 Description bas niveau des étages de WISP-0

La conception manuelle du WISP-0 réalisée par l'équipe de Moritz repose sur sept fichiers PLA de référence :

- **add_mul.out** l'unité arithmétique et logique.
- **dec.out** le décodeur d'instructions.
- **dec24.out** le décodeur activant l'écriture dans les registres (l'instruction nop ne produit aucune activation).
- **inc4.out** l'incrémenteur du compteur d'instructions.
- **mux21.out** multiplexeur permettant la sélection entre la valeur d'un registre et la valeur immédiate.
- **mux41.out** multiplexeur de sélection de registres.
- **rom.out** la mémoire contenant le programme à exécuter.

3.3.2 Composants du WISP-0

Dans le but de modéliser le processeur WISP-0, une classe Smalltalk-80 a été implémentée après une analyse des différents fichiers PLA. Les méthodes de la classe WISP-0 représentent les étages du processeur (incrémenteur, ALU, ...). Cette représentation en Smalltalk permet la génération automatique des fichiers PLA correspondants et la simulation du circuit.

Opérations binaires

Les opérations binaires telles que les décalages sont exprimées de la manière suivante (la première définition indique la notation mathématique et la seconde est son équivalente en notation Smalltalk) :

- Décalage à gauche de n bits
  - $bin * 2^n$
  - `bin * (2 raisedTo: n)`
- Décalage à droite de n bits
  - $bin/2^n$
  - `bin quo: (2 raisedTo: n)`
- Récupération des n bits de poids faible
  - $bin\, MODULO\, 2^n$
  - bin \\ (2 raisedTo : n)

Compteur d'instructions La méthode implémentant le compteur incrémente l'argument de 1 et retourne le reste de la division de l'incrémentation par la taille de la mémoire (MAxProgramLength) comme le montre le code défini dans la table 3.2.

ROM La mémoire est représentée par un tableau adressé par un index donné en argument.

```
increment:a
^ a + 1 \\ MaxProgramLength
```

TAB. 3.2 – Code du compteur

Décodeur d'instructions Cette méthode décode les instructions lues dans la ROM en les traduisant vers un format 9 bits utilisable par l'ALU. Le format de l'instruction obtenue est détaillé par le tableau 3.3.

opcode[3 bits]	$operand_1$[2 bits]	$operand_1$[2 bits]	$operand_2$[2-bits]

TAB. 3.3 – Données en sortie du décodeur d'instructions

L'adresse 1 est répliquée car elle est utilisée par le décodeur 2 vers 4 pour permettre l'écriture dans le registre de destination. L'argument est l'instruction lue en mémoire.

Les informations sont récupérées en faisant une succession de décalages de bits (voire la table 3.4).

```
instructiondecoderInternal: a
instruction := a quo: (2 raisedTo: 2 * OperandWidth).
addr1 := (a quo: (2 raisedTo: OperandWidth)) \\ (2 raisedTo: OperandWidth).
addr2 := a \\ (2 raisedTo: OperandWidth).
^ instruction * (2 raisedTo: 3 * OperandWidth)
   + (addr1 * (2~raisedTo: 2 * OperandWidth))
   + (addr1 * (2 raisedTo: OperandWidth)) + addr2
```

TAB. 3.4 – Code du décodeur d'instructions

Cette donnée passe par les multiplexeurs du banc de registres pour sélectionner les registres à adresser. Operand1 permet la sélection par un multiplexeur 4 vers 1 du registre destination. Operand2 permet la sélection soit d'un registre dans un cas `opcode registre, registre` ou bien l'envoi direct de l'opérande à l'ALU dans le cas `opcode registre, entier`.

Unité arithmétique et logique La méthode consiste à faire une succession de comparaisons pour exécuter l'opération correspondant à l'opcode. L'opcode est codé sur 2 bits. Lorsque la valeur opcode correspond à 1 ou 2, l'argument b est retourné. Opcode est égal à 3, les arguments a et b s'additionnent modulo une valeur maximale qui représente la taille du registre. La dernière valeur que prend opcode donne lieu à une multiplication de deux arguments. La table 3.5 donne le code définissant cette méthode.

```
aluInternalNew: opCode arg: a arg: b
^true = (opCode = 1 "movi"|(opCode = 2 "mov"))
    ifTrue: [b]
    ifFalse:
     [opCode = 3 "add"
     ifTrue: [(a + b) \\ MaxValue]
     ifFalse:
       [opCode = 4
ifTrue: [a * b \\ MaxValue "mult"]
       ifFalse: [ 0 "nop"]]]]
```

TAB. 3.5 – Code de l'ALU

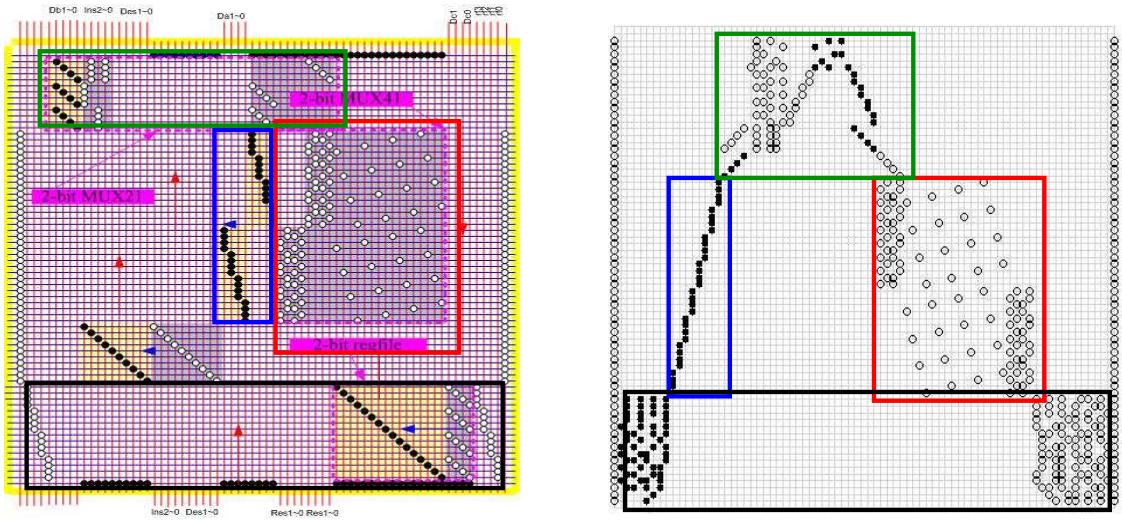
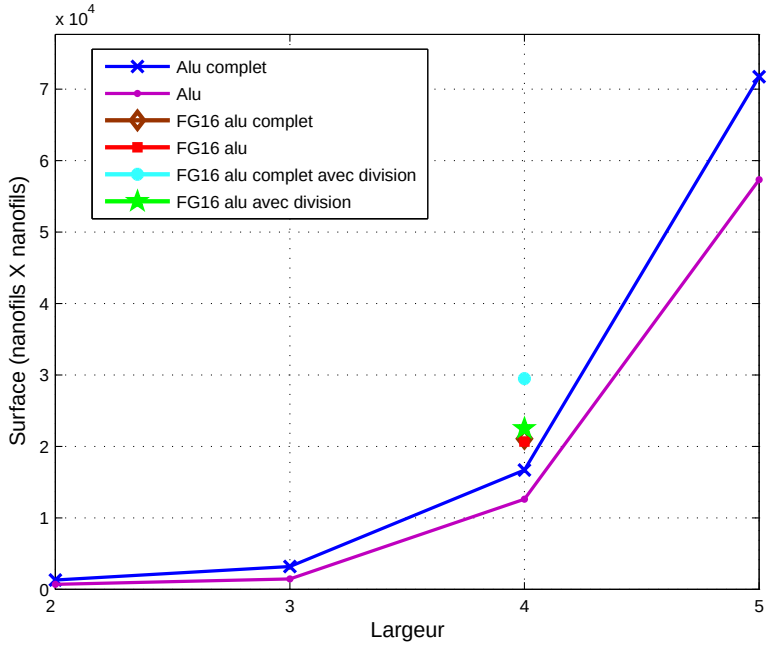


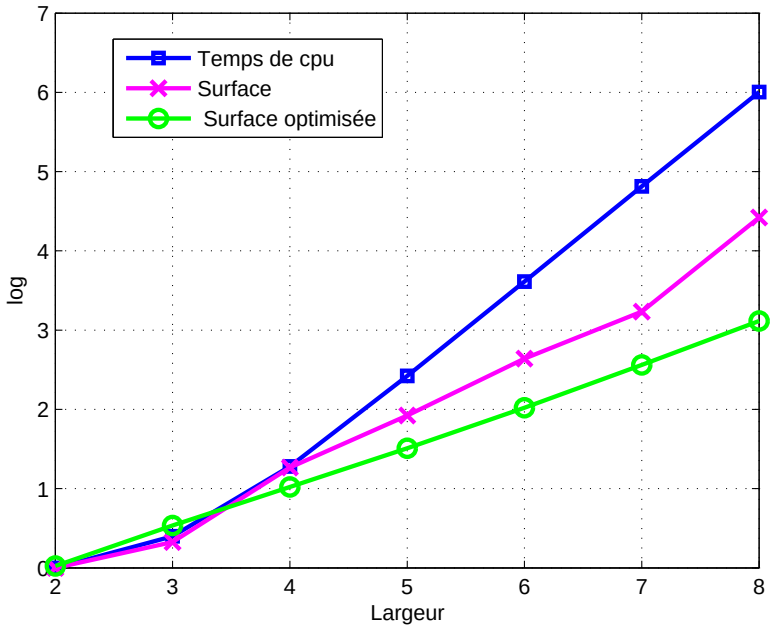
FIG. 3.3 – Comparaison des représentations (manuelle et automatique).

3.3.3 Modèles automatiques du WISP-0

Une automatisation primaire des tracés des étages de WISP-0 a été réalisée sous Madeo. Les captures d'éditeurs de quelques modules sont illustrées dans la figure 3.2. Une description haut niveau a permis de générer les PLAs de tous les composants de WISP-0. Pour vérifier la validité des PLAs obtenus, une comparaison a été faite avec les PLAs réalisés manuellement. L'équivalence des fichiers (PLAs manuels et PLAs générés) garantit l'exactitude de la description Smalltalk. L'équivalence de la sémantique des tracés topologiques manuels et automatiques fiabilise la démarche malgré la différence de localisation de certains blocs de transistors due aux optimisations.



(a) Prospection de la surface d'une ALU.



(b) Evolution logarithmique de la surface de l'ALU et du temps CPU.

FIG. 3.4 – Croissance de la surface de l'ALU en fonction de la largeur et du temps mis pour calculer celle-ci. Variation de l'arithmétique dans l'ALU.

3.3.4 Analyse comparative

Les descriptions logiques générées ont été confrontées aux fichiers originels comme l'illustre la figure 3.3 qui présente les mises en œuvre (manuelle et automatique). Les différents blocs implémentés sont identifiés par un jeu de couleurs (le rectangle rouge marque l'identification et la localisation des multiplexeurs 4 vers 1 dans les deux tracés). Les tracés obtenus sont comparables en surface par rapport aux tracés de référence.

Une analyse profonde permet de mettre en évidence certaines erreurs introduites dans l'implémentation originale de WISP-0. Elle permet aussi de souligner quelques propriétés qualitatives. En effet, la réalisation permet d'envisager une rapidité dans le processus de conception des systèmes complexes et d'étudier la mise à l'échelle (dimensionnement des nanotuiles) qui permet d'envisager le calcul d'impacts : par exemple, faire varier la largeur des ressources en regardant l'évolution du coût spatial.

La disponibilité des outils a par ailleurs permis de quantifier le coût de certaines options architecturales. Par exemple, une prospection architecturale a consisté à faire varier la taille des opérandes de WISP-0 de deux à huit bits. La figure 3.4 montre l'évolution de la surface et du temps CPU en fonction de la largeur des opérandes d'une ALU. Le point intéressant est l'aspect logarithmique des courbes représentant la surface dans la figure 3.4(b). Cet aspect souligne la linéarité de l'accroissement de la surface. La courbe de temps CPU révèle un facteur limitant de l'approche puisque l'évolution du temps CPU n'est pas linéaire. Quand la taille de PLA (n bits de sortie) est trop grande pour être directement traitée, le PLA est divisé en $\frac{n}{2}$ PLAs à deux bits (normal et complementé). Ces sous PLAs sont alors minimisés et fusionnés pour obtenir le PLA originel. Toutefois, une différence de linéarité entre les courbes de l'évolution de la surface de l'ALU est observée et s'explique par le fait que la synthèse utilisée ne sait pas faire la logique combinatoire.

Un essai a consisté de faire varier l'arithmétique mise en œuvre (l'opération dans les corps de Galois GF^{16} pour un surcoût de 26.3% en surface). Le bénéfice est que cette arithmétique supporte les divisions inversibles. Le coût mesuré d'une extension du jeu d'instructions se lève à 38.9% en surface par rapport à une simple ALU.

3.3.5 Modèle schématique du processeur WISP-0

La nécessité de ce modèle est de disposer d'une version simulée du processeur WISP-0 et d'observer son extension en augmentant la taille des opérateurs. Un bénéfice est d'avoir un modèle simulable du processeur. Le principe utilisé sera alors appliqué pour les autres modèles architecturaux (FPGA nanoscopique), en définissant le comportement fonctionnel de tous les briques de base. Dans le chapitre suivant nous allons utiliser le même principe structurelle pour la mise en œuvre d'une architecture de FPGA sur un support nanotechnologique.

La démarche consiste à définir schématiquement les composants du WISP-0 dans l'outil de simulation schematic (Madeo). Ces composants sont alors structurés pour éditer globalement le WISP-0 : création des connexions entre les ports d'entrée et de sortie des composants. La figure 3.5 montre une vue structurelle du processeur. Le code source de

cette vue schématique est produit. Il est analysé et simplifié pour générer le modèle simulable de façon programmée : toutes les sorties d'un module sont reliés à une même instruction.

Une extension du modèle est réalisée. Elle permet de passer d'un modèle limité (version initiale du processeur a 2 bits) à un modèle plus large où la taille des opérandes peut varier.

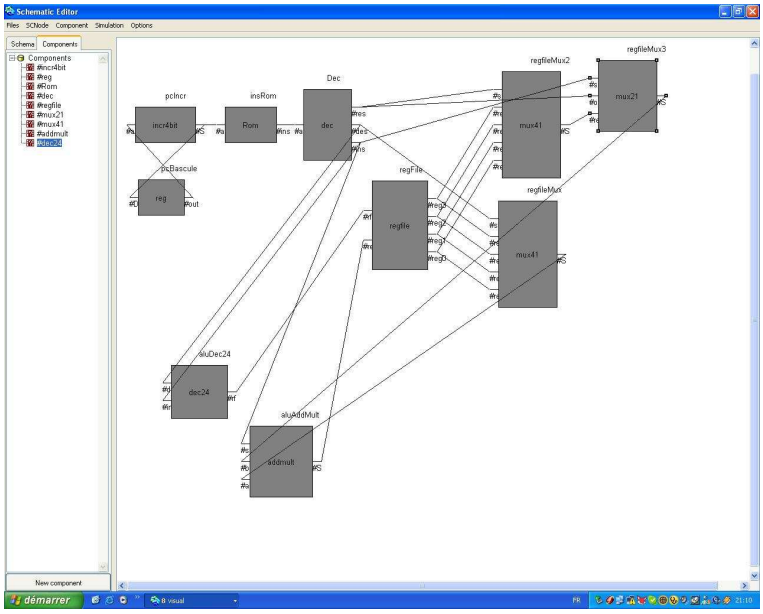


FIG. 3.5 – Schéma simulable du WISP.

3.4 Conclusion

L'adaptation de l'environnement de Madeo pour programmer, calibrer et implémenter des architectures sur un support nanotechnologique a été montrée. Elle révèle une extension quantitative et qualitative de la bibliothèque de l'environnement. Les outils ne changent pas fondamentalement.

Pour valider cette adaptation, l'exemple du WISP-0 a été traité. Le WISP-0 a été considéré comme une application décrite sous forme de méthodes. Cette description, au moyen de l'utilisation de PLAs fournis par Umass, a permis de produire les tracées topologiques des composants du processeur. Une analyse comparative de tracés produits et de tracés originels montre la pertinence des représentations générées.

Le chapitre suivant aborde un autre champ axé sur les architectures reconfigurables. Les bonnes caractéristiques (prospection,rapidité, calcul d'impacts,...) tirées de cette étude sont retenues pour adresser les architectures reconfigurables.

Chapitre 4

Construction d'un FPGA en NASICs

4.1 Hypothèses technologiques

4.1.1 Nature des supports technologiques

La réalisation des supports technologiques intégrant les nanodispositifs repose sur la disponibilité de ces nanomatériaux. La nature reconfigurable de tels supports est conditionnée par les hypothèses nanotechnologiques, en particulier la capacité ou non d'un aiguilleur d'être reprogrammable. La première hypothèse parait simplifier la conception des circuits mais elle est actuellement très en avance de phase par rapport aux démonstrateurs technologiques. Elle offre directement une reprogrammabilité aux supports technologiques. Cet aspect de reconfigurabilité est rigoureusement argumenté par André DeHon dans [21]. Il atteste que les dispositifs aiguillés utilisant les nanotubes suspendus avec une jonction bistable expérimentés dans [52] peuvent avoir les connexions reconfigurables.

Les choix de mise en œuvre retiennent la seconde hypothèse qui confère aux supports une configurabilité sur un mode PROM. Cela impose la mise en œuvre de mémoires de configuration en interne dans les circuits. Cette option est pénalisante en surface par rapport à une solution bénéficiant d'une technologie reprogrammable, mais elle paraît plus réaliste à l'heure actuelle. De plus, cette solution permet d'envisager aisément des dispositifs d'auto-configuration puisque la configuration correspond aux valeurs maintenues dans le circuit, et potentiellement produites par le circuit. La reconfigurabilité dans les architectures se fait alors par construction instaurant des contextes de configuration dans les supports technologiques.

4.1.2 Des FPGAs virtuels aux nano FPGAs

La recherche de modèles stables de calcul se trouve également dans le domaine du reconfigurable. De travaux préliminaires ont ciblé la définition d'une architecture FPGA virtuelle [56], couplée aux outils d'exploitation. Cette structuration en couche apparaît très bénéfique pour exploiter les ressources en constante croissance. L'expertise développée dans ce cadre trouve un nouveau champ d'application pour une technologie cible pros-

pective, telle que décrite dans les travaux de Moritz.

La mise en œuvre d'une application repose par exemple de bas en haut sur : une technologie (CMOS), un support de programmation (un FPGA de commerce), un modèle de calcul (un réseau linéaire d'opérateurs systoliques). Pour minimiser la perte de performances occasionnée par la sur-couche, des fonctionnalités spécifiques au domaine sont intégrées dans les opérateurs.

De façon similaire, il apparaît possible de réaliser un FPGA (modèle de calcul) en appui sur un support de programmation (les tuiles NASICs) réalisé sur une technologie (grilles de nanotubes de carbones). A l'inverse de la démarche précédente pour laquelle les outils d'implémentation existaient (outils Xilinx) mais pas les outils de programmation (Madeo), les outils de programmation sont immédiatement disponibles (Madeo) mais pas les outils d'implémentation.

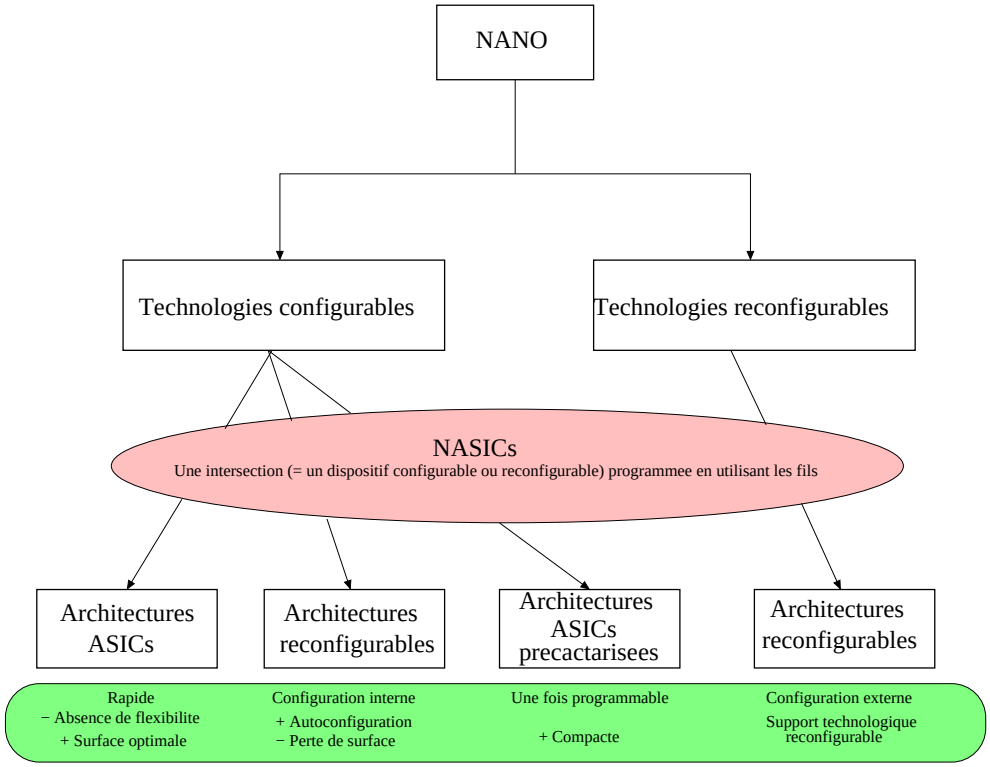


FIG. 4.1 – Différentes options possibles de réaliser les supports reconfigurables en fonction de la capacité des nanodispositifs (reprogrammables ou non).

4.1.2.1 FPGAs virtuels

Au même titre qu'une machine virtuelle (par exemple, la machine virtuelle java) permet de pérenniser un support de programmation en découplant la cible technologique (un processeur ou un système) du modèle de calcul (la machine virtuelle) au prix d'une certaine dégradation des performances, la notion de FPGA virtuel (modèle de calcul) a été

présenté pour permettre de porter directement les applications d'une famille de FPGAs à une autre et ainsi bénéficier des progrès matériels.

4.1.2.2 Des FPGAs sur supports NASICs

L'exploitation du reconfigurable dans des tels supports permet d'envisager des architectures reconfigurables puisque la reconfiguration suppose un changement de configurations, donc un choix des parties d'un support abritant des configurations spécifiques. En considérant les circuits NASICs, une nanofabrique peut avoir l'avantage d'être reconfigurable, ce qui permet de la rapprocher naturellement d'une fabrique reconfigurable comme le FPGA.

Les supports de nanotuiles implémentant les ressources de routage et de logique obéissent aux mêmes règles de programmation. Par conséquent, la faisabilité d'un FPGA sur une nanofabrique s'avère possible si les mécanismes et les ressources potentielles de configuration dans une grille de NASICs sont formels (présents et accessibles). Pour formaliser ces mécanismes dans un FPGA, les mémoires de configuration des éléments sont alors implémentées sous forme de RAM, proche en structure du banc de registres de WISP. Deux options de configuration s'avèrent possibles : une configuration interne avec une utilisation d'une RAM et une configuration externe avec l'utilisation d'une ROM. la figure 4.1 illustre les choix possibles de faire du reconfigurable selon la nature disponible des dispositifs nanotechnologiques de base.

4.2 Architecture

4.2.1 Briques architecturales

La mise en œuvre d'une architecture reconfigurable sur une nanofabrique de NASICs suppose de savoir implémenter un circuit sur cette technologie et de spécifier l'architecture à réaliser. Les briques de base de toute architecture de type FPGA sont généralement les mêmes et possèdent une même constitution.

Nous proposons les équivalents NASICs de ces briques (LUTs, multiplexeurs, aiguilleurs, ...) vues dans la section 1.1.2 pour entreprendre ensuite une implémentation simple d'une architecture reconfigurable.

4.2.1.1 Décodeur

La différence entre la description classique et celle qui dérive des NASICs se situe au niveau du nombre d'entrées/sorties : il double dans le cas des NASICs à cause de la complémentarité des signaux imposée par la conception. La notion du décodeur devient un circuit qui se compose de 2*n lignes d'entrées qui symbolisent une adresse sur n bits et de 2^{n+1} lignes de sorties possibles.

La figure 4.2 représente un layout d'un décodeur d'adresses à 4 bits. Le schéma comporte 8 nanofils qui servent d'entrées au décodeur (les signaux a_i et leurs complémentaires).

Il contient 16 nanofils commme sorties (les signaux s_i et leurs complémentaires). La partie inférieure montre les différentes combinaisons qui conduisent à la sélection d'une ligne de sortie. Cette représentation favorise la variation paramétrique du décodeur.

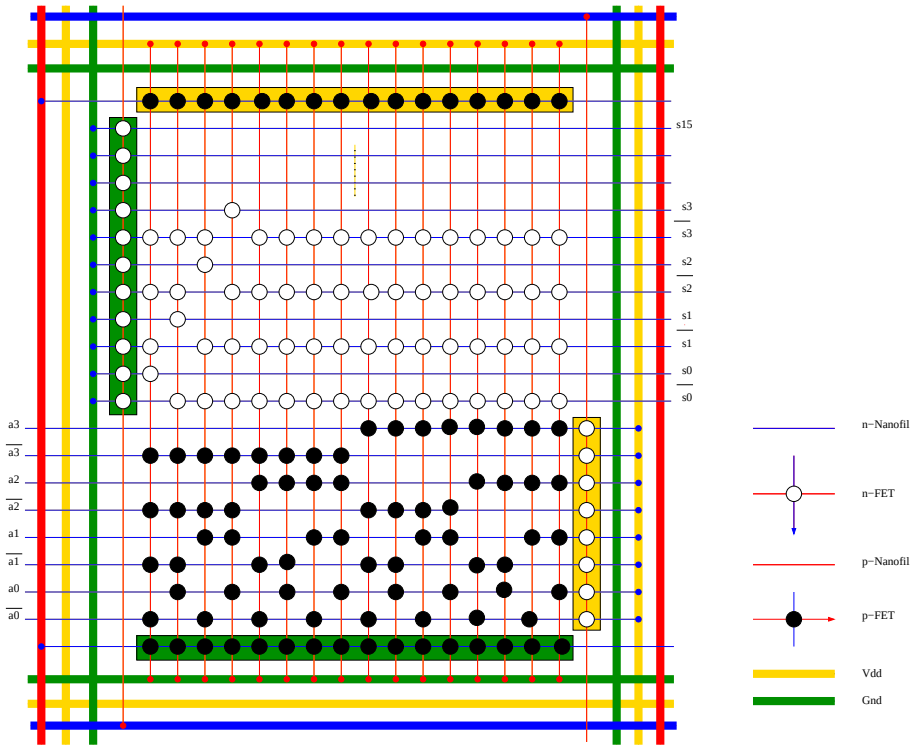


FIG. 4.2 – Décodeur d'adresses *(en bas de la figure, une zone de sélection d'adresses qui associe une adresse à une sortie : les sorties sont marquées par les points blancs).*

4.2.1.2 Représentation de la LUT

De façon générale, la LUT contient 2^n entrées de sélection et 2*k fils d'autorisation d'écriture (k entrées dans une configuration classique). En sortie, elle possède symétriquement 2^n lignes de sélection et 2*k lignes de validation de lecture. Entre les lignes d'autorisation d'écriture et celles de validation de lectures, il y a 2^{n+1} lignes de réduction contenues dans un plan de OU. La figure 4.3 exemplifie une partie de la conception. Elle montre 2 fils d'écriture (wVal et son complémentaire) et 4 fils d'entrée de sélection. Ces derniers sont couplés aux fils d'écriture pour former les lignes de réduction (les points dans la figure formant une droite oblique). Les signaux passant par les lignes de sélection sont dirigés vers la sortie (rVal et son complémentaire) par un mécanisme de routage représenté par deux droites obliques de points.

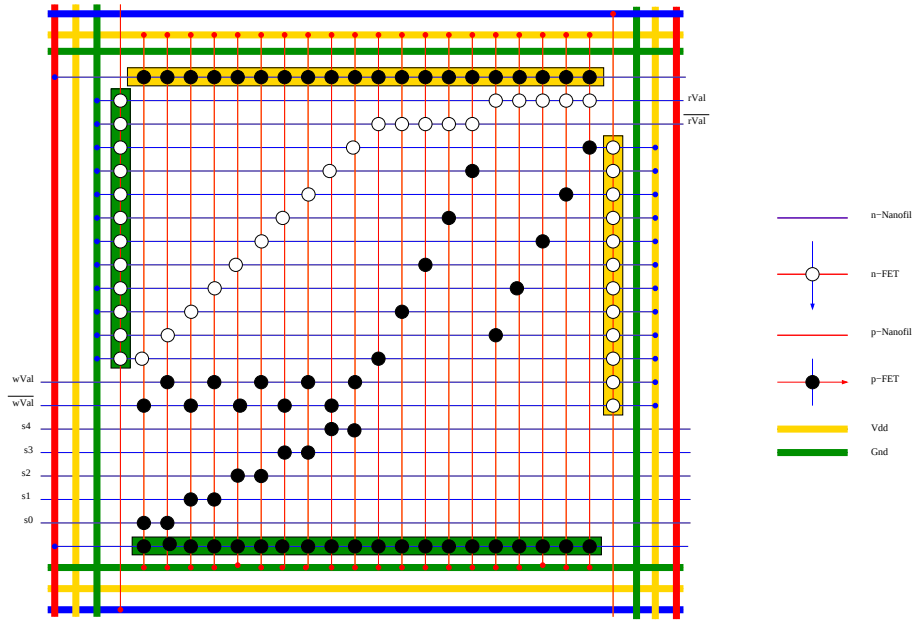


FIG. 4.3 – LUT en NASICs. *(En bas du schéma, localisation de la zone d'écriture (affectation de valeurs wVal et son complémentaire aux valeurs s_i). Au dessus de cette zone, mécanisme de lecture des données).*

4.2.1.3 Configuration du routage

Un PIP (Programmable Pnterconnect Point) est constitué de multiplexeurs de 2*n entrées et une sortie. Chaque multiplexeur contient 2*n lignes à l'entrée et q= 2*$\log_2$n lignes de sélection correspondant au nombre de lignes de configuration de la RAM. Le réseau de routage présente k lignes d'extraction des signaux qui correspondent aux k sorties des multiplexeurs. La sortie de la mémoire de configuration présente 2k*$\log_2$n lignes de décodage. La figure 4.4 montre un bloc RAM qui mémorise la configuration de 4 lignes d'extraction de signaux ($selX_{1\leq i\leq 4}$) grâce 4 lignes de configuration ($config_{0\leq i\leq 1}$ et leurs complémentaires). La sortie du bloc représente un décodeur d'adresses de 16 lignes (ou 8 bits d'adresses).

Les fonctions X_k sont les sorties des multiplexeurs qui composent l'unité fonctionnelle.

Les points de configuration symbolisés classiquement par les couples de bit ($config_0$, $config_1$) permettent d'implémenter n'importe quelle fonction.

4.2.1.4 Aiguilleur

L'aiguillage est un élément fondamental de routage. Il permet d'orienter un signal suivant un chemin en croix dans 3 directions en fonction de l'adresse affectée à chaque direction. Une présentation d'un tel aiguillage pour le NFPGA est montré à la figure 1.5(a). C'est un multiplexeur 4 → 1. Tous les aiguillages de la cellule de celui-ci ont donc une structure semblable mais des directions d'accès aux cellules adjacentes qui diffèrent.

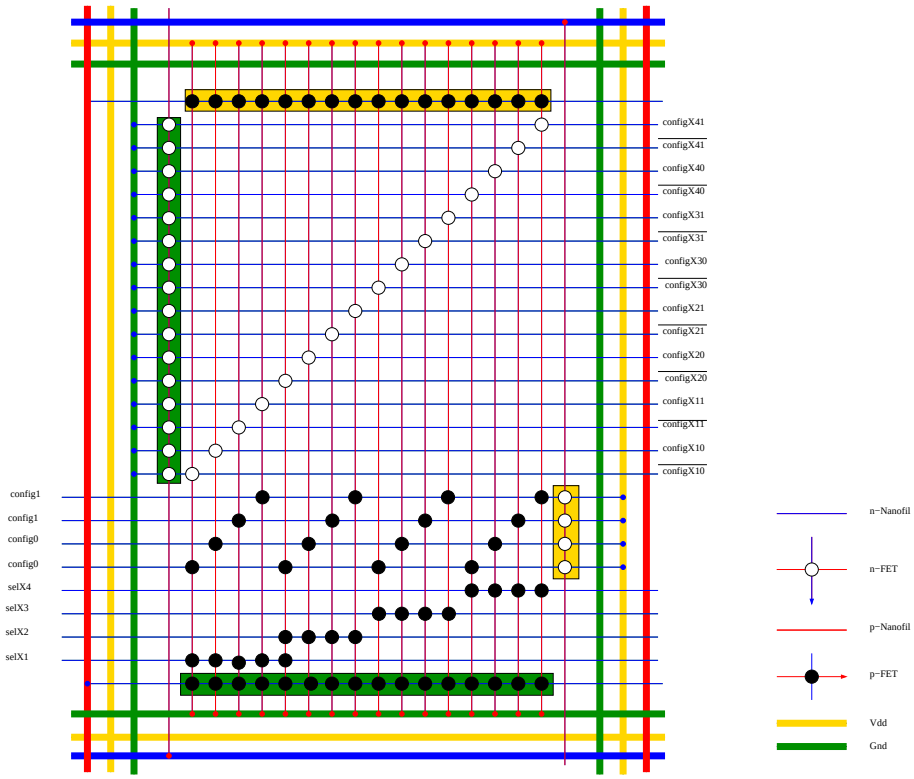


FIG. 4.4 – Configuration du routage : attribution aux valeurs selXi d'une valeur de configuration.

A titre d'exemple, nous présentons l'aiguilleur qui prend les signaux F, N, S et E en entrée, et le signal E_{out} en sortie. La réalisation d'un tel module est représentée à la figure 4.5. Les entrées a_i sont des bits d'adresses qui sélectionnent les 4 signaux à aiguiller. La partie inférieure est la zone d'adressage des 3 directions (North, South, East) : à droite l'adressage de direction East, au centre celle de la direction South et à gauche celle de la direction North. La partie supérieure constitue l'ensemble de combinaisons possibles de sortie selon l'adresse.

Nous avons présenté en technologie NASICs les modules de base pour faire une cellule de FPGA mais il reste un problème à résoudre. C'est celui de la structuration des modules dans la cellule pour réaliser une exécution cohérente d'un calcul. Cette structuration doit conserver au mieux les avantages de la technologie NASICs et amoindrir l'impact des inconvénients de cette dernière. La technique envisagée pour structurer une telle cellule est d'abouter les modules élémentaires les uns après les autres en respectant la redirection des signaux et les règles de conception des NASICs. Les signaux transmis à travers ces modules doivent suivent le sens de calcul. Cela impose un surcoût de surface pour rediriger un signal qui est supposé sortir dans une direction autre que celle du calcul. Il est donc nécessaire de grouper les modules de même type pour assurer la clairvoyance dans la direction des signaux et pour avoir un ajustement optimale dans la configuration des

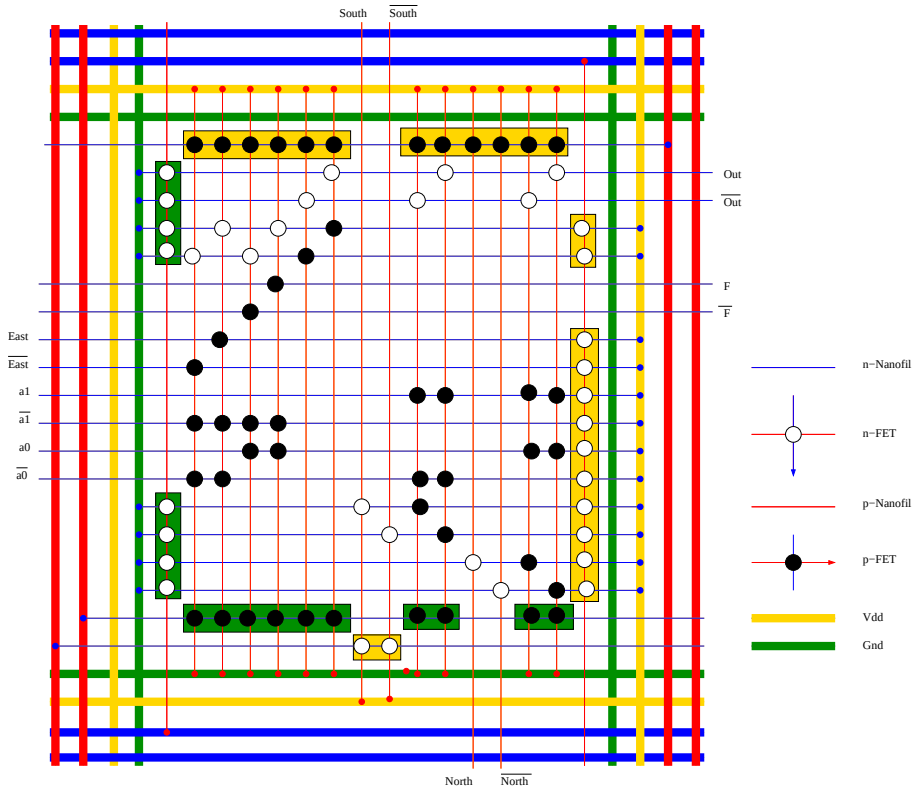


FIG. 4.5 – Les points de connexion d'un aiguilleur.

modules.

En somme, la mise en œuvre d'une architecture reconfigurable sur des tuiles NASICs présuppose donc de savoir implémenter un circuit sur cette technologie et de spécifier l'architecture à réaliser. Ce choix d'architecture est partiellement conditionné par la facilité attendue dans l'implémentation, et nous conduit à reconsidérer en tant que démonstrateur une architecture développée quelques années en arrière.

4.2.2 Une architecture de référence

La famille XC6200 [1] a fait l'objet de nombreux travaux académiques. Ceci s'explique d'une part par la nature ouverte de la description matérielle qui rend possible pour des tiers la réalisation des outils. Elle représente à cet égard un instrument pour expérimenter divers enjeux d'importance (la gestion du système de ressources, la routabilité, la programmation des architectures). D'autre part, la seconde caractéristique intéressante réside dans la capacité de reconfiguration dynamique, sur un mode RAM, à une interface FASTMAP. De plus, la simplicité et la facilité attendues dans l'implémentation contribuent largement à ce choix.

L'arrêt de cette famille a clairement été motivé par le déséquilibre entre les ressources de routage et les ressources logiques qui rendait la programmation difficile. Près de dix ans

après cet échec relatif, il est clair que reconsidérer cette architecture ne présente d'intérêt qu'à condition de pouvoir lever ces limitations. Pour cela, il convient, non pas de se focaliser sur une puce existante, mais plutôt d'en conserver la structure tout en fournissant une variante paramétrique. Dès lors, la disponibilité en amont des outils adaptables validés est nécessaire pour étalonner l'architecture en fonction des applications ciblées.

Une cellule de base de la famille XC6200 est faite de multiplexeurs et d'une unité fonctionnelle. Les multiplexeurs sont assemblés pour supporter le routage de signaux vers l'unité fonctionnelle et les cellules adjacentes. Une telle cellule possède ses propres ressources de routage directement connectées aux cellules voisines les plus proches.

4.2.3 Organisation structurée du FPGA

Ayant connaissances de règles de conception de la technologie NASIC, nous proposons une version préliminaire d'un FPGA qui peut être implémenté sur un ensemble des nanotuiles. L'utilisation des tuiles de NASICs comme une plateforme d'implémentation pour un telle architecture exige uniquement l'usage de la non bidirectionnalité du retour d'information et de connexions. Cela nous amène à reconsidérer l'architecture de XC6200 qui présente des caractéristiques similaires. Une reformulation de l'architecture permet de coupler cette conception avec celle des NASICs. Cela permet d'exploiter des tracés physiques simples pour une architecture reconfigurable avec une nanotechnologie.

4.2.3.1 Adéquation entre le support et l'architecture

Les NASICs, utilisés comme support de mise en œuvre de l'architecture du XC6200, présentent certaines limitations comme la difficulté de réaliser des bascules ou la nature directionnelle des canaux. Cependant ces limitations ne sont pas trop sévères du fait des besoins limités de l'architecture : aucun canal bidirectionnel, aucune mémoire. Par ailleurs, l'expérience du WISP-0 a démontré la faisabilité de banc de registres et d'ALU sur ce support.

4.2.3.2 Formulation du NFPGA

Toutes les ressources de routage au sein de la cellule sont directionnelles et orientées selon le sens de la propagation du signal pour adapter ou approprier la cellule à la nature des NASICs. L'unité fonctionnelle se limite à une LUT (une petite mémoire) qui peut implémenter une certaine fonction. La figure 4.6 illustre la conception globale d'une telle cellule.

Pour rendre la cellule reprogrammable, les zones de configuration sont ajoutées. Ces zones peuvent être des registres de configuration qui profitent des mécanismes de stockage temporaire de bits offerts par la technologie NASIC pour rendre l'interconnexion programmable : les données transmises peuvent être mémorisées sur un fil. La configurabilité de l'unité fonctionnelle se garantie par une sélection des nanofils programmant la RAM de la LUT.

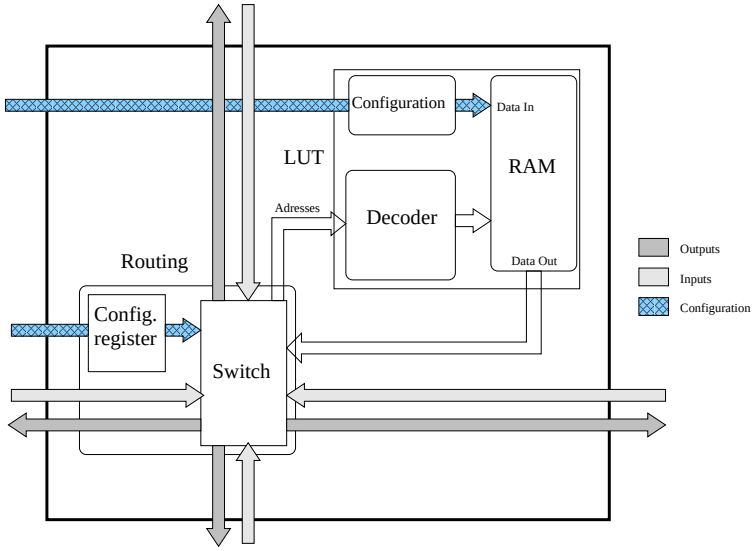
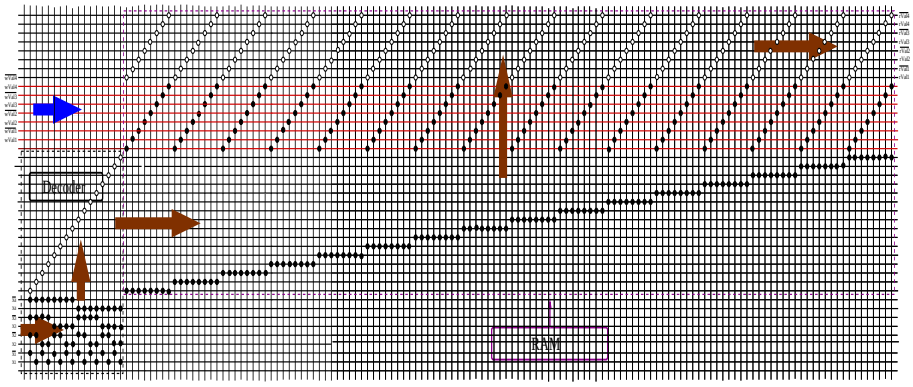


FIG. 4.6 – Vue globale de la cellule du NFPGA *(les connexions internes sont directionnelles).*

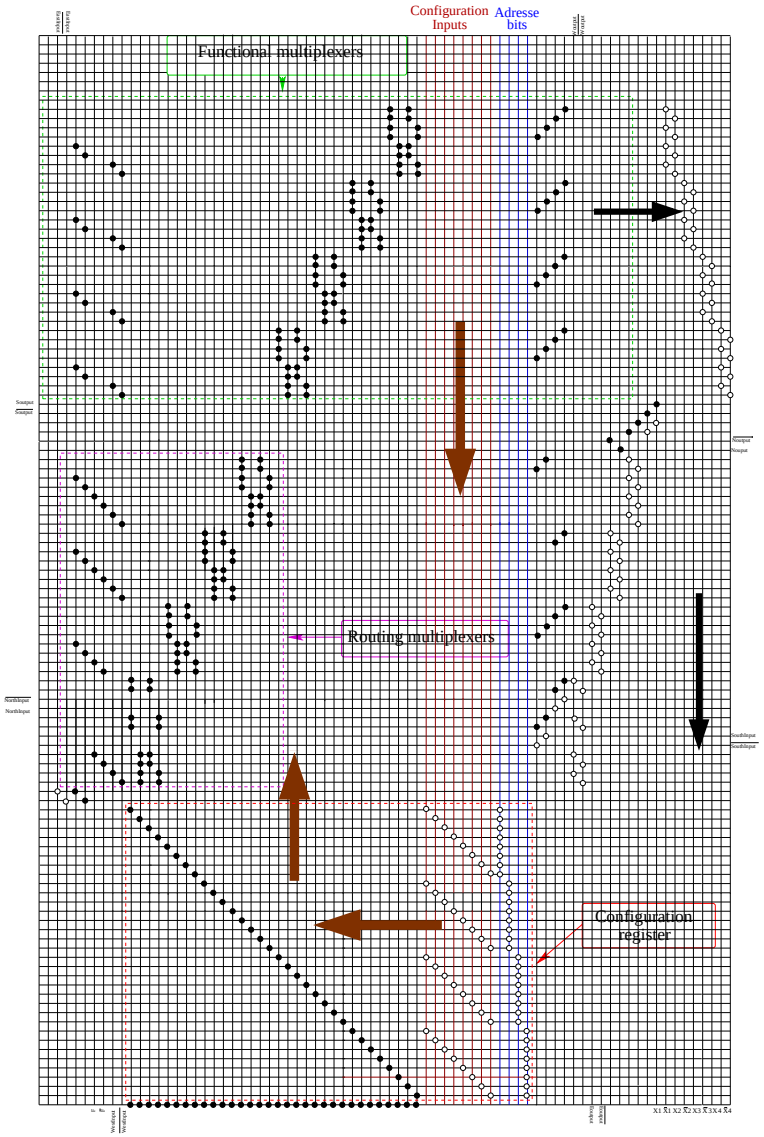
4.2.3.3 Layout du NFPGA

Sur la base de règles et de principes de conception des NASICs établis par Moritz et al., nous avons développé un prototype primaire du layout de la cellule du NFPGA. Pour des raisons de clarté, le domaine de routage et le domaine logique sont implémentés séparément sur deux crossbars différents. En réalité ils peuvent former une nanotuile ou un ensemble de nanotuile pour déceler les aspects du dessin et expliciter le principe de l'estimation de la surface englobant toute l'implémentation de la cellule. La figure 4.7 illustre le layout manuel de ressources de routage et de la logique au sein d'une cellule en combinaison des circuits NASICs. La première nanotuile sur la gauche implémente une 4-LUT. Elle possède bien deux majeures parties : un décodeur à 4 bits d'adresses en bas à gauche et une RAM en bas à droite. Les nanofils d'écriture/lecture se trouvent au dessus de la partie mémoire. Les nanofils constituant les entrées du décodeur permettent de sélectionner individuellement les nanofils directement connectés à la RAM. Dans la zone mémoire, une sortie du décodeur peut être programmée pour personnaliser les fonctions logiques à travers une zone de configuration (programmation d'écriture et de lecture).

La seconde nanotuile de droite est le layout du bloc de routage interne de la cellule. Il se trouve un registre 4-bits où les données sont temporairement stockées sur 32 nanofils horizontaux. Ces données permettent de configurer les deux blocs de multiplexeurs localisés au dessus du registre de configuration. Le bloc situé au centre de la grille permet d'aiguiller les signaux vers les nanotuiles implémentant les cellules voisines. Les signaux de sorties sont donc redirigés dans toutes les directions où se trouvent les cellules adjacentes. Les sorties du second bloc sont directement connectées à l'implémentation de la LUT.



(a) Layout de l'unité fonctionnelle. C'est une LUT dont la nanotuile implémente un décodeur suivie d'une RAM.



(b) Layout du domaine de routage. La partie inférieure représente le registre de configuration. Au centre se trouve la zone de multiplexage de routage.

FIG. 4.7 – Les layouts doivent être assemblés sur une même nanotuile pour former toute la cellule. Les points noirs et blancs représentent respectivement les transistors PFET et NFET.

4.3 Caractérisation du coût spatial de NFPGA

Le principe est d'avoir une métrique convenable pour toutes les architectures reconfigurables de type NFPGA qui peuvent être exploitables dans un outil générique. Pour caractériser cette métrique, nous avons exploité les informations sur le dessin pour formuler les équations qui traduisent l'aspect spatial de la structure d'une cellule.

La composition de l'implémentation d'une telle cellule de NFPGA montre qu'elle contient $4x$ multiplexeurs utilisés pour router les signaux vers les cellule voisines et k multiplexeurs attachés à la LUT (k entrées des fonctions X_k) : 4 indique le nombre de directions possibles et x la largeur de chaque multiplexeur. La taille de la chaîne binaire du registre de configuration est donnée par :

$$L = (16x + 2k \log_2(4x)) \tag{4.1}$$

En terme de conception, cette équation signifie que chaque multiplexeur de routage nécessite 4 nanofils de configuration et chaque multiplexeur alimentant la LUT exige $2 \log_2(4x)$ de nanofils de configuration. Le nombre de mots de configuration peut s'exprimer en fonction de la largeur de canaux, du nombre d'entrées de la LUT et de la taille d'un mot de configuration. C'est la partie entière du rapport de la chaîne de configuration par la longueur n d'un mot (équation 4.2).

$$m = \lceil \left(\frac{16x + 2k \log_2(4x)}{n} \right) \tag{4.2}$$

où $\lceil(\cdot)$ est l'arrondi supérieur de la partie entière de $(\cdot)$.

4.3.1 Méthodologie

La surface du cœur de la structure crossbar s'estime à partir du nombre de fils nécessaire pour son implémentation. La technique consiste donc à évaluer les surfaces englobant toutes les implémentations de blocs contenues dans une tuile et de les sommer pour déterminer la surface de l'implémentation globale. Chaque bloc élémentaire consomme une surface de la nanotuile symbolisée par un nombre de nanofils verticaux et horizontaux (les zones où sont concentrés les dispositifs actifs). Dans ce cas, l'aire d'un bloc est approchée en évaluant la taille de la grille de $h_i \times v_i$ nanofils de son implémentation (h_i et v_i sont respectivement le nombre de nanofils horizontaux et verticaux) comme le montre la figure 4.8. Dans le cas des nanotuiles, compte tenu de l'effet diagonal, cela revient déterminer les composantes de toutes les formes diagonales des implémentations de blocs contenus dans une tuile. Le coût du routage entre les implémentations peut être relativisé en l'emboitant dans ceux des formes diagonales des modules caractéristiques (registre, LUT, etc). Le routage entre cellules se manifeste à travers le bloc d'entrées/sorties.

Registre de configuration Le bloc registre situé en bas à gauche dans la figure 4.8 contient une nanogrille d'implémentation de taille $h_1 \times v_1$. En considérant la tuile de son

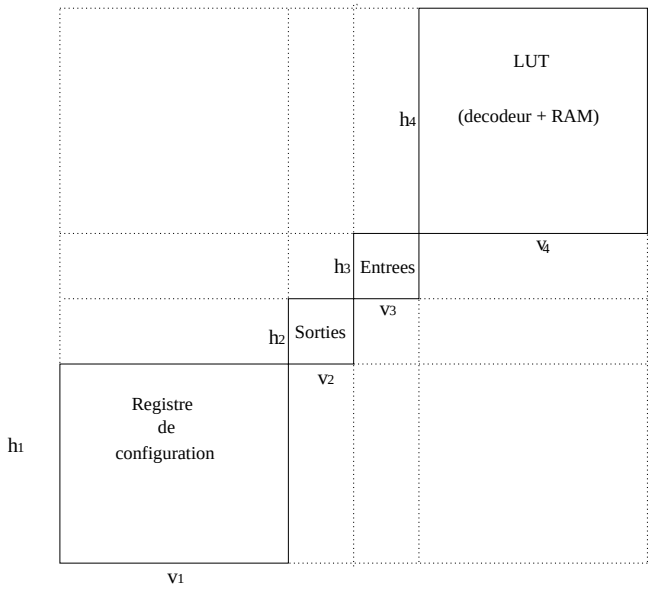


FIG. 4.8 – Principe d'estimation de la surface d'une nanotuile de la cellule de NFPGA.

implémentation (figure 4.7(b)), sa taille s'exprime directement en fonction des paramètres ci-dessus :

- 4x multiplexeurs de routage ($4 \rightarrow 1$) : chaque multiplexeur est implémenté à partir 8 nanofils horizontaux représentant les 4 directions et 4 nanofils verticaux de configuration. On obtient ainsi 32x nanofils horizontaux et 16x nanofils verticaux.
- Multiplexeurs d'alimentation ($4x \rightarrow 1$) : chaque multiplexeur est implémenté avec 8 nanofils horizontaux et $2 * \log_2(4x)$ nanofils verticaux. Il en résulte 8k nanofils horizontaux et $2k * \log_2(4x)$ nanofils verticaux.
- Une partie importante de ce bloc est la zone de stockage de la configuration. Cette zone est une tuile carrée de L nanofils où L est la longueur de la chaîne configuration. De plus, le registre se compose de n nanofils verticaux pour les bits de configurations. Chaque groupe de n bits de configuration est codé par une même adresse. Le nombre nanofils d'adressage correspond au nombre m des mots de configuration. En définitive, nous avons $n + m$ verticaux et L nanofils horizontaux pour compléter le registre.

En regroupant tous les fils suivant chaque dimension (verticaux ou horizontaux), nous déterminons les expressions des composantes du bloc registre :

$$\begin{aligned} h_1 &= 48x + k\,(8 + 2\log_2(4x)) \\ v_1 &= 16x + 2k\log_2(4x) + n + m \end{aligned} \tag{4.3}$$

bloc d'entrées/sorties C'est la partie de la cellule qui concerne les ressources de routage interne regroupant les entrées et les sorties. Elle se divise en deux sous blocs : le sous bloc de ressources d'entrée et celui de ressources de sortie d'une tuile.

Le premier renferme donc 4 points d'entrée qui correspondant aux différentes positions des cellules voisines. La réalisation d'une entrée nécessite 2 nanofils symbolisant une valeur et son complémentaire. En supposant que chaque point d'entrée possède une largeur x, l'implémentation de toutes les entrées du sous bloc exige $4x$ nanofils horizontaux et $4x$ nanofils verticaux. Certains signaux suivant une direction donnée sont redirigés dans une autre direction pour respecter le sens de propagation conventionnellement choisi. D'où l'utilisation $4x$ nanofils dans une direction verticale ou horizontale. Par exemple une entrée verticale est redirigée en consommant $2x$ nanofils verticaux et $2x$ nanofils horizontaux. Cette disposition n'implique pas que les entrées soient regroupées au sein de la nanotuile. Elle est purement représentative pour approximer l'aire occupée par les entrées dans une cellule. Les dimensions d'un tel bloc sont exprimées par les relations suivantes :

$$\begin{aligned} h_2 &= 6x \\ v_2 &= 10x \end{aligned} \tag{4.4}$$

Le même raisonnement appliqué au bloc représentant les sorties permet de définir sa taille. Les valeurs des composantes sont les suivantes :

$$\begin{aligned} h_3 &= 6x \\ v_3 &= 10x \end{aligned} \tag{4.5}$$

LUT Son implémentation en technologie NASIC renferme deux modules : le premier représente le décodeur d'adresses et le second bloc est une RAM. Les nanofils de sortie du décodeur constituent les entrées du module RAM. D'autres nanofils qui symbolisent les valeurs à écrire et à lire dans la RAM doivent être présents. Nous avons donc $2w$ nanofils d'autorisation d'écriture (les valeur à écrire et leurs complémentaires). De même pour la lecture, RAM possède $2r$ nanofils de validation. Pour simplifier, nous considérons le cas où le nombre de nanofils des valeurs à écrire est égal au nombre de nanofils des valeurs à lire (w=r).

Pour la détermination des composantes horizontale et verticale du bloc, nous considérons l'effet de bouclage du registre de configuration pour implémenter la partie décodeur. L'exemple d'implémentation de la cellule, illustré à la figure 4.7, montre dans la moitié droite de la tuile la disposition du bloc LUT : en haut l'implémentation du décodeur dont les entrées sont placées verticalement. Nous avons $2k$ nanofils d'adressage pour 2^k nanofils de sortie. De plus la RAM renferme verticalement $4w$ nanofils d'écriture/lecture (nanofils de données unidirectionnels) et $w2^{k+1}$ nanofils de mémoire.

La composante horizontale se restreint uniquement à la composante de la partie RAM en bas de la figure. Elle se résume aux $w2^{k+1}$ nanofils horizontaux pour la mémoire et $w2^{k+1}$ nanofils regroupant toutes les valeurs de lecture et leurs complémentaires dans les nanofils de sortie. Pour compléter l'implémentation de la partie logique de la cellule, nous ajoutons à chaque composante de la LUT $2w$ nanofils utiles pour diriger les signaux

de sortie de la LUT vers les différents multiplexeurs de sortie. Les relations donnant les valeurs des composantes sont exprimées par les équations 4.6.

$$\begin{aligned} h_4 &= 2w\left(2^{k+1}+1\right) \\ v_4 &= 2\left(k+3w\right)+\left(1+2w\right)2^k \end{aligned} \tag{4.6}$$

Les composantes de la nanotuile, représentant de une cellule, sont obtenues en sommant toutes les composantes verticales ou horizontales des formes diagonales des implémentations entre elles. Les composantes globale sont exprimées dans les équations 4.7.

$$\begin{aligned} X &\leq \sum_i h_i \\ Y &\leq \sum_i v_i \end{aligned} \tag{4.7}$$

X et Y représentent les composantes de la tuile globale. Ces deux équations sont alors utilisées dans la formule donnant la surface intégrale de la tuile définie dans [68]. Sachant que la taille du coeur de la nanotuile étant donnée par X*Y fils, la surface totale d'une nanotuile est approchée en admettant un minimum de surface représentant le réseau de pullup/pulldown et les microfils nécessaires pour le contrôle, l'acheminement de la configuration et l'alimentation électrique. Dans ce cas, il y a $2\log_2 X$ nanofils verticaux et $2log_2 Y$ nanofils horizontaux pour l'implémentation du réseaux de pullup/pulldown (dispositifs de rappel au niveau haut ou au niveau bas).De la même manière, il y a $2\log_2 X+4$ microfils verticaux et $2\log_2 Y+4$ microfils horizontaux (4 microfils d'alimentation électrique et de connexion à la masse). La surface totale d'une nanotuile est donnée par l'équation 4.8 définie par Golstein.

$$S = \left[(X)+2\log_2(Y)+s\left(\log_2 X+4\right)\right].\left[(Y)+2\log_2(X)+s\left(\log_2 Y+4\right)\right]. \tag{4.8}$$

Dans cette équation, le paramètre s représente le ratio d'espacement entre les nanofils et les microfils.

4.3.2 Evaluation

Toute phase de conception d'une architecture reconfigurable exige une stratégie d'évaluation de performances qui garantit l'efficacité de la méthodologie utilisée dans cette opération et peut renseigner sur les optimisations à envisager pour améliorer l'architecture. Une prédiction des performances d'une architecture en particulier la surface dans le cas présent par les simulations permet de nous munir de la mesure de sa faisabilité. Cela peut contribuer à valider l'architecture ou l'abandonner. C'est dans ce cadre les simulations suivantes ont été menées.

Evaluation de la surface de la nanotuile de cellule de NFPGA Pour évaluer la surface totale de la grille de la cellule de NFPGA en fonction de la variabilité des paramètres précédents, nous avons considéré que la tuile englobant l'implémentation de la cellule est un regroupement de trois nanotuiles : la nanotuile du bloc registre, la nanotuile de la LUT et la nanotuile de bloc d'entrées/sorties. La surface de chaque nanotuile se composant de 3 parties, il est donc nécessaire de préciser les facteurs technologiques des dispositifs. Nous supposons comme dans [68] que l'espacement entre les nanofils est de $10nm$ et que le ratio entre les nanofils et microfils est de 9. Nous supposons que le nombre de valeurs lues ou écrites dans une LUT et la longueur du mot de registre de configuration sont fixés pour étudier le coût spatial de la tuile implémentant la cellule de NFPGA en fonction de la largeur des canaux et du nombre d'entrées d'une LUT. Pour cette étude, nous avons fixé le nombre de valeurs lues ou écrites lors des opérations de lecture et d'écriture à 4 et la longueur du mot de configuration à 4. La figure 4.9 montre l'évolution des surfaces de nanotuiles. Elle illustre une forte influence du nombre d'entrées de la LUT dans la densité de la nanotuile de cellule de NFPGA. Les LUTs dont le nombre d'entrées n'excède pas 7 peuvent être pratiques d'autant que leurs surfaces peuvent être relativement insignifiantes malgré l'importance de la largeur des canaux.

Nous notons que la longueur d'un mot de configuration peut légèrement impacter sur la surface totale de cellule de NFPGA car la surface du registre de configuration est relativement faible. Par contre l'augmentation du nombre de valeurs de lecture ou d'écriture peut faire varier cette surface qui dépend beaucoup de celle de la LUT. Dans une prospection architecturale, il semble nécessaire de faire un compromis dans le choix des paramètres pour obtenir un modèle de layout de NFPGA de bonne qualité.

4.4 Programmation des architectures de NFPGA sous Madeo

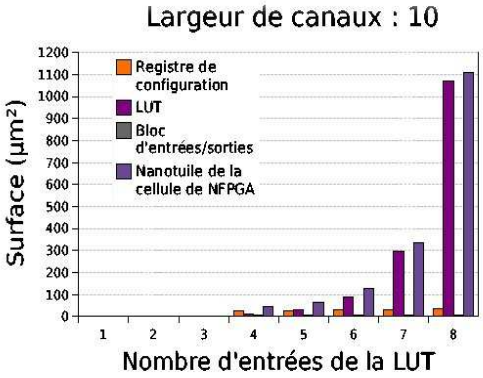
4.4.1 Modèle

Le modèle est défini sur la base des hypothèses simples pour assurer sa routabilité. Il se caractérise par une paramétrisation qui lui confère la capacité d'être extensible dans la programmation.

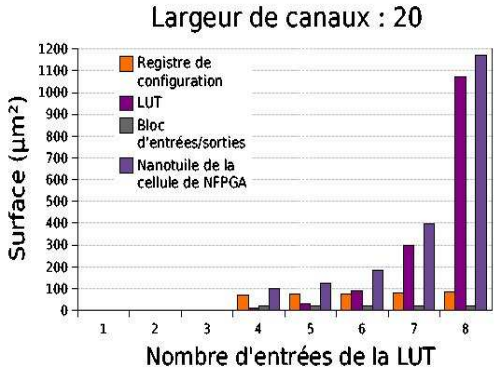
4.4.1.1 Hypothèses

Dans les FPGAs classiques, le routage interne et intracellulaire constituent un élément clé qui impacte significativement la performance et la densité logique. En effet le choix d'une topologie du bloc d'aiguillage est déterminant pour la routabilité d'une puce, l'aire et le délai des circuits implémentés sur un FPGA [85, 50, 29].

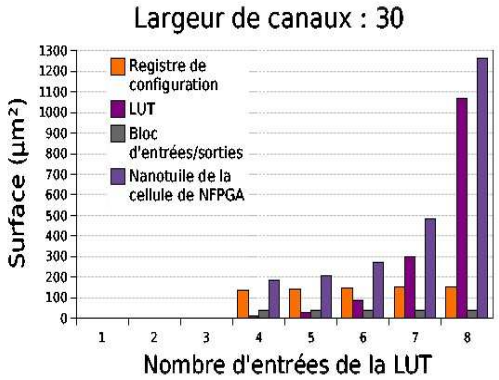
Les composants de base contenus dans une cellule sont assemblés par le biais des outils de composition selon une topologie de routage choisie par le concepteur. Toutefois tous les types de motifs d'interconnexion sont disponibles : l'interconnexion disjointe qui possède



(a) Les surfaces des nanotuiles sont relativement faibles pour les petites valeur de la taille de la LUT. Au delà de 8 entrées de la LUT, La surface de nanotuile de la cellule est fortement influencée par celle de la LUT.



(b) Augmentation légère des surfaces du registre de configuration et du bloc d'entrées/sorties. La surface de la LUT reste constante.



(c) Augmentation de la nanotuile de cellule de NFPGA due à la croissance de la densité dans le bloc registre de configuration et le bloc des entrées/sorties.

FIG. 4.9 – Estimation de la surface de la nanotuile de cellule de NFPGA en fonction des paramètres.

une structure simple contribue efficacement à préserver la densité. Un signal rentrant sur une piste étiquettée d'un canal de routage ne peut sortir du bloc d'aiguillage que sur une piste de même label. un autre type d'interconnexion dit Wilton [96] apporte plus de flexibilité dans le routage en supprimant les domaines de routage engendrés par la nature disjointe de la topologie précédente. Il produit une meilleur densité quand les ressources sont de longueur unitaire. D'autres types d'interconnexion existent mais semblent plus denses (topologie de connexion complète).

Pour simplifier la structure des multiplexeurs, le routage par défaut est de type disjoint bien que d'autres motifs puissent être retenus.

4.4.1.2 Caractérisation (géométrique et paramétrique)

Toutes les cellules de l'architecture correspondent à un modèle géométrique. Il ne tient pas compte de la largeur physique des modules contenus dans la cellule mais met en jeu tous les types de connectique : possibilité de connecter les modules les uns au dessus des autres. Ce modèle permet donc d'optimiser le routage et les interconnexions.

La structure du modèle est complexifiable car il est paramétrique. Sa complexité évolue quand la largeur des canaux et le nombre des entrées de la fonction logique augmentent. En effet le nombre de modules de la cellule croît exponentiellement en fonction de ces deux paramètres : le nombre de multiplexeurs contrôlant les ressources de routage dans les 4 directions est fonction de la largeur et celui des multiplexeurs alimentant la fonction logique équivaut au nombre des entrées de cette dernière.

De manière générale, si la fonction contient k entrées et s sorties, et que la largeur des canaux est x, les caractéristiques de module de routage (taille, ...) peuvent être bien définies. En production, La cellule contient k multiplexeurs d'entrées de types $4x \rightarrow 1$ et $4x$ multiplexeurs de sorties de types $3 + s \rightarrow 1$ (chaque multiplexeur de sortie contient à l'entrée toutes les sorties de la fonction et 3 entrées directionnelles). En consommation, une sortie labelisée d'une cellule est connectée à tous les multiplexeurs d'entrées et à 3 multiplexeurs de sortie d'une autre cellule . Donc elle se connecte à $3 + k$ multiplexeurs d'une cellule voisine.

4.4.2 Description dans Madeo

La structuration de l'architecture est hiérarchique. La chaîne de description est définie par un pavage de cellules, une cellule et ses éléments atomiques. Chaque cellule logique est un composite disposant une fonction (LUT) et deux types de multiplexeurs : les multiplexeurs contrôlant les ressources d'entrée et ceux commandant les ressources de sortie. Les éléments atomiques présentent une série d'alternatives qui peut être générée automatiquement en fonction de différents paramètres. Madeo fournit une description d'architecture contenant ce type d'éléments. Le but est d'automatiser la génération des méthodes comportementales correspondant aux éléments pour générer les PLAs implémentables sur les tuiles NASICs.

4.4.2.1 Architecture programmée

L'architecture est grammaticalement décrite dans un langage spécifique dans lequel le mécanisme d'agglomérat englobant les éléments atomiques et leur motif d'interconnexion dans la tuile sont précisés. Les multiplexeurs commandent les entrées et les sorties de la LUT :

- chaque entrée de l'unité fonctionnelle est associée à un multiplexeur. La table 4.3 présente un exemple de description grammaticale d'un tel multiplexeur.
- chaque sortie de l'unité fonctionnelle est associée à plusieurs multiplexeurs. La table 4.4 illustre le code d'un multiplexeur de sortie.

```
(
  (MULTIPLEXER
   (INPUTS  N S E W)
   (OUTPUT X1)
   )"END of MULTIPLEXER"
NAMED Mux_X1 )" end of  NAMED"
```

TAB. 4.1 – Code de définition d'un multiplexeur d'entrée. La largeur de canal est fixée à 1 et la fonction logique a 4 entrées.

```
(
  (MULTIPLEXER
   (INPUTS  N E W outf)
   (OUTPUT  Nout)
   )"END of MULTIPLEXER"
NAMED Mux_North )" end of  NAMED"
```

TAB. 4.2 – Code de multiplexeur de sortie. Une seul sortie notée outf de la fonction logique.

Les seules ressources qui ne sont pas considérées dans le modèle sont celles qui concernent les signaux de contrôle (le signal d'horloge, le signal d'effacement communs à toutes les cellules), les signaux chargés de la gestion de l'horloge globale de l'architecture.

En outre, la définition d'une LUT correspond à une fonction dans laquelle une liste des entrées et des sorties est indiquée.

Pour programmer une cellule selon la variabilité des paramètres, le principe envisagé est de grouper toutes les ressources de routage d'entrée ou de sortie de la cellule sous forme de métamodules (extension de la grammaire voire section 1.2.6). Les multiplexeurs d'entrée sont transformés en un métmultiplexeur qui prend en entrée toutes les $4x * k$ entrées des multiplexeurs d'entrée et produit plusieurs sorties correspondant aux k entrées de la fonction logique. Le même principe est appliqué aux multiplexeurs de sortie où ils sont symbolisés par un métmultiplexeur de $4x(3 + s)$ entrées pour une sortie de largeur $4x$. Cette formulation des métmodules ne modifie en rien la manière dont les objets sont maillés entre eux. Elle apporte une simplification représentative et souple de la cellule

pour des valeurs élevées des paramètres. La figure 4.10 illustre le type de couplage de multiplexeurs. Le métmultiplexeur envisagé (voir figure 4.10(d)) se définit comme une ressource disposant d'un nombre paramétrable de groupes de canaux à l'entrée, d'un port de sélection et d'un nombre paramétrable de sorties. A chaque groupe de canaux doit correspondre une sortie précise, déterminée par le port de sélection : un seul canal d'un groupe donné est connecté au canal de sortie correspondant à une sortie. Les tables 4.3 et 4.4 illustrent le code de tels métmultiplexeurs.

```
(
  (METAMULTIPLEXER
   (INPUTS  E1 E2 .... E4x*k)
   (OUTPUT  S1 S2 .... Sk)
  )"END of METAMULTIPLEXER"
NAMED MetaMuxFunctional)"end of NAMED"
```

TAB. 4.3 – Code du métamultiplexeur fonctionnel. Il contient $4x * k$ entrées et k sorties correspondant aux entrées de la fonction logique (extension de la grammaire voir la section 1.2.6).

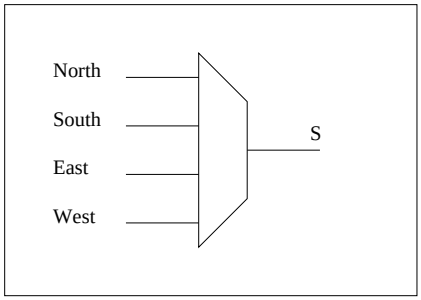
```
(
  (METAMULTIPLEXER
   (INPUTS  E1 E1 .... E4x(3+s))
   (OUTPUT  S1 S2 .... S4x)
  )"END of METAMULTIPLEXER"
NAMED MetaMuxOut)"end of NAMED"
```

TAB. 4.4 – Code du métamultiplexeur de sortie. Il contient $4x(3+s)$ entrées et $4x$ sorties équitablement orientées dans chaque direction.

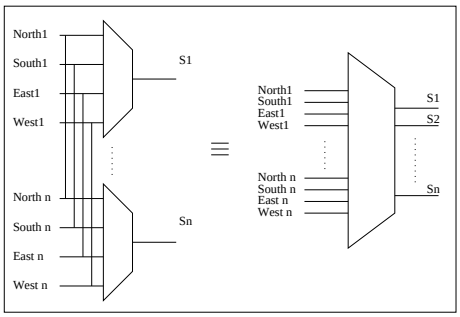
La description de l'architecture est alors annotée de façon à produire une représentation graphique comme présentée dans la figure 4.11.

4.4.2.2 Architecture implémentable

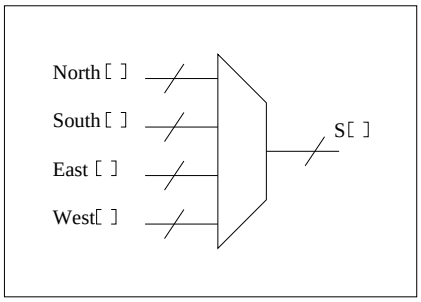
C'est une étape qui consiste à automatiser le tracé des cellules du NFPGA sur les tuiles. Dans ce cas, nous considérons les tuiles comme des cibles architecturales dans lesquelles le NFPGA apparaît comme une application à implémenter. L'écriture du code décrivant tous les éléments architecturaux devient nécessaire. Le paramètrage se traduit par une amplitude de 0 à $2^n - 1$ valeurs possibles pour représenter les variables. L'architecture finale se présente comme un assemblage de fichiers de PLAs.



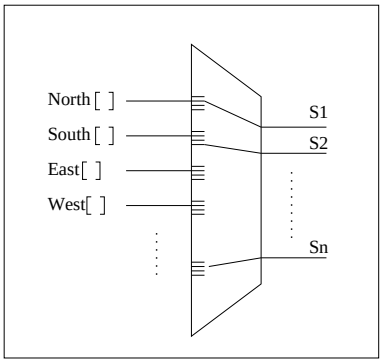
(a) Concept du multiplexeur classique. Les entrées et la sortie en un fil.



(b) type d'assemblage classique. La combinaison de connexions se fait au niveau des entrées.



(c) Passage du concept de fil au concept de canaux. Les entrées et les sorties sont des canaux.



(d) Multiplexeur envisagé. A chaque canal de sortie correspond un groupe des canaux d'entrée sélectionnables.

FIG. 4.10 – Transformation des multiplexeurs en métamultiplexeurs.

4.4.3 Extension de Madeo

Une classe smalltalk du NFPGA a été définie. Elle met en évidence un ensemble de méthodes qui caractérise les modules d'une cellule du NFPGA. L'ajout de nouveaux éléments comme les multiplexeurs vectoriels pour implémenter le domaine de routage dans la cellule contribue l'enrichissement de l'outils dans sa représentation du modèle architectural. Cette représentation faite à l'image de celle du WISP-0 permet en plus de générer les PLAs et d'implémenter les tuiles du modèle.

Pour garantir une conformité entre les deux approches d'architectures (architecture implémentée et architecture programmée), un générateur est nécessaire. La mise en œuvre d'un tel générateur offre deux avantages intéressants. D'une part elle élimine le risque de divergence pouvant survenir entre les deux niveaux de représentation du NFPGA lors d'une re-écriture manuelle et d'autre part, elle permet de réduire le temps de cycle, favorisant ainsi la prospection architecturale.

Etant donné une description figée de l'architecture, deux approches sont possibles pour la description de l'application. La première alternative est de générer directement des

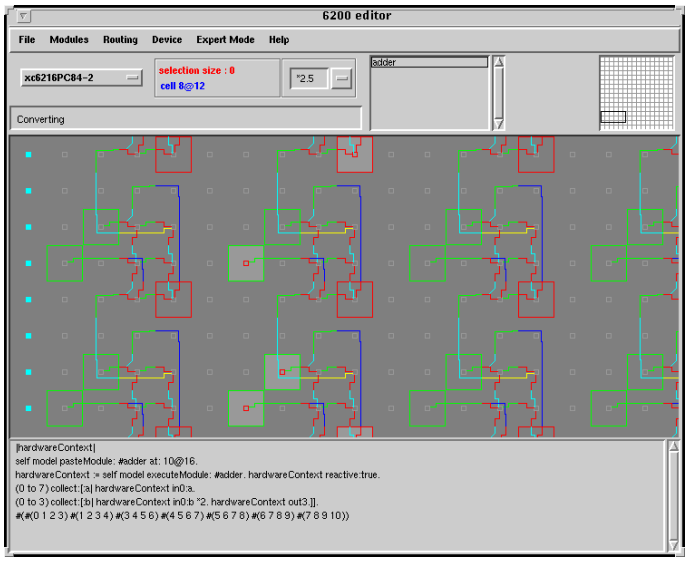


FIG. 4.11 – Vue d'une application placée et routée dans Madeo sur le XC6216 *(les entrées et sorties s'appuient sur l'interface FASTMAP (carrés verts)).*

graphes de fichiers PLA. La seconde est de tirer profit de la synthèse comportementale de Madeo générer un code Smalltalk couplé à un typage. cette dernière approche est préférée car elle permet d'obtenir une meilleure lisibilité du résultat, bien qu'elle comporte certaines manipulations fastidieuses dues à l'usage de la logique complémentée.

4.5 Implémentation du NFPGA

Les éléments de l'architecture étant organisés de manière hiérarchique, les règles de mise en œuvre agissent à deux niveaux. D'une part, elles décrivent le mode de visite de la description de l'architecture, et d'autre part, elles explicitent le code à produire pour chaque élément.

4.5.1 Eléments composites

Ces éléments sont mis en œuvre dans une tuile. Les sous éléments se décrivent et leur assemblage est maintenu par l'emploi des variables intermédiaires produites en sortie d'un élément et utilisées en entrée d'un autre élément.

4.5.2 Eléments atomiques

Ils sont implémentés suivant les règles spécifiques et sont décrits sous forme de méthodes. Ils sont alors appelés pour produire un composite.

Multiplexeur Le multiplexeur de base décrit est le multiplexeur de type 4 vers 1. La méthode de description définit 5 arguments en entrée (une adresse avec une valeur assignée à la sortie et 4 valeurs représentant les signaux). La table 4.5 présente le code sous forme d'une série d'alternatives : une valeur (aValue) est retournée en fonction de la valeur de l'adresse (anAdr).

```
mux41Internal: anAdr i0: aValue0 i1: aValue1 i2: aValue2 i3: aValue
        ^true = (anAdr = 0)
                ifTrue: [aValue0]
                ifFalse: [
                        anAdr = 1
                                ifTrue: [aValue1]
                                ifFalse: [
                                    anAdr = 2
                                      ifTrue: [aValue2]
                                      ifFalse: [aValue3]
                                           ]
                           ]
```

TAB. 4.5 – Code d'un multiplexeur.

Décodeur Il traduit l'information binaire sur ses lignes et l'utilise pour activer une des lignes correspondant à une sortie. Le code est présentée dans la table 4.6 : le bit dont l'index est donné en argument(anAdr) est retourné.

```
decoder: anAdr

^ 2 raisedTo: anAdr
```

TAB. 4.6 – Code d'un décodeur $4 \rightarrow 1$.

LUT C'est un tableau de valeurs prédéfinies accessibles via l'adresse fournie en entrée. La table 4.7 définit le code d'accès aux valeurs de la mémoire (array) en connaissant leurs adresses (adr).

4.5.3 Exemple de mise en œuvre

Cette mise en œuvre de la cellule de NFPGA est divisée en deux implémentations : l'implémentation du domaine de routage et celle de l'unité fonctionnelle.

```
funtionInternalx0: aValue0 x1: aValue1 x2: aValue2 x3: aValue3 ...
                   xk-1: aValuek-1
    |adr|
adr := (aValue0 * (2 raisedTo: 0)) +
       (aValue1 * (2 raisedTo: 1)) +
       (aValue2 * (2 raisedTo: 2)) +
       (aValue3 * (2 raisedTo: 3)) +
                  . . .
       (aValuek-1 * (2 raisedTo: k-1)).

^ array at: adr + 1
```

TAB. 4.7 – Code d'une k-LUT

4.5.3.1 Domaine de routage

L'implémentation regroupe deux mises en œuvre : la mise en œuvre du routage à l'entrée et celle du routage à la sortie. Les deux mises enœuvre sont fonctionnellement identiques mais les singularités résident dans le nombre de multiplexeurs et au niveau de connexions à la sortie. Le nombre de multiplexeurs est fixé à 4 dans la première mais variable dans la dernière. Le nombre de multiplexeurs dans les deux mises en œuvre est le même. Les sorties de la première sont liées à la LUT et celles de la seconde sont réservées pour la communication entre les cellules (ou tuiles implémentant les cellules). L'annexe A.2 montre le code définissant le routage de sortie. Il présente 4 multiplexeurs. La configuration est donc fixée à 8 bits, 2 bits pour chaque multiplexeur. Chaque multiplexeur est identifié par une adresse correspondant à sa configuration (paramètre adrMux). Le code de routage à l'entrée est le même sans le paramètre aValueF. Ce paramètre est remplacé, tour à tour dans le code, par aValueW, aValueE, aValueS et aValueN.

L'implémentation réalisée pour le routage de sortie est montré dans la figure 4.12. Cette figure montre une zone de quatre blocs en escalier qui repésente quatre multiplexeur. Au dessus de cette zone, il y a la zone de sélection des entrées de la tuile.

4.5.3.2 Unité fonctionnelle

Le code fournit pour une 3-LUT est présenté dans la table 4.9. Il prend en entrée trois bits et retourne un seul. La table 4.9 illustre le code du PLA produit pour cette LUT à trois entrées. Les premiers bits correspondent à la configuration de la LUT, les trois bits suivants aux signaux d'adresse issus des multiplexeurs fonctionnels. Les deux derniers bits représentent la sortie codée sur un bit et son complémentaire.

La figure 4.13 montre la localisation des différentes parties de la LUT à 3 entrées implémentée sur une tuile de NASIC.

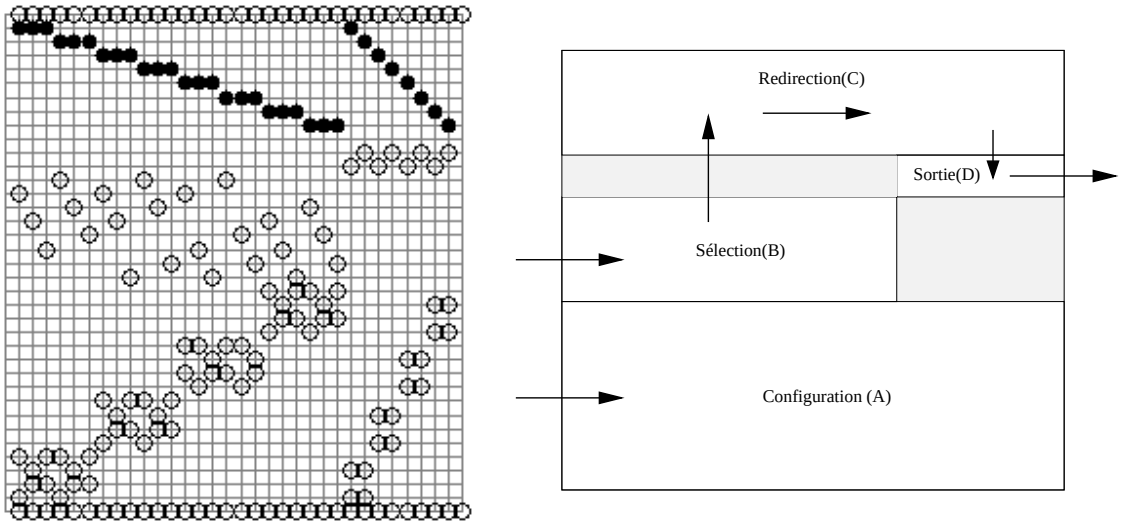


FIG. 4.12 – Layout du domaine de routage : mise en œuvre de 4 multiplexeurs de sortie *(à gauche la localisation de différentes parties de la logique dans la tuile).*

```
configLUTInternal: aValue input: a input: b input: c

^aValue bitAt: (a*4) + (b*2) + 1.
```

TAB. 4.8 – Code produit pour représenter la LUT.

Partie A **et** $A1$ L'écriture de la configuration est en deux temps. La partie A prend en entrée soit les bits de poids fort de la configuration soit les bits de poids faible. La sous partie $A1$ prend en entrée 2 bits activant la zone de mémoire dans laquelle les bits seront écrits. cette méthode permet de minimiser l'effet diagonal dans le cas où l'ensemble de la configuration serait prise en entrée.

Partie B La configuration est stockée sur nanofil [67] et prise en entrée par la partie C.

Partie C Les valeurs de la mémoire sont lues et renvoyées vers la partie D.

Partie D **et** $D1$ La sélection de la valeur dans la mémoire est déterminée par l'adresse donnée en entrée de la LUT (partie D). La partie $D1$ extrait le signal de sortie et son complémentaire de la LUT.

```
.i 6
.o 2
-------0 000 01
------0- 001 01
-----0-- 010 01
----0--- 011 01
---0---- 100 01
--0----- 101 01
-0------ 110 01
0------- 111 01
-------1 000 10
------1- 001 10
-----1-- 010 10
----1--- 011 10
---1---- 100 10
--1----- 101 10
-1------ 110 10
1------- 111 10
.e
```

Tab. 4.9 – Code PLA de la LUT à 3 entrées.

4.6 Passage au multicontexte

Dans le cadre des technologies à support configurable, la reconfigurabilité d'une architecture est possible en implémentant différents contextes de configuration dans les unités de base, mémoires ROM correspond à différentes fonctions. Chaque unité contient donc une mémoire de configuration divisée en plusieurs contextes qui sont choisis pour spécifier une fonction particulière pendant une exécution. La reconfiguration d'une telle unité consiste à changer de contexte de configuration selon un mécanisme qui permet de le pointer et de le rendre actif. C'est une commutation de configurations : un ensemble de configurations est stocké dans un FPGA et la configuration exécuter pourrait être précisée dans un cycle d'horloge. Les prototypes de puces avec une telle disposition ont été développés [33, 77]. La notion de registre de configuration introduite dans la cellule de NFPGA représente une mémoire de stockage de configurations pour les ressources logiques et de routages.

Une caractéristique importante dans cette option de reconfigurabilité est que le temps de configuration est considérablement faible puisque les configurations sont contenues d'avance sur le support. Dans ce cas, le support ne nécessite qu'un cycle d'horloge visiblement contrôlé par un microfil pour passer d'une configuration à une autre.

Pour tirer bénéfice de cette souplesse et améliorer le rendement de la surface d'une tuile (le ratio entre la surface occupée par la grille de nanofils et la surface totale de la tuile) [68], nous introduisons les contextes de configuration dans la cellule de l'architecture

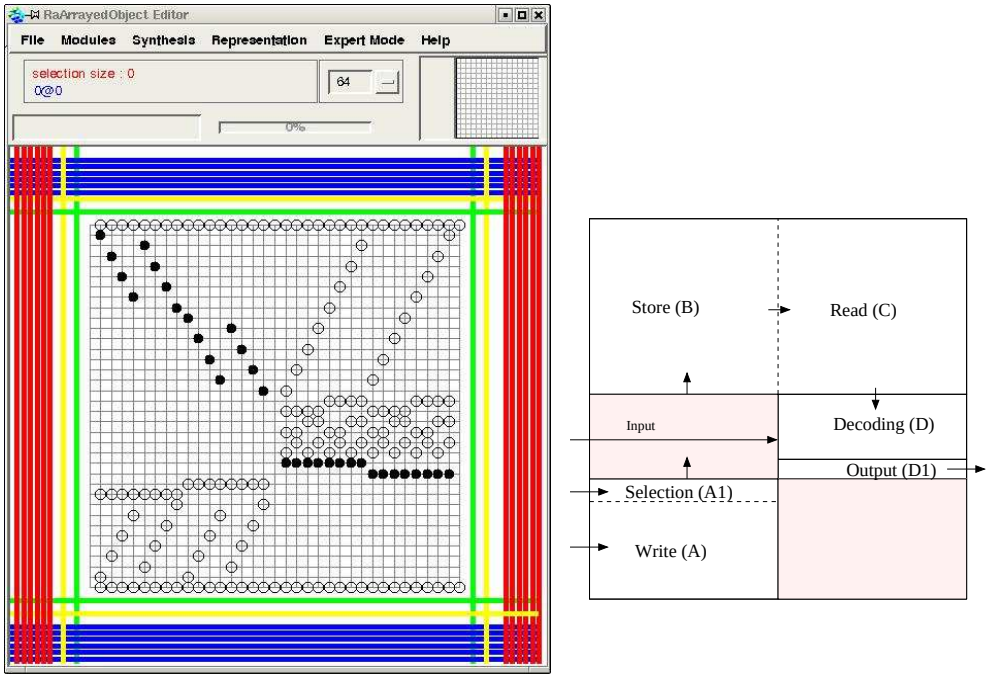


FIG. 4.13 – Vue dans Madeo d'une LUT implémentée [76] : *(la partie droite illustre le découpage schématique de la tuile en plusieurs sous parties).*

NFPGA et calculons le coût spatial de celle-ci. Cette technique de contextes de configuration permet de réaliser une conception compacte et dense. DeHon et al ont montré que les implémentation multicontextes sont plus efficaces que les implémentations avec un seul contexte si la surface et la densité d'intégration correspondante sont optimales [88]. Autrement dit, un FPGA dans lequel une cellule logique contient plusieurs contextes de configuration a besoin de quelques cellules pour implémenter une application qu'un FPGA dans lequel une cellule a un contexte unique. Cela permet de croître le rendement de surface dans la conception NASIC puisque la surface dédiée aux microfils (contrôle et alimentation en tension) est beaucoup plus importante que la surface de calcul.

Pour évaluer l'impact en terme de coût spatial du multicontexte, nous examinons différents cas de conception de celui-ci selon la manière dont le choix de l'indice de contexte est opéré dans la cellule.

4.6.1 Aspects de l'architecture de routage du NFPGA

L'architecture de routage global se compose essentiellement de microfils et spécifie la largeur de ces microcanaux. Cette largeur dépend majoritairement de la taille de la grille logique d'une nanotuile. Les aspects tels que le grain de commutation de la configuration font intervenir des paramètres dans la conception de l'architecture. Outre les microfils réservés pour le contrôle et l'alimentation en puissance d'une tuile, les microfils gérant la configuration sont prévus : (1) les fils représentant l'indice de contexte pour sélectionner

une configuration donnée dans la cellule, (2) les fils d'adresses utiles pour choisir une cellule ou un groupe de cellules à configurer et les fils de configuration qui permettent de fixer la donnée de configuration à travers une jonction de nanofils dans la grille lorsque une adresse est donnée.

Le premier paramètre est inclus dans l'ensemble de microfils contrôlant la logique d'une tuile. Le second paramètre qui permet de configurer partiellement l'architecture dépend du nombre de cellules contenues dans l'architecture. Dans ce dernier cas, le nombre de microfils est au moins égal au logarithme à base deux du nombre de cellules logiques dans la puce de NFPGA.

4.6.1.1 Etude de routabilité

A l'échelle de la cellule, les détails sur le routage sont représentés dans la figure 4.14 qui définit la manière dont les briques élémentaires sont connectées entre elles dans une tuile. Une telle tuile reçoit des signaux venant des tuiles voisines à travers des canaux de x voies. Elle peut les utiliser à travers la LUT (connexions établies entre l'interface d'entrée et les multiplexeurs fonctionnels) ou les diriger vers les autres tuiles (connexions établies entre l'interface d'entrée et les multiplexeurs de sorties). Les connexions existantes entre les multiplexeurs et la LUT sont fixées. Il existe 6 types de connexions possibles. une connexion est caractérisée par deux côtés de la tuile. Les types de connexions liant deux côtés opposés de la tuile sont directes et les autres sont courbes. Dans une telle configuration de connexions, le structure du routage dans la tuile est similaire à celle d'un bloc d'aiguilleurs classique. Dans ce cas, la conception du routage peut nécessiter plusieurs topologies dépendantes d'une application donnée.

Le coût du routage dans un FPGA est très élevé. Il consomme une bonne partie de la surface et est responsable de délais d'un circuit. Deux facteurs sont à l'origine du coût en surface : la surface d'aiguilleurs et la surface de stockage de configuration. Visiblement, l'utilisation d'aiguilleurs moléculaires, du stockage temporaire de données sur les nanofils et l'implémentation du routage local et de la logique dans une tuile peuvent considérablement diminue l'impact de ces facteurs. Les délais dominant proviennent des microfils (longueur de fil) et l'activité de zones de routage dans une nanogrille. Les analyses de surface et de performances de FPGAs utilisant les nanofils et les aiguilleurs moléculaires dans la structure de routage ont été présentées [35, 78]. Elles montrent la possibilité de réduire la surface de routage jusqu'à 75% et d'améliorer les performances.

Une connexion dans une tuile est établie par programmation. La flexibilité d'une telle connexion est clairement la clé de la routabilité de la tuile. Le choix d'une topologie de connexion adéquate possède un effet signifiant sur les performances de la tuile (surface, délai et routabilité). Cela nous amène à considérer deux topologies pour leurs performances acceptables : la topologie complètement connectée et la topologie disjointe.

Topologie complètement connectée Cette topologie garantit la connectivité totale dans une tuile. Chaque entrée peut potentiellement se connecter avec toutes les sorties possibles de la tuile. De part le nombre important des points d'interconnexion, le motif

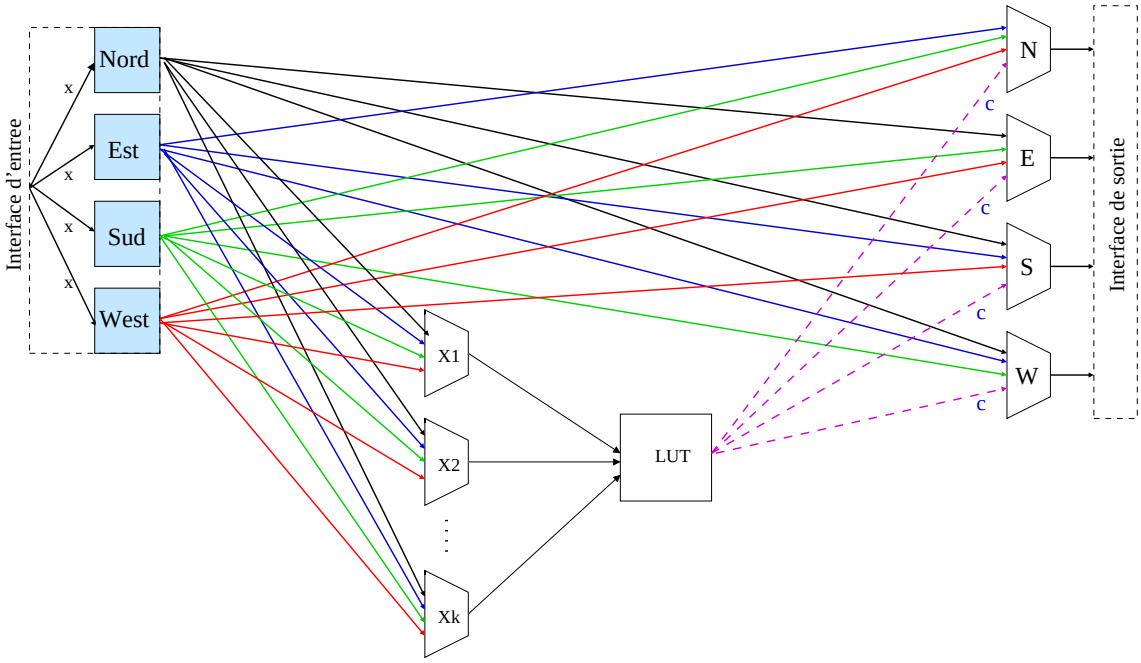


FIG. 4.14 – Configuration de connexions dans la tuile d'une cellule.

devient complexe dans la mise en œuvre. La taille de la tuile devient très importante pour des applications concrètes qui nécessitent des milliers de connexions. Le coût en surface pour cette topologie est déraisonnable. Un bénéfice réside dans la simplification des algorithmes de routage dans les outils. La topologie est plus routable que toutes les autres solutions.

Une telle topologie peut être envisagée pour les petites applications pour lesquelles une tuile de la cellule du NFPGA nécessiterait peu d'éléments de base à connecter (multiplexeur, LUTs). Cela permet d'avoir une solution optimale qui propose un compromis entre la routabilité maximale et le coût en surface minimal.

Topologie disjointe C'est une topologie qui permet de connecter une entrée d'un côté d'une tuile aux sorties de même label de trois autres côtés de la tuile [45]. Chaque entrée labelisée possède donc un domaine de routage limité qui regroupe toutes les sorties possibles de même label. Le choix de routage se restreint à établir une connexion par programmation entre deux broches de même étiquette dans la tuile. Cela partitionne le routage en sous domaines, ce qui réduit la routabilité et le coût en surface. Les outils de routages doivent être adaptés pour assurer une bonne routabilité à travers les sous domaines de différentes tuiles.

Lorsque le nombre d'entrées de multiplexeurs fonctionnels diminue (un multiplexeur X_i prend $4x/k$ entrées au lieu de $4x$, figure 4.14), le coût en surface décroît car certaines connexions entre l'interface d'entrée et les multiplexeurs fonctionnels doivent être supprimées. Le coût spatial de l'implémentation du routage à l'entrée est réduit. Cette réduction peut s'avérer faible sur l'ensemble de la tuile si le nombre de multiplexeurs est

important (le nombre d'entrée d'une LUT est déterminant pour la surface de la tuile). Par ailleurs, la réduction du nombre de connexions entre l'interface d'entrée et les multiplexeurs fonctionnels a un effet capital sur la routabilité de la tuile : le choix d'établir certaines connexions directes peut augmenter la routabilité dans les sous domaines de routage. Ce gain en routabilité peut également être limitée par le multicontexte si le nombre de contextes devient important car il caractérise les entrées des multiplexeurs de sortie.

4.6.1.2 Configuration du NFPGA

La configuration est le processus de changement des données de programmation d'une conception spécifique dans un FPGA pour définir la fonctionnalité de blocs logiques et de l'interconnexion. Le FPGA utilise plusieurs bits de configuration par bloc(ou par une tuile dans le cadre de NASICs). Chaque bit de configuration peut définir l'état d'un aiguilleur du registre de configuration contrôlant une entrée du multiplexeur ou un bit de la LUT. Pour la conception de NFPGA, deux niveaux de configuration sont envisagés pour spécifier sa fonctionnalité : la configuration au niveau technologique et la configuration au niveau architectural. Dans les deux cas, il faut un décodage pour adresser tous les nanofils d'une tuile afin d'accéder à toutes les jonctions programmables. Les interfaces CMOS/NANO [98] sont envisagées pour réaliser le décodage et les contacts (nanovias) entre microfils et nanofils au moyen de dispositifs moléculaires. Chaque microfil disposent de zones de dopage qui peuvent permettre de fixer plusieurs nanofils. Un bit d'adresse est caractérisé par la présence ou l'absence de contact entre microfil et un nanofil au niveau de zone de dopage. Cela permet un espace d'adressage large pour les nanofils avec un petit nombre de microfils [22, 53, 83].

Niveau technologique L'approche de configuration des dispositifs actifs dans une tuile est similaire à celle des antifusibles dans la technologie FPGA classique, présentée dans [40]. ces dispositifs logés à chaque interconnexion de deux nanofils peuvent être programmés en sélectionnant les nanofils avec les microfils de programmation. Les dispositifs sont initialement dans un état neutre où la grille est vide de configuration. Les microfils d'alimentation en puissance sont préchargés pour fournir une différence de tension qui équivaut à la tension de programmation appropriée (chacune des extrémités d'un nanofil étant connectée un microfil qui fournit une source de tension, l'intersection de deux nanofils reçoit une tension qui est égale à la somme des tensions de ces deux nanofils). Donc, l'application de cette tension de programmation à travers la jonction adressée permet de fixer l'état du dispositif.

Niveau architectural C'est une configuration qui permet d'écrire les données dans les registres de configuration. Elle correspond à la configuration des ressources (logiques et routages) pour implémenter la conception d'un utilisateur de FPGA. Dans ce cas, un certain nombre de mécanismes de configuration sont prévus : un mode de configuration (série ou parallèle) qui explicite la manière dont les données de configuration des registres sont envoyées à travers la puce, une horloge de configuration qui dépend du mode et une

activation de la configuration (basculer d'une configuration à une autre après écriture de données de configuration). Le second mécanisme fait partie du domaine de contrôle.

Le choix du mode configuration est caractérisé par les bits de configuration (les micro-fils spécifiques sont dédiés aux bits de configuration de registres). L'approche pour mettre les données de configuration dans les registres peut se réaliser en deux étapes. La première étape consiste à remplir bit par bit les registres d'une tuile. La seconde permet de faire basculer le contenu des registres d'une tuile entière dans une autre tuile. Cela peut se dérouler en parallèle puisque l'ensemble de bits peut être envoyé d'un coup.

4.6.2 Types de multicontextes

4.6.2.1 Multicontexte à sortie unique

Dans ce cas de figure 4.15, le calcul englobe une LUT à plusieurs sorties correspondant au nombre de contextes de configuration. Toutes les sorties correspondant à chaque contexte de configuration sont connectées à un multiplexeur contrôlé par les signaux de sélection de contexte. Une sortie de contexte de configuration est active lorsque les fils (signaux) correspondant au numéro du contexte sont conduits. La sortie active est alors connectée aux multiplexeurs de routage de sortie de la cellule.

Pour l'implémentation de la tuile, le coût spatial supplémentaire est apporté par le coût spatial de $C-1$ contextes de configuration (C est le nombre total de contextes dans une cellule), le coût spatial du multiplexeur de sélection du contexte et le coût des fils de contexte (un coût additionnel peut être par la manière dont les composants sont disposées et dans la tuile).

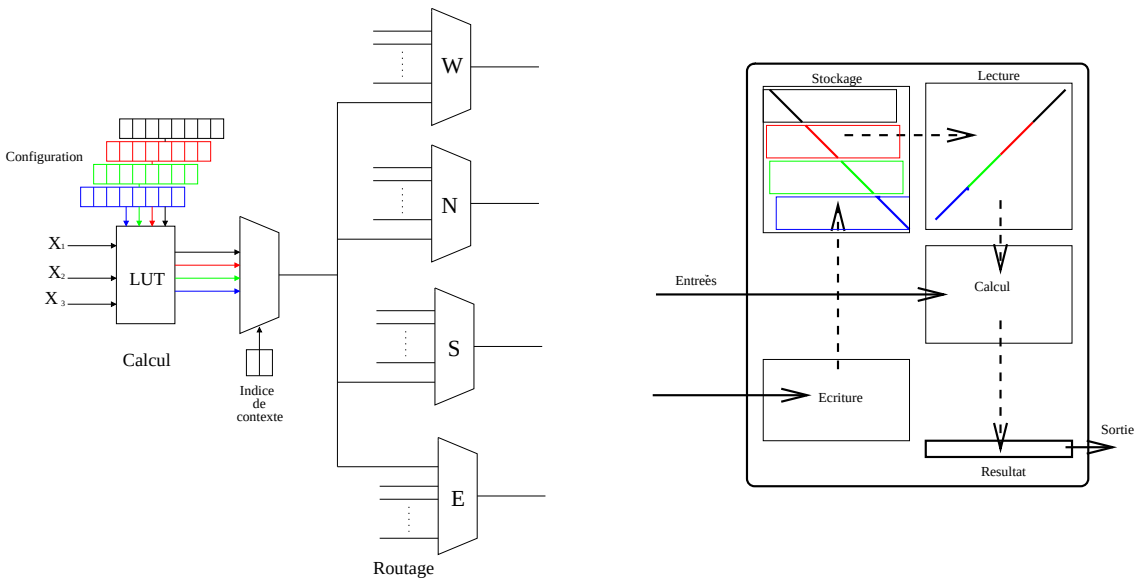


FIG. 4.15 – Schéma structurel de la cellule multicontexte *(utilisation possible d'un multiplexeur pour sélectionner un contexte ; à gauche, une vue schématique d'implémentation).*

LUT La conception de la LUT en tuile de NASICs se compose de 4 grandes parties : écriture, stockage, lecture et calcul. La figure 4.16 illustre la localisation des différentes parties de la k-LUT. Elle contient C contextes de configuration pour changer la fonctionnalité de la cellule à un moment donné. Le coût spatial est fortement dépendant de la manière dont les modules sont implémentés sur une tuile.

La partie A présente le mécanisme d'écriture qui se déroule en deux étapes. La première partie A1 prend en entrée soit le bit de poids fort soit le bit de poids faible de C contextes de configuration. Cette partie A nécessite 2^k nanofils horizontaux et $2^{k+1}C$ nanofils verticaux pour son implémentation. La seconde partie $A1$ prend $2C$ bits pour activer la zone de mémoire dans laquelle les bits doivent être écrits. Elle prend $2C$ nanofils horizontaux et $2^{k+1}C$ nanofils verticaux. Les composantes de cette partie d'écriture sont définies par $\left(2^{k+1}C, 2^k + 2C\right)$.

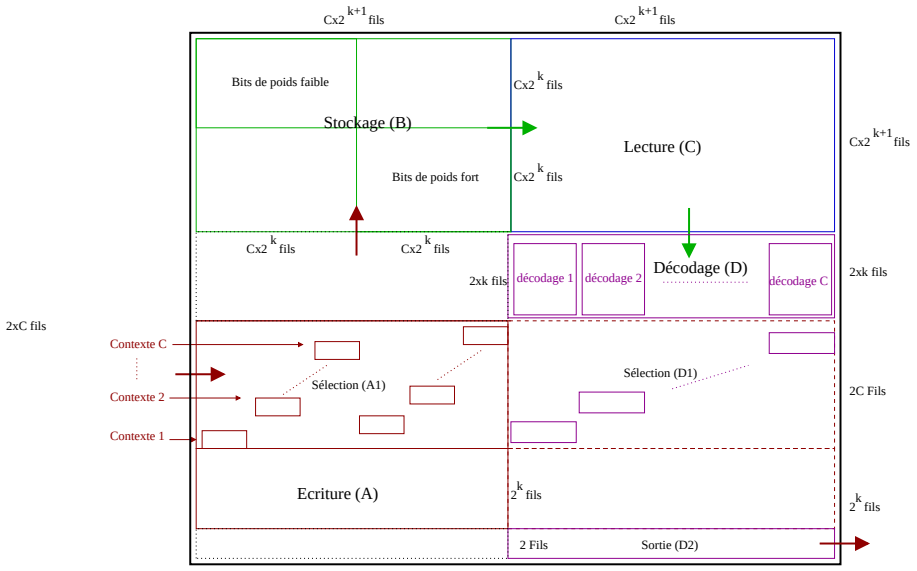


FIG. 4.16 – Méthodologie de détermination des composantes de la LUT *(localisation des différentes parties d'implémentation).*

La partie B constitue le stockage de toutes les configurations des contextes sur les nanofils. L'implémentation du stockage d'un contexte induit un effet diagonal qui se propage dans le cas d'une implémentation multicontexte. Cette partie contribue largement à augmenter le coût en surface de l'implémentation. Cette implémentation est définie par les composantes suivantes $\left(2^{k+1}C, 2^{k+1}C\right)$.

Dans la partie C, les valeurs stockées dans la mémoire sont lues et acheminées vers la partie D pour le décodage. Pour chaque contexte, 2^{k+1} valeurs possibles sont lues (2^k valeurs de bits de poids faible et 2^k valeurs de bits de poids fort). Ces valeurs sont alors envoyées vers la partie D. Globalement, l'implémentation de cette partie de lecture possède les composantes suivantes $\left(2^{k+1}C, 2^{k+1}C\right)$.

La partie D regroupe tous les décodages de C contextes. Cette partie permet de sélectionner une valeur dans la mémoire au moyen d'une adresse selon le contexte de

configuration activé en entrée de la LUT. La partie extrait les signaux de sortie et leurs complémentaires de différents contextes. L'implémentation de cette partie de calcul (décodage) nécessite $2k$ nanofils horizontaux pour les entrées de la LUT et $2^{k+1}C$ nanofils verticaux pour adresser chaque valeur à lire. La partie D1 permet d'activer l'un de contextes à la sortie (partie D2). C'est le prolongement de la sélection A1. Les composantes de ces deux parties sont les mêmes. La partie D2 présente 2 nanofils horizontaux pour la sortie et son complémentaire. Les composantes de l'ensemble de la partie D sont définies par $\left(2^{k+1}C, 2\left(k+C+1\right)+2^{k}\right)$.

Les composantes finales de la LUT dans la disposition présente sont estimées en associant les composantes relatives des modules de son implémentation de même direction entre elles. Le coût en terme de nanofils de la LUT est approximé par les composantes suivantes $\left(2^{k+2}C, 2\left(1+k+C\right)+\left(C-1\right)2^{k+1}\right)$.

Sélection de contexte La sélection de C contextes pour une sortie est réalisée dans la partie D1 de la figure 4.16. Cette partie récupère tous les décodages et rend active une seul décodage en sortie. Les bits de poids faible de ce dernier sont alors invalides et ceux de poids fort valides. Ce mécanisme de sélection de contexte est implémenté à l'image d'un multiplexeur. En définitive, dans ce cas de multicontexte, l'unité fonctionnelle est la partie de la cellule du NFPGA qui varie fortement. Dans ce cas nous pouvons négliger les autres modules. Par conséquent, le coût en surface de la cellule peut se réduire à celui de la LUT (cf section 4.3.2). Les composantes de cette unité sont alors données par les relations suivantes :

$$\begin{aligned} X_{unitef1} &\simeq \left(2^{k+2}C\right) \\ Y_{unitef1} &\simeq 2\left(1+k+C\right)+\left(C-1\right)2^{k+1} \end{aligned} \tag{4.9}$$

Pour définir le coût spatial de cellule du NFPGA multicontexte, nous nous introduisons ces deux composantes dans l'équation 4.8 donnant la surface physique.

4.6.2.2 Multicontexte à plusieurs sorties

Dans cette configuration, toutes les sorties de contextes sont connectées à chaque multiplexeur de sortie de la cellule comme le montre la figure 4.17. Il n'y a pas de multiplexeur pour sélectionner une sortie particulière. L'unité fonctionnelle se présente comme une LUT à plusieurs sorties. Puisque chaque contexte donne lieu à une sortie qui est liée au routage, l'indice de contexte doit être stocké dans le registre de configuration au même titre que la configuration des multiplexeurs. Par conséquent, les modifications de surface sont apportées par l'unité fonctionnelle et les différent multiplexeurs de routage de sortie où la largeur de canal doit être adaptée au nombre de sortie de la LUT.

LUT Les composantes de l'implémentation de l'unité fonctionnelle restent pratiquement les mêmes que celles de l'unité fonctionnelle du multicontexte à sortie unique. Mais la

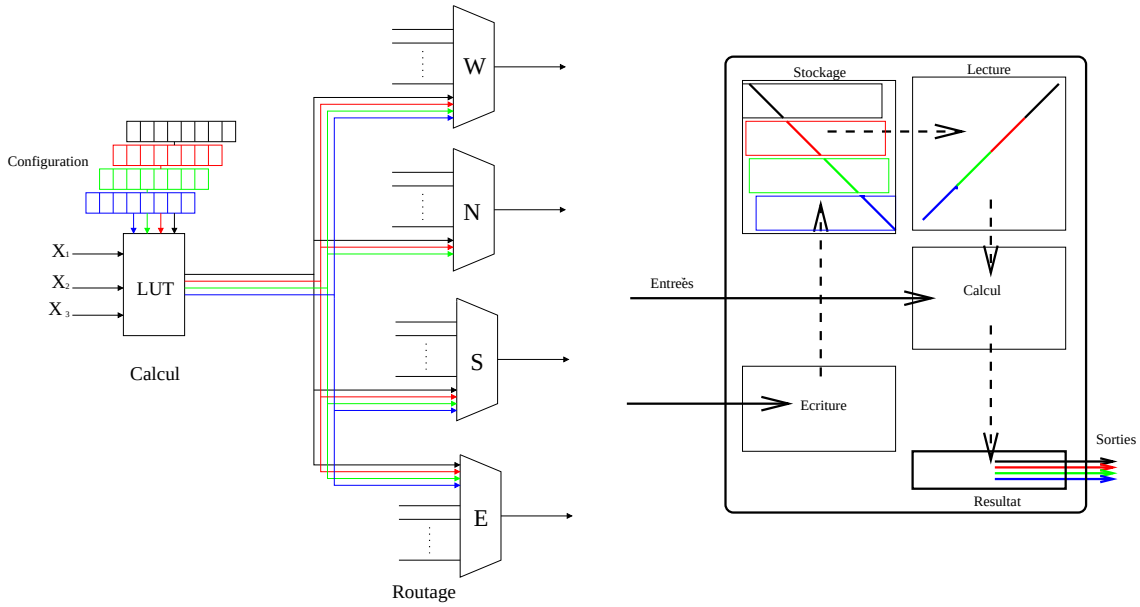


FIG. 4.17 – Configuration du multicontexte avec toutes les sorties de contextes dirigées vers le domaine de routage *(le schéma de gauche illustre une vue d'implémentation).*

partie D1 de la figure 4.16 n'est plus mise en œuvre et la partie D2 contient plusieurs sorties. Il y a alors C sorties et leurs complémentaires. Cela donne lieu à $2C$ nanofils de sortie dans la mise en œuvre de la partie D2. Les composantes finales de la LUT sont alors données par les équations suivantes :

$$\begin{aligned} X_{unitef2} &\simeq \left(2^{k+2}C\right) \\ Y_{unitef2} &\simeq 2\left(k+2C\right)+\left(C-1\right)2^{k+1} \end{aligned} \tag{4.10}$$

4.6.2.3 Multicontexte avec resynthèse logique

Dans ce cas de multicontexte, le contexte de configuration intervient directement dans le calcul logique. Le coût spatial de l'implémentation dépend fortement de la resynthèse dans l'unité fonctionnelle. Pour améliorer la conception de la tuile de la cellule, il est préférable de changer la topologie de l'implémentation. Dans ce cas, la resynthèse devient un outil efficace pour transformer un circuit logique en des circuits équivalents en terme de résultats avec des caractéristiques d'implémentation différentes. La figure 4.18 illustre la disposition des modules dans la cellules.

Ce cas de figure du mulitcontexte donne lieu aux optimisations pour simplifier l'architecture et réduire la partie variable. La simplification de l'architecture contribuerait à la perte de flexibilité de l'architecture. La réduction de la partie variable de l'architecture consisterait à mettre en œuvre une partie de la configuration en dur (spécification d'une partie donnée de la configuration). Le bénéfice attendue est visiblement le gain en surface.

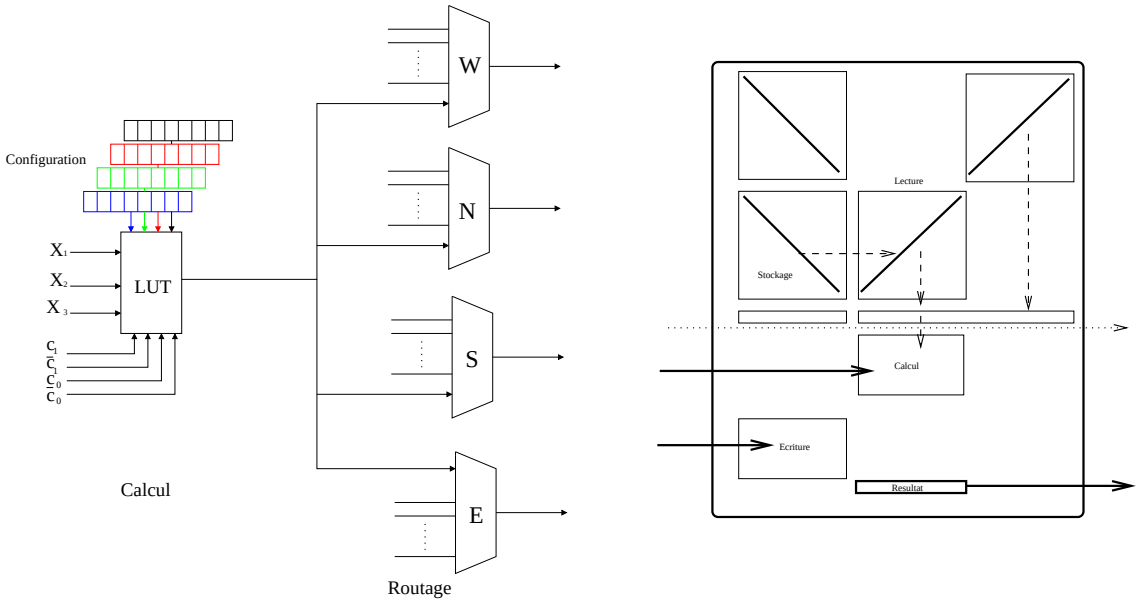


FIG. 4.18 – Aspect schématique du multicontexte avec resynthèse *(le modèle schématique de la vue d'implémentation).*

Le coût spatial de cette configuration de multicontexte ne sera pas calculé car il nécessite de d'implémenter les circuits benchmarks pour mapper les k-LUTs.

4.6.3 Coût du modèle de la surface d'une LUT

L'objectif dans cette section est de mener une étude comparative et évolutive d'une LUT multicontexte dans le cadre de la conception NASIC. Une analyse du coût spatial du modèle d'implémentation de cette dernière selon le contexte technologique prospectif (International Technology Roadmap for Semiconductors[2]) permet d'envisager sa performance.

D'une part, nous utilisons les hypothèses technologiques données dans [2], basées sur l'évolution technologique CMOS pour calculer les coûts en surface de différentes LUTs NASICs. D'autre part, nous confrontons ce modèle de surface de la LUT NASICs à un modèle de la surface d'une LUT classique. Ce dernier modèle considéré a été proposé par DeHon dans [20] pour le prototypage d'un DPGA CMOS avec C contextes de configuration. C'est un modèle simple et linéaire qui est caractérisé par l'équation suivante :

$$A_{LUT} = A_{base} + C * A_{context} \tag{4.11}$$

où A_{LUT} désigne la surface de LUT avec C contextes. Les quantités A_{base} et $A_{context}$ représentent respectivement la surface de la LUT sans contextes et le coût en surface d'un contexte de configuration. Pour une comparaison optimale de deux modèles, le modèle classique considéré s'appuie sur les cellules de confifguration à base de DRAMs puisque,

du point de vue de mise en œuvre de bas niveau (niveau physique), la LUT NASIC possède la même abstraction de conception qu'une LUT à base de cellules DRAMs. Mais le mécanisme de fonctionnement de la LUT NASIC permet de conserver la valeur stockée dans un point de configuration à l'image d'une SRAM. La surface du modèle classique est calculée en considérant les grandeurs suivantes données dans [20] :

$$A_{base} = 544K\lambda^2$$
$$A_{context} = 12K\lambda^2$$
$$K = 1024$$

où λ est la donnée technologique.

Resultats Les courbes de deux configurations de multicontexte mesurées (une sortie ou plusieurs sorties) montrent un coût quasi-identique. En effet pour chaque type de LUTs, comme nous pouvons l'observer dans la figure 4.19, la progression entre les deux surfaces tend à s'égaliser pour un même nombre de contextes. Nous avons mesuré l'effet de l'évolution technologique sur le modèle de la surface d'une LUT NASIC. Nous constatons que plus le facteur technologique est réduit, meilleur est le modèle. cela peut s'explique par la différence de surcoût dû aux microfils de contrôle qui dépend fortement de ce facteur technologique. De plus, la préférence du multicontexte favorise la flexibité dans le routage. Les figures 4.19, 4.20 et 4.21 exposent une telle situation à travers les mesures réalisées pour différents types de LUTs dans chacune de deux configurations de multicontexte.

Nous avons également effectué une comparaison du modèle de surface NASIC avec le modèle de la LUT classique CMOS présenté ci-haut. Les mesures sont réalisées dans le cadre d'une 4-LUT. C'est le type de LUT utilisé pour décrire le modèle classique. Il permet de témoigner une observation réaliste de la méthodologie.

La surface d'une 4-LUT NASIC est considérablement plus faible que celle d'une 4-LUT classique. Les figures 4.22, 4.23 et 4.24 signifient la progression des surfaces de ces deux modéles en fonction du nombre de contextes et de l'évolution technologique. Quand le nombre de contextes de configuration croît, l'aire de la 4-LUT NASIC progresse de manière exponentielle et converge vers la surface du modèle classique. La diminution de la technologie permet d'observer une réduction continuelle de l'écart entre les deux coûts en surface.

4.7 Discussion

Nous avons démontré la faisibilité d'une architecture reconfigurable basée sur les circuits NASICs avec une bonne utilisation de la fabrique (les vues implémentées sont optimales dans Madeo). Malgré le coût en surface excessif de la partie micro, la surface de notre architecture NFPGA est considérablement faible par rapport à celle de FPGAs traditionnels. Par exemple, la partie coûteuse en surface de notre architecture est essentiellement la LUT. La surface de son implémentation peut atteindre 80% de la surface

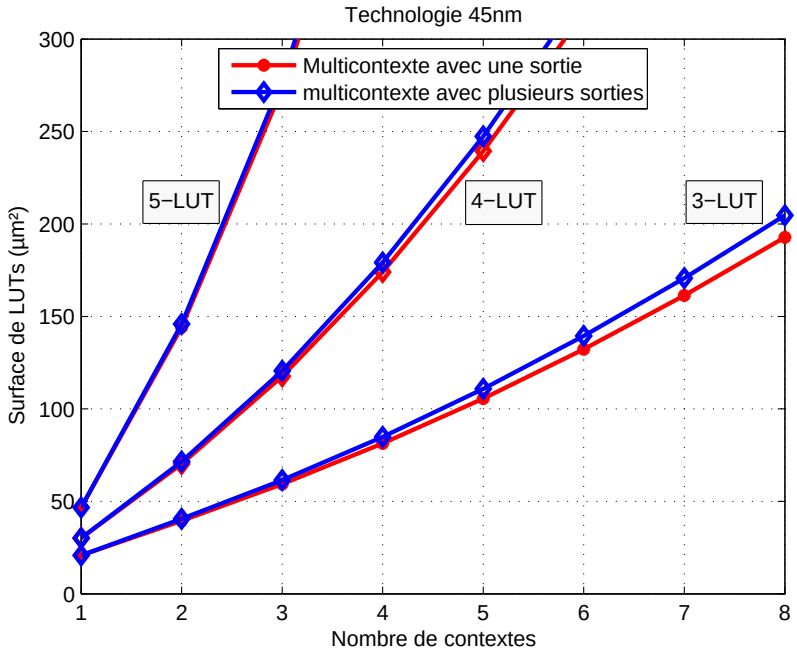


FIG. 4.19 – Surface des LUTs NASICs en fonction du nombre de contextes de configuration : technologie 45nm.

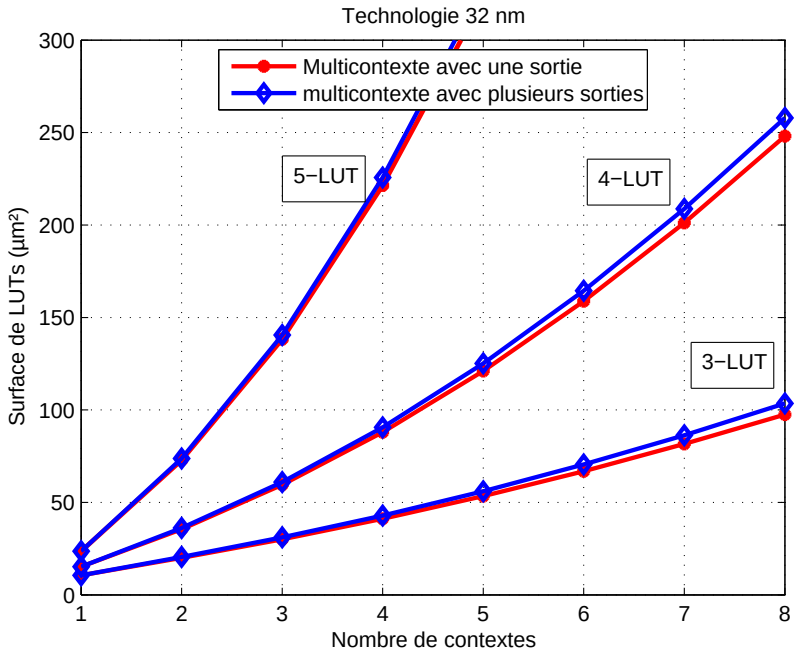


FIG. 4.20 – Surface des LUTs NASICs en fonction du nombre de contextes de configuration : technologie 32 nm.

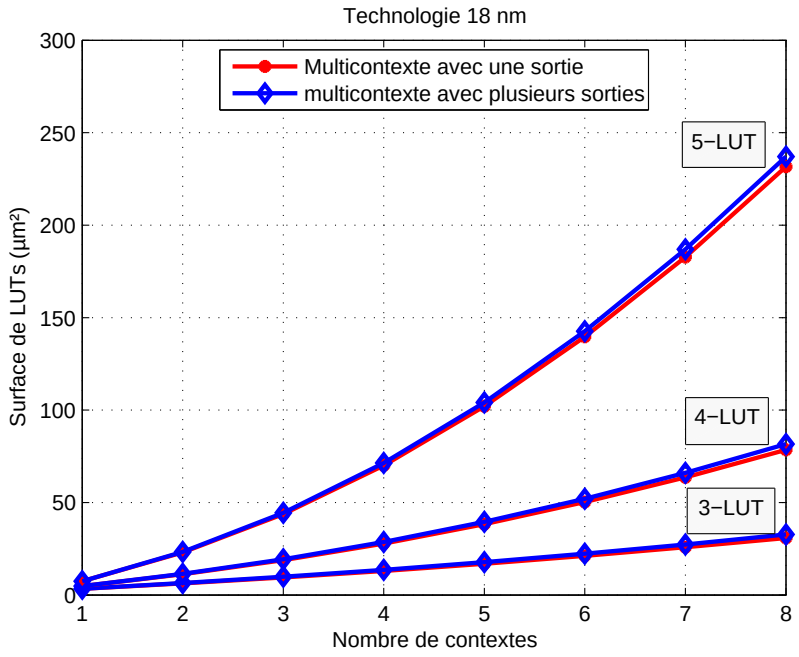


FIG. 4.21 – Surface des LUTs NASICs en fonction du nombre de contextes de configuration : technologie 18 nm.

totale d'une cellule (microfils de contrôle et d'alimentation compris) lorsque la largeur de canaux et le nombre d'entrées de la LUT sont les plus grands possibles (cf figure 4.9). Cette surface dans le cadre du multicontexte est au moins 4 fois inférieure à cellule d'une LUT classique comme le montrent les résultats dans la section 4.6.3.

Dans les FPGAs traditionnels, la portion consommée par le câblage peut facilement atteindre 60 à 70% de la surface de la puce. Le routage représente un point critique du développement d'une application sur ces FPGAs. Ainsi, pour les FPGAs "mer de portes", la capacité limitée de ressources de routage est la raison pour laquelle certaines applications ne peuvent être réalisées. L'augmentation de la capacité de ressources de routage est une solution envisageable à ce problème avec l'évolution technologique. En effet, l'architecture présentée est un exemple d'effort dans ce sens. Elle exhibe deux bénéfices. Le premier se fonde sur la réalisation à l'échelle nanoscopique de la matrice de routage grâce à la présence de registres de configuration. Cela contribue à réduire la surface de la réalisation physique de celle-ci basée sur les cellules SRAM qui sont coûteuses en surface dans les FPGAs classiques. Le second bénéfice réside dans le couplage de la matrice de routage avec la logique dans une tuile. Dans ce cas, l'augmentation logarithmique de ressources de routage améliore le rendement de la densité de nanofils. Cela peut visiblement équilibrer le coût en surface dû à l'usage de microfils pour contrôler et alimenter en tension les tuiles.

D'autres contraintes présentes sur un circuit conditionnent sa réalisation physique. Par exemple, la qualité d'estimation des temps de propagation des signaux dans le cir-

cuit dépend fortement des structures logiques utilisées et de la surface occupée par le circuit selon l'implantation d'une conception. A l'echelle nanoscopique, l'interdépendance entre toutes les contraintes (surface, consommation, délai, intégrité du signal transmis,...) implique des modèles d'estimation différents des modèles classiques. Pour mieux tenir compte des délais d'interconnexion dans l'architecture présentée, il faut s'affranchir de modèles traditionnels d'estimation de délais entre les portes qui peuvent donner une vision pessimiste de celle-ci. Le modèle d'estimation de délais dépendant de l'implantation de la conception NASICs se met en œuvre avec une analyse de la synchronisation qui repose sur une structure en grille de la tuile et la disposition linéaire des dispositifs actifs dans celle-ci.

Pour l'heure, nous nous satisfaisons à une évaluation de l'impact de l'utilisation d'une fabrique NASIC présentée dans [67] montrant qu'une densité de fabrique NASIC de 5% donne une bonne accélaration par rapport à une fabrique CMOS de même surface.

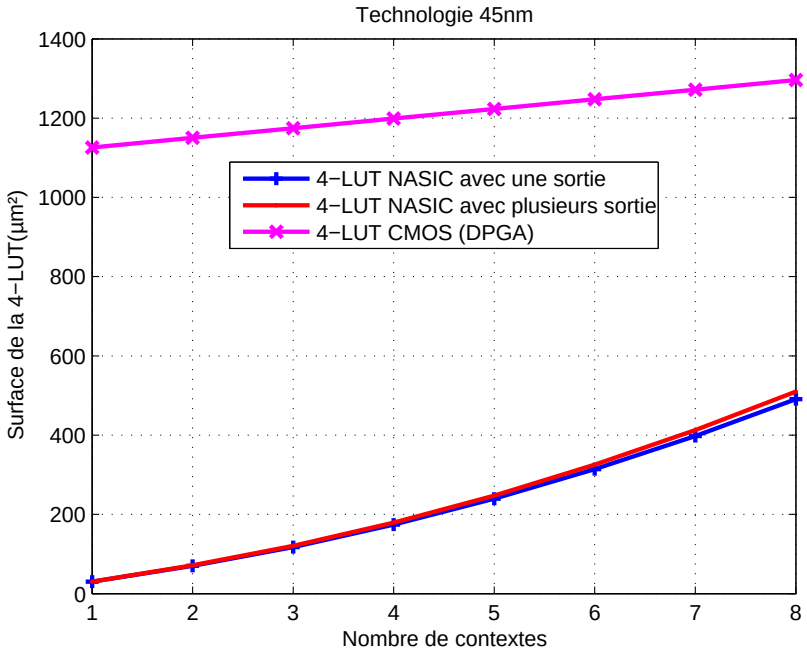


FIG. 4.22 – Technologie 45nm.

4.8 Conclusion

Le chapitre a étudié la possibilité de faire une architecture reconfigurable basée sur un support technologique configurable. Nous avons défini un modèle d'architectures basé sur une architecture classique. Nous avons proposé une implémentation NASIC du modèle et fourni un modèle analytique pour quantifier le coût spatial. Par ailleurs, nous avons étendu l'outil Madeo pour permettre d'implémenter, programmer et calibrer le modèle.

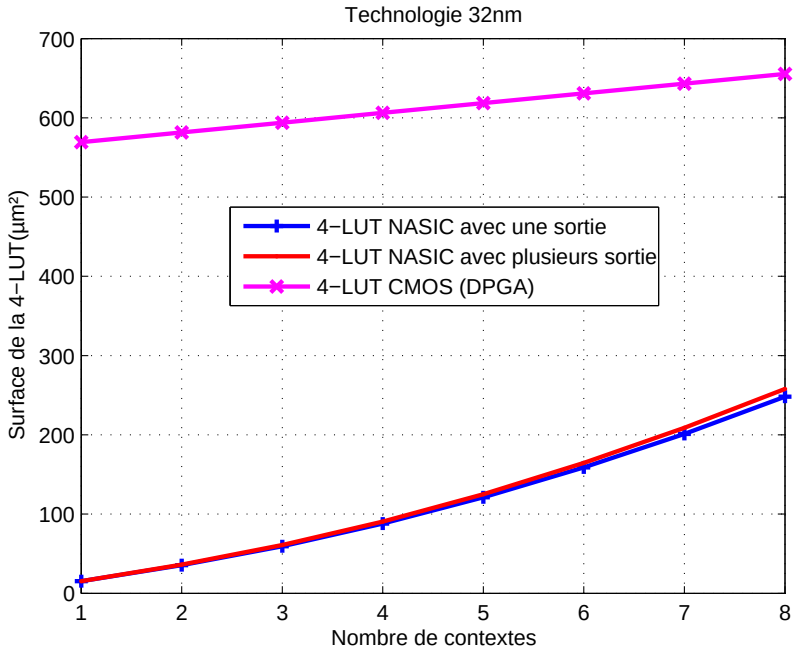


FIG. 4.23 – Technologie 32 nm.

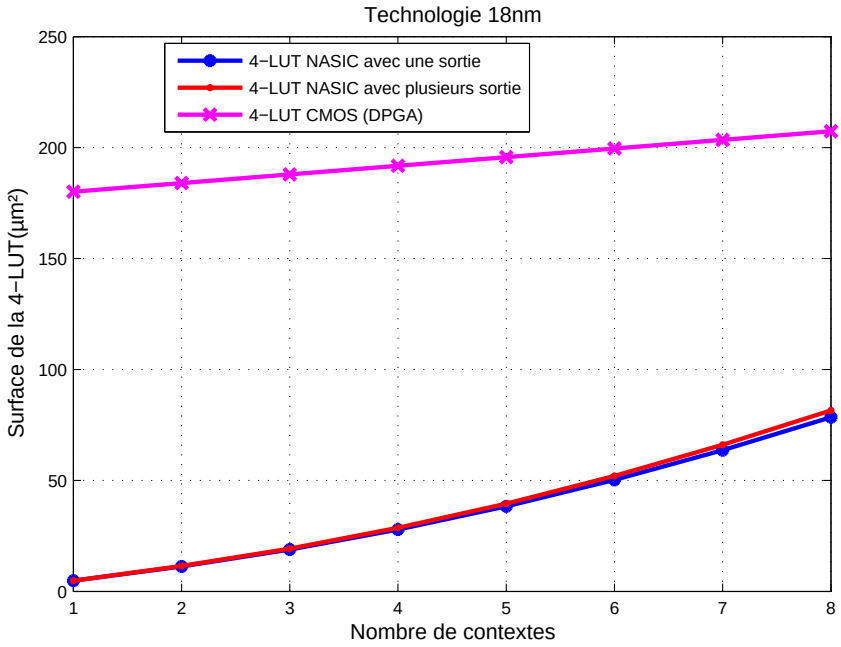


FIG. 4.24 – Technologie 18 nm.

Le modèle a été décrit sous forme de classes et un générateur de PLAs a été mise en œuvre pour placer les dispositifs de base dans une grille produite. Le modèle programmé a été définie pour disposer aussi de ce modéle permettant d'implémenter les applications.

Nous avons étudié la possibilité d'implémenter les supports architecturaux reconfigurables à base de la technologie NASIC. Elle a consisté à construire une LUT avec plusieurs contextes de configuration. Cette étude souligne que les implémentations avec un ou plusieurs sorties ont sensiblement le même coût en surface pour un nombre de contextes inférieur à 8. De deux configurations de multicontexte, la configuration avec plusieurs sorties apporte un gain supplémentaire au routage dans une cellule car il y a un large choix de fonctions au niveau des multiplexeurs de routage. Une autre orientation de la démarche s'est basée sur la comparaison de notre modèle analytique à un modèle classique. Cela nous a permis d'observer la progression de deux modèles selon une évolution technologique. Lorsque le facteur technologique diminue, les deux modèles convergent pour un nombre croissant de contextes de configuration. Le modèle proposé est efficace si le nombre de contextes reste en dessous de la limite de convergence imposée par la technologie.

Conclusion générale

La fin attendue de la loi de Moore pour les technologies CMOS motive les recherches menées actuellement dans le champs nanotechnologique. De nombreux dispositifs physiques ont été présentés qu'il s'agit désormais de savoir exploiter au sein d'une chaîne de CAO pour produire des circuits. En particulier, l'agencement des dispositifs pour former des supports d'exécution n'est pas régi par les mêmes règles que dans le cadre CMOS (interférence électrique, ...).

Résumé de la contribution

L'objectif de ces travaux était de présenter une nouvelle réalité de recherche qui s'articule autour des nanotechnologies. Une étude bibliographique a permis de mieux cerner le contexte du sujet en s'appuyant sur des travaux tiers. Les principales technologies émergentes ont été présentées, au travers de leurs constituants de base qui exhibent le comportement de dispositifs utilisés en électronique classique(transistor diode, ...), et des techniques d'assemblage (ascendante ou Bottom-up, autoassemblage, etc).

La piste architecturale dominante repose sur une structure matricielle construite à l'échelle nanoscopique. En particulier, nous nous sommes intéressé aux nanofabriques telles qu'introduites par l'équipe de Moritz à UMass. Une nanofabrique est constituée de tuiles, chaque tuile étant elle même une matrice régulière de nanotubes de carbone, entouré de micro fils servant à la programmation et l'alimentation en contrôle .

La programmation de ces architectures était réalisée à la main par l'équipe UMass, avec à la clé un manque d'évolutivité et un risque d'erreur. L'adaptation du méta atelier de CAO Madeo au cadre de nanotechnologies a permis d'offrir un jeu d'outils de base pour la mise en œuvre de traitements sur ces structures. En particulier, Madeo a été étendu pour accepter de nouveaux mécanismes de structuration de support architectural, et de nouveaux éléments de base dans la description du support. Dans un second temps, les algorithmes d'implémentation de traitement ont été modifiés pour permettre la prise en considération de logique deux niveaux (ou réseau de descriptions de logique à deux niveaux) et la synthèse de ces descriptions sur les nanomatrices. Un exemple de référence a permis de valider cette approche, tout en démontrant l'apport de Madeo en terme de prospection architecturale. De plus la sémantique non typée des descriptions de haut niveau des traitements a permis de faire varier rapidement l'arithmétique des traitements (introduction d'opérateurs en corps de Galois).

Cette étape s'est traduite en terme d'outils par l'incorporation de règles de dessin (sur la nano matrice) relativement basiques mais extensibles. Ces règles ont permis l'automatisation de la production d'autres architectures de traitement sur la base de nanotuiles. En particulier, une description d'architecture reconfigurable a été proposée, qui tire partie d'une connaissance du support d'implémentation et de ses contraintes (telles que le coût élevé des chemins de contrôle ré-entrants). Cette architecture est une mer de portes, munies de connexions de voisinage directionnelles. Cette simplicité, facilitant la démonstration des mécanismes mis en œuvre, s'accompagne de capacité de prospection (largeur des chemins de données, grain de l'opération logique).

Un modèle paramétrique a également été développé pour caractériser le coût en surface de l'architecture. Ce modèle permet de calculer rapidement l'impact de l'architecture reconfigurable sur son implémentation selon les critères suivants : largeur des canaux de communication, grain logique (nombre d'entrées, nombre de sorties), population des segments, nature du réseau d'interconnexion. La prise en considération d'architectures multicontexte permet d'appréhender le coût des signaux de contrôle (microfils) par rapport à la taille des tuiles.

Le routage dans l'architecture définie peut se baser sur celui des motifs "corner-turn" illustré dans [93]. Cette technique permet la création des matrices de routage physiquement dépeuplées en capacité pour équilibrer le routage et les aspects directionnels. Cette architecture de routage offre des avantages par rapport aux architectures de FPGAs traditionnels : les aiguileurs sont efficacement pipelinés et le routage est rapide. La topologie d'interconnexion présente aussi une tolérance aux fautes puisque les canaux de routage verticaux et horizontaux disposent de plusieurs fils. Un fil coupé (ou avec un connexion décollée) n'est pas utilisé. Il est donc remplacé par un autre fil en bon état.

Perspectives

Les travaux présentés dans ce mémoire démontrent la faisabilité d'outils génériques pour la synthèse sur les NASICs. Un bénéfice direct est d'offrir un retour d'information aux technologues. Ce retour permet de qualifier sinon de quantifier l'utilisabilité des dispositifs technologiques pour implémenter des circuits. Un premier exemple est la mise en œuvre partielle d'une architecture reconfigurable grâce aux outils proposés.

Les travaux présentés n'ont pas vocation à clore un champ de recherche mais constituent tout au contraire un premier jalon vers une solution complète. A ce titre, les perspectives les plus immédiates concernent la finalisation du démonstrateur architecural, en particulier via le prise en considération d'un mécanisme de gestion de l'alimentation en données. A ces perspectives s'ajoutent des interrogations quant à l'intégration d'architectures nanométriques reconfigurables au sein d'un SOC, à la prise en compte d'un taux de défauts souvent élevés.

Architecture de traitement

Au cours de cette thèse, nous avons proposé un modèle d'architecture de traitement. Cependant l'alimentation en données constitue un probleme ouvert. Plusieurs modes de transfert de données s'avèrent possibles, suivant que l'on souhaite permettre un accès aléatoire ou non aux ressources. La préférence d'un mode par rapport à un autre peut avoir l'impact sur les performances de l'architecture (temps de configuration, coût en surface). Une étude sur cet aspect permettrait de mesurer son impact sur des supports architecturaux à technologie indépendante.

Par ailleurs, le spectre des architectures pourrait être étendu, par exemple, par la prise en considération d'opérateurs cablés ou de chemins de données reconfigurables. De tels travaux ont déjà été menés [7] en utilisant les mêmes outils.

Prise en compte de défauts

Un second axe que nous envisageons de prospecter concerne l'impact de mise en œuvre d'une politique de prise en compte de défauts qui est un problème connexe. Une des difficultés à surmonter lors de l'implémentation de traitement en nanotechnologie est le taux élevé de défauts. Nous n'avons pas abordé cette problématique dans nos travaux, mais des recherches connexes [92, 71] montrent plusieurs voies pour tenter de remédier à ce problème, qui pourraient être incorporées dans les outils. Une première approche consiste à introduire une redondance couplée à un mécanisme d'élection. Cette solution peut être mise en œuvre au niveau des outils de bas niveau. Une seconde solution consiste à distribuer la redondance de sorte que l'architecture s'auto corrige. Cette solution peut également être mise en œuvre dans les outils de bas niveau. Une troisième solution est de raisonner en terme de bloc de donnees, associés à des correcteurs d'erreur. Des travaux connexes sont en cours au laboratoire dont nous pourrions bénéficier.

Enfin à supposer que l'architecture physique supporte la reconfiguration, la détection de défaut permettrait de reconditionner les traitements pour ignorer les portions défectueuses ou n'en exploiter que des sous fonctionnalités saines. Cette perspective présentée dans [19], cependant, est actuellement en avance de phase, et demanderait une compétence poussée en terme de test.

Maîtrise technologique

La prise en considération des effets technologiques est actuellement masquée par l'introduction du support d'implémentation que constitue les nanogrilles des tuiles. Cependant, maîtriser les effets (par exemple inductifs) d'un transistor sur l'autre dans la grille, permettrait d'incorporer dans les outils de placement des règles minimisant certains défauts dynamiques, et permettant d'augmenter la fréquence des circuits. De façon évidente cette étape, à venir, requiert des compétences plurielles qui nécessitent la composition d'une équipe transversale (électronique, informatique, chimie ?) ou des collaborations suivies.

Toutes ces perspectives vont permettre un travail en amont des recherches technologiques pour fournir un retour d'information et ainsi potentiellement contribuer à l'orientation de ces recherches en identifiant des gains attendus d'évolutions technologiques.

Annexe A

Annexes

A.1 Code de la description des architectures

```
NanoTile  "
Method: NanoTile
Defines classes :
- NanoArrayTest
- NanoCellTest
- NanoGridTest
"


(
(
(ARRAY
(DOMAIN  1 1 4 1) "END of DOMAIN"
(
(
(COMPOSITE
(
(
(
(
( (NANOGRID 20 20) "END of NANOGRID"
REPRESENTATION
(COLOR red ) "END of COLOR"
 (RECTANGLE 12 12             68 68   ) "END OF RECTANGLE"
(TEXT 60 40 'function name') "END of TEXT"
) "END of REPRESENTATION"
 NAMED f   ) " end of  NAMED  "
```

```
PRODUCE NanoGridTest   ) " END of  PRODUCE  "
CATEGORY LPPGA ) "END of CATEGORY"
(
(
( WIRE ( WIDTH (VALUE channelW '#(3)')) EXPANDED )
REPRESENTATION
(COLOR gray ) "END of COLOR"
 (CHANNEL 1 0 1 80 2 0 ) "END OF CHANNEL"

) "END of REPRESENTATION"
 NAMED ctrlWest   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH (VALUE channelW '#(3)')) EXPANDED )
REPRESENTATION
(COLOR gray ) "END of COLOR"
 (CHANNEL 75 0 75 80 2 0 ) "END OF CHANNEL"

) "END of REPRESENTATION"
 NAMED ctrlEast   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH (VALUE channelW)) EXPANDED )
REPRESENTATION
(COLOR gray ) "END of COLOR"
 (CHANNEL 0 2 80 2 0 2 ) "END OF CHANNEL"

) "END of REPRESENTATION"
 NAMED ctrlNorth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH (VALUE channelW)) EXPANDED )
REPRESENTATION
(COLOR gray ) "END of COLOR"
 (CHANNEL 0 74 80 74 0 2 ) "END OF CHANNEL"

) "END of REPRESENTATION"
 NAMED ctrlSouth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR yellow  ) "END of DEFAULTCOLOR"
```

```
(COLOR red ) "END of COLOR"
 (LINE 0 72 80 72 ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullDownSouth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR yellow ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 0 8 80 8 ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullDownNorth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR red ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 0 70 80 70 ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullUpSouth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR red ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 0 10 80 10 ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullUpNorth   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR yellow  ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE   72 0  72 80  ) "END OF LINEX"
```

```
) "END of REPRESENTATION"
 NAMED pullDownEast   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR yellow ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 8 0 8 80  ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullDownWest   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR red ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 70 0 70 80 ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullUpEast   ) " end of  NAMED  "
(
(
( WIRE ( WIDTH 1) )
REPRESENTATION
(DEFAULTCOLOR red ) "END of DEFAULTCOLOR"
(COLOR red ) "END of COLOR"
 (LINE 10 0 10 80  ) "END OF LINEX"

) "END of REPRESENTATION"
 NAMED pullUpWest   ) " end of  NAMED  "

 ) "END of ELEMENTS"
 ) "END OF COMPOSITE"


        (CONNECTION
   'self f connectTo: self ctrlEast   using: NanoRaConnect new direction: #right'
   'self f connectTo: self ctrlNorth  using: NanoRaConnect new direction: #up'
   'self f connectTo: self ctrlWest   using: NanoRaConnect new direction: #left'
   'self f connectTo: self ctrlSouth  using: NanoRaConnect new direction: #down'
 ) "END of CONNECTION"
```

```
(CONNECTION
  'self f connectTo: self pullUpEast   using: NanoRaConnect new direction: #right'
  'self f connectTo: self pullUpNorth  using: NanoRaConnect new direction: #up'
  'self f connectTo: self pullUpWest   using: NanoRaConnect new direction: #left'
  'self f connectTo: self pullUpSouth  using: NanoRaConnect new direction: #down'
) "END of CONNECTION"

(CONNECTION
  'self f connectTo: self pullDownEast   using: NanoRaConnect new direction: #right
  'self f connectTo: self pullDownNorth  using: NanoRaConnect new direction: #up'
  'self f connectTo: self pullDownWest   using: NanoRaConnect new direction: #left'
  'self f connectTo: self pullDownSouth  using: NanoRaConnect new direction: #down'
) "END of CONNECTION"

(CONNECTION
  'self f connectTo: (self relativeAt: 1@0) f   using:
                            NanoCrossOverConnect new direction: #right '
          'self f connectTo: (self relativeAt: 0@1) f   using:
            NanoCrossOverConnect new direction: #up '
) "END of CONNECTION"
      ) "END OF COMPOSITE"

PRODUCE NanoCellTest   ) " END of  PRODUCE  "
CATEGORY LPPGA ) "END of CATEGORY"
)  "END of ARRAY"
PRODUCE NanoArrayTest   ) " END of  PRODUCE  "
CATEGORY LPPGA ) "END of CATEGORY"
```

A.2 Code du domaine de routage de sortie

```
 routingInternalCellConfig: aConfig north: aValueN east: aValueE south: aValueS
  west: aValueW function: aValueF


"Routing multiplexers block implementation.
 A configuration is coded with 8 bits, 2 bits for each mux."


        | adrMuxX1 outX1 adrMuxX4 outX4 adrMuxX3 outX3 adrMuxX2 outX2 |
```

```
adrMuxX4 := (aConfig quo: (2 raisedTo: 6)) \\ (2 raisedTo: 2).
outX4 := self
                        mux41: adrMuxX4
                        i0: aValueE
                        i1: aValueF
                        i2: aValueS
                        i3: aValueN.
adrMuxX3 := (aConfig quo: (2 raisedTo: 4)) \\ (2 raisedTo: 2).
outX3 := self
                        mux41: adrMuxX3
                        i0: aValueF
                        i1: aValueW
                        i2: aValueS
                        i3: aValueN.
adrMuxX2 := (aConfig quo: (2 raisedTo: 2)) \\ (2 raisedTo: 2).
outX2 := self
                        mux41: adrMuxX2
                        i0: aValueE
                        i1: aValueF
                        i2: aValueW
                        i3: aValueN.
adrMuxX1 := aConfig \\ (2 raisedTo: 2).
outX1 := self
                        mux41: adrMuxX1
                        i0: aValueE
                        i1: aValueF
                        i2: aValueW
                        i3: aValueS.


"outputs connected to the neighbourhood"


^outX1 * (2 raisedTo: 3) +
    (outX2 * (2 raisedTo: 2)) +
    (outX3 * (2 raisedTo: 1)) +
    outX4
```

Bibliographie

[1] Xc6200 field programmable gate arrays. Technical Report 1.0, Xilinx, juin 1996.

[2] International technology roadmap for semiconductors. Technical report, Semiconductor Industry Association, 2006.

[3] T. Akutagawa, Takanori Ohta, T. Hasegawa, T. Nakamura, Christian A. Christensen, and Jan Becher. Formation of oriented molecular nanowires on mica surfaces. *PNAS*, 99(8), 16 Avril 2002.

[4] G. H. Berstein. Quantum-dot cellular automata : Computing by field polarisation. *Design Automation Conference*, page 268, Décembre 2003. USA.

[5] V. Betz and J. Rose. Vpr : A new packing, placement and routing tool for fpga research. In *International Workshop on Field Programmable Logic and Applications*, pages 213–222, 1997.

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

[7] Joel Cambonie, Sylvain Guérin, Ronan Keryell, Loïc Lagadec, Bernard Pottier, Olivier Sentieys, Bernt Weber, and Samar Yazdani. Compiler and system techniques for soc distributed reconfigurable accelerators. In *Synthesis, Architectures and Modeling of Systems (SAMOS 4)*, volume 3133. Springer-Verlag, 2004.

[8] H. Chen, M. F. Jacome, and G. de Veciana. A reconfiguration-based defect-tolerant design paradigm for nanotechnologies. *Design & Test of Computer*, 22(4) :316–326, Juillet/Aoôt 2005.

[9] R. H. Chen, A. N. Korotkov, and K. K. Likharev. Single electron transistor logic. *Applied Physics Letters*, 68(14) :1954–1956, 1996.

[10] Y. Chen, D. A. A. Ohlberg, X. Li, D. R. Stewart, R. S. Williams, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, D. L. Olynick, and E. Anderson. Nanoscale molecular-switch devices fabricated by imprint lithography. *Applied Physics Letters*, 82(10) :1610–1612, 10 Mars 2003.

[11] Y. Chen, G. Young, Douglas A. A. Ohlberg, X. Li, D. R. Stewart, J. O.Jeppeson, K. A. Nielson, J. F. Stoddart, and R. S. Williams. Nanoscale molecular-switch crossbar circuits. In *Nanotechnology*, pages 462–468. Institute of Physcis Publishing, 2003.

[12] M. F. Chisholm, Y. Wang, A. R. Lupini, A. A. Puretzky, B. Brinson, A. V.Melecheho, D. B. Geohegan, H. Cui, M. P. Johnson, S. J. Pennycook, D. H. Lowndes, S. Arepalli,

C. Kittrell, S. Sivaram, M. Kim, G. lavin, J. Kono, R. Hauge, and R. E. Smalley. Comment on single crystals of single-walled carbon nanotubes formed by self-assembling. *Science*, 300, 2003.

[13] P.-W. Chiu. *Towards Carbon Nanotube-based Molecular Electronics*. PhD thesis, Walker Schottky Institut, Allemagne, 2003.

[14] D. H. Chow, H. L. Dunlap, W. W. Williamson III, S. Enquist, B. Gilbert, S. Subramaniam, and P.-M. Lei et G. H. Berstein. Inas/alsb resonant interband tunneling diodes and au-on-inas/alsb-superlattice schottky diodes for logic circuits. *IEEE Electron Device Letters*, 17(2) :69–71, Février 1996.

[15] M. Chu, K. Sulimma, N. Weaver, A. DeHon, and J. Wawrzynek. Object oriented circuit-generators in Java. In Kenneth L. Pocek and Jeffrey Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 158–166, Los Alamitos, CA, 1998. IEEE Computer Society Press.

[16] Y. Cui, X. Duan, J. Hu, and C. M. Lieber. Doping and electrical transport in silicon nanowires. *Physical chemistry*, 104(22) :5213–5216, Juin 2000.

[17] Y. Cui and C. Lieber. Functional nanoscale electronic devices assembled using silicon nanowire building blocks. *Science*, 291 :851, 2001.

[18] Y. Cui, C. Lieber, L. Lauthon, M. Gudiksen, and J. Wang. Diameter-controlled synthesis of single crystal nanowires. *Applied Physics Letters*, 78(14) :2214–2216, 2001.

[19] R. Culbertson, R. Amerson, R. Carter, P. Kuekes, and G. Snider. Defect tolerance on the teramac custom computer. In *1997 IEEE Symposium FPGAs Custom Computer Machines*, pages 116–123. IEEE, 1997.

[20] A. DeHon. *Reconfigurable Architectures for General-Purpose Computing*. PhD thesis, Massachusetts Institute of Technology (MIT), Septembre 1996.

[21] A. DeHon. Arrays-based architecture for molecular electronics. In *First Workshop on Non-Silicon Computation*, 2001.

[22] A. DeHon. Nanowire-based programmable architectures. *ACM on Emerging Technologies in Computing Systems*, 1(2) :109–162, Juillet 2005.

[23] C. Dekker. Carbon nanotubes as molecular quantum wires. *Physics today*, 52(5) :22–28, Mai 1999.

[24] C. Dezan, E. Fabiani, C. Gouyen, L. Lagadec, B. Pottier C. Andriamisaina, and A. Poungou. Synthèse portable pour micro-architectures à grain fin : Application aux turbo décodeurs et nanofabriques. In Hermès-Lavoisier, editor, *Architecture des ordinateurs*, volume 25, pages 893–920, Octobre 2006.

[25] C. Dezan, E. Fabiani, L. Lagadec, B. Pottier, A. Poungou, and S. Yazdani. Abstract execution mechanisms in a synthesis framework . In N. Carter and S. Goldstein, editors, *Workshop Non-Silicon Computations (NSC3), (conjoint avec ISCA 2004, ACM et IEEE)*, Munich (Allemagne), Juin 2004.

[26] D. C. Dixon, C. P. Heij, P. Hadley, and J. E. Mooij. Single-electron tunneling logic devices. *Applied Physics and Dimes*, 1997.

[27] D. Erts, J.D. Holmes, D. Lyons, M.A. Morris, H. Olin, E. Olsson, B. Polyakov, L. Ryen, and K. Svensson. Silicon nanowires studied by tem-stm. *Processing of 7th international conference on Nanometer-Scale Science and Technology and 21st European Conference on Surface Sciences*, juin 2002.

[28] T. Miyazaki et al. Cad-oriented fpga and dedicated cad system for telecommunications. In *Field Programmable Logic and Applications*, volume 1304 of *LNCS*, 1997.

[29] H. Fan, Y.-L. Wu, and C. L. Zhou. Augmented disjoint switch boxes for fpgas. In *Proceedings of the 4th international symposium on Information and communication technologies*, volume 92, pages 129–134. ACM International Conference Proceeding Series, Trinity College Dublin, 03-06 Janvier 2005.

[30] R. P. Feynman. There's plenty of room athe bottom. *Caltech's Engineering and science Magazine*, 23(5), Fevrier 1960.

[31] M. R. B. Forshaw. Algorithms and architectures for use with nanoelectronic computer. Technical report, University College London, Janvier 1999.

[32] S. E. Fost, A. F. Rodrigues, A. W. Janiszewski, R. T. Rausch, and P. M. Kogge. Memory in motion : A study of storage structures in qca. In 1^{th} *workshop on Non-Silicon computation, held in conjunction with* 8^{th} *Int. Symp. on High Performance Computer Architecture*, Boston,MS, 3 Fevrier 2002.

[33] T. Fujii and al. A dynamically reconfigurable logic engine with a multi-context multi-mode unified cell architecture. *Processings of the IEEE International Solid State Circuits conference*, pages 15–17, février 1999. Voir : http ://www.nec.co.jp/press/en/9902/1502.html.

[34] E. Gautrin and L. Perraudeau. *MADMACS : an environment for the layout of regular arrays*. INRIA, Campus de Beaulieu 3542 Rennes Cedex France, 1993 Avril.

[35] A. Gayasen, N. Vijaykrishnan, and M. J. Irwin. Exploring technology alternatives for nano-scale fpga interconnects. In *Design Automation Conference*, pages 921–926, 2005.

[36] S. C. Goldstein and M. Budiu. Nanofabrics : Spatial computing using molecular electronics. In *The Processings of the* 28^{th} *International Symposium on Computer Architecture*, Sweden, 2001. International Conference on Computer Architecture, ACM Press New York, NY, USA.

[37] S. C. Goldstein and D. Rosewater. What makes a good molecular-scale computer device ? Technical Report CMUC-CS-02-181, School of Computer Science, Computer Science Documentation School of Computer Science Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213-3890, 26 septembre 2002.

[38] H. Goronkin, P. V. Allmen, R. K. Tsui, and T. X. Zhu. Nanostructure sciences and technlogies : R&d status and trends in nanoparticles, nanostructured, materials and nanodevices. Technical Report Ch.5, WTEC, Septembre 1999.

[39] K. Goser and C. Pacha. System and circuit aspect of nanoelectronics. *European Solid-State Circuits Conference*, Septembre 1998. The Hague.

[40] J. Greene, E. Hamdy, and S. Beal. Antifuse field programmable gate arrays. In *Processing of the IEEE*, juillet 1993.

[41] A. Guccione, D. Levi, and P. Sundararajan. Jbits : A java-based interface for reconfigurable computing. In 2^{nd} *Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD)*, 1998.

[42] M. S. Gudiksen, L. J. Lauhon, J. W., D. C. Smith, and C. M. Lieber. Growth of nanowire superlattice structures for nanoscale photonics and electronics. *Nature*, 415(6872) :617–620, 7 Fevrier 2002.

[43] L. Guo, E. Leobandung, and S. Y. Chou. A room-temperature silicon single-electron metal-oxide-semiconductor memory with nanoscale floating-gate and ultranarrow channel. *Applied Physics Letters*, 70(7), 17 Mai 1997.

[44] J. Han and P. Jonker. A defect- and fault-tolerant architecture for nanocomputers. *Nanotechnology*, 14 :224–230, Janvier 2003.

[45] H. Hseih and al. Third- generation architecture boosts speed and density of field programmable gate arrays. In *Proceedings of the IEEE Custom Integrated Circuit Conference*, pages 3121–3127, mai 1990.

[46] Y. Huang, X. Duan, Y. Cui, L. Lauthon, K-H. Kim, and C. Lieber. Logic gates and computation from assembled nanowires building blocks. *Science*, 294 :1313–1327, 9 Novembre 2001.

[47] J. O. Jeppesen, C. Patrick Collier, J. R. Heath, Y. Luo, K. A. Nielsen, J. Perkins, J. Stoddart, and E. Wong. Artificial molecular devices based on tetrathiafulvalene. *Journal de physique IV*, 114 :511–513, 2004.

[48] T. Kamins. Advanced device concepts and research. *NSF Nanostructures Workshop*, 29-30 Janvier 2001. Arlington, Virginia.

[49] T. I. Kamins, R. S. Williams, Y. Chen, Y. L. Chang, and Y. A. chang. Chemical vapor deposition of si nanowires nucleated $tisi - 2$ islands on si. *Applied Physics Letters*, 2000.

[50] S. K. Kim and S. Inamdar. Performance evaluation study of 3d synchronous and asynchronous fpga circuits. Technical report, Cornell University,Ithaca, NY, 27 Decembre 2004.

[51] H. Krupnova, A. Abbara, and G. Saucier. A hierarchy-driven fpga partitioning method. In *DAC*, 1997.

[52] T. Kueckes, K. Kim, E. Joselevich, G. Tseng, C. Cheung, and C. Lieber. Carbon nanotube based nonvalatile random access memory for molecular computing. *science*, 289 :94–97, 2000.

[53] P. J. Kuekes and R. S. Williams. Demultiplexer for a molecular wire crossbar network (mwcn demux). Technical Report 6256767, US Patent (Hewlett-Packard, juillet 2001.

[54] C. Kyungchee. Ethical issues of nanotechnology development in the asia-pacific region. Technical Report 1.0, Ethics of Science and Technology, 5-7 Novembre 2003.

[55] L. Lagadec. *Abstraction, modélisation et outils de CAO pour les architectures reconfigurables*. PhD thesis, Université de Rennes I, 35065 Rennes cedex, France, 2000.

[56] L. Lagadec, D. Lavenier, E. Fabiani, and B. Pottier. Placing, routing and editing virtuals fpgas. *Computer Science*, 2147 :357–366, 2001.

[57] L. Lagadec and B. Pottier. Object-oriented meta tools for reconfigurable architectures. In Schewel J., editor, *Photonics East*, Boston, Ms, novembre 2000. SPIE.

[58] L Lagadec, B. Pottier, and O. Villellas-Guillen. An lut-based high level synthesis framework for reconfigurable architectures. In S.S. Batttacharyya, E. Deprettere, and J. Teich, editors, *Domain-Specific Processors : Systems, Architectures, Modeling, and Simulation*, pages 19–39. Marcel Dekker, novembre 2003.

[59] M. Lahmani, C. Dupas, and P. Houdy. Les nanosciences nanotechnologies et nanophysique. In Belin, editor, *Collection Echelles*. NSTI, Octobre 2004.

[60] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, 10 :346–365, 1961.

[61] G. Lientschnig, I. Weymann, and P. Hadley. Simulating hybrid circuits of single-electron transistors and field-effect transistors. *Jpn. J. Applied Physics*, 42(Part 1, 10) :6467–6472, Octobre 2003.

[62] Y. Luo, J. O. Jeppesen, C. Patrick Collier, K. A. Nielsen, E. Delonno, G. Ho, J. Perkins, H. R. Tseng, T. Yamamoto, J. Stoddart, and J. R. Heath. Two dimensional molecular electronics circuits. *CHEMPHYSCHEM*, 3 :5519–525, 2002.

[63] R. Martínez, I. Ratera, A. Tárraga, P. Molina, and J. Veciana. A simple and robust reversible redox-fluorescence molecular switch based on a 1,4-disubstituted azine with ferrocene and pyrene units. *ChemComm*, pages 3809–3811, 2006.

[64] C. Mead and L. Conway. *Introduction to VLSI systems*. Addison-Wesley publishing company, 1980.

[65] Mintmire and al. Are fullerene tubes metallic ? *Physical review letters*, 68 :631–634, 1992.

[66] M. Mishra and S. C. Goldstein. Scalable defect tolerance for molecular electronics. In 1^{th} *workshop on Non-Silicon computation*, Cambridge,MA, Février 2002.

[67] C. A. Moritz and T. Wang. Latching on the wire and pipelining in nanoscale designs. In *third Workshop on A non-Silicon Computation*, pages 39–45, Allemagne, juin 2004. ISCA.

[68] C. A. Moritz, T. Wang, M. Ben-Naser, and Y. Guo. Wire-streaming processor on 2-d naniwires fabrics. In *Nanotech*. NSTI, 2005.

[69] L. M. Murchie and C. Ebeling. Pathfinder : A negotiation-based performances-driven router for fpgas. *In FPGAs*, pages 111–117, Février 1995.

[70] C. Murray. Self-assembling nanocrystal superlattices : building with artificial atoms. IBM Corporation, T. J. research Center, septembre 2003.

[71] G. Norman, D. Parker, M. Kwiatkowska, and S. K. Shukla. Evaluating the reliability of defect-tolerant architectures for nanotechnology with probabilistic model checking. In 17^{th} *International Conference on VLSI Design*, pages 907–914. IEEE Computer Society, Janvier 2004.

[72] W. II Park, J. S. Kim, G.-C. Yi, M. H. Bae, and H.-J. Lee. Fabrication and electrical characteristics of high-performance zno nanorod field-effect transistors. *Appliied Physics Letters*, 85(21) :5052–5054, Novembre 2004.

[73] J. I. Pascual, N. Lorente, Z. Song, H. Conrad, and H. P. Rust. Selectivity in vibrationally mediated single molecule chemistry. *nature*, 423(6939) :525–528, Mai 2003.

[74] D. Picard, L. Lagadec, and A. Poungou. Vers un fpga paramétrique en nanotechnologie. In *MAJESTIC'2006*, Novembre 2006.

[75] A. Poungou, E. Fabiani, L. Lagadec, and B. Pottier. Du reconfigurable aux nanofabriques : composants nano-électroniques et outils de modélisation. In *Sympa'2005*, pages 47–56. GDR ARP, Avril 2005.

[76] A. Poungou and L. Lagadec. Conception d'un fpga en nanotechnologie. In *Sympa'2006*, pages 198–210. GDR ARP, Octobre 2006.

[77] A. Mei R. Sidhu, S. Wadhwa and V. K. Prasanna. A self-reconfigurable gate array architecture. In *Field-Programmable Logic and Applications. The Roadmap to Reconfigurable Computing : 10th International Conference (FPL 2000)*, volume 1896, pages 106–120. Springer Berlin / Heidelberg, 2000.

[78] R. M.P. Rad and M. Tehranipoor. A new hybrid fpga with nanoscale clusters and cmos routing. In *Design Automation Conference*, pages 727–730, 2006.

[79] S. Ranganathan, I. Steidel, F. Anariba, and R. L. McCreery. Covalently bonded organic monolayers in a carbon substrate : A new paradigm for molecular electronics. *NANO Letters*, 1(9) :491–494, 2001.

[80] J. rose and D. Hill. Architectural and physical design challenges for one-million gate fpgas and beyond. *In FPGAs*, pages 129–132, 1997.

[81] L. Rotkina, J.-F Lin, and J. P. Bird. Nonlinear current-voltage characteristics of pt nanowires and nanowire transistors fabricated by electron-beam deposition. *Applied Physics Letters*, 83(21), novembre 2003.

[82] Saito and al. Electronic structure of graphene tubules based-on c_{60}. *Physical review*, B(46) :1804–1811, 1992.

[83] J. E. Savage, E. Rachlin, A. DeHon, C. M. Lieber, and Y. Wu. Radial addressing of nanowires. *Emerging Technologies in Computing Systems*, 2(2) :22–28, Avril 2006.

[84] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli. Sis : A system for sequential circuit synthesis. Technical report, Department of Electrical Engineering and Computer Science, Berkeley, Mai 1992.

[85] S. Sivaswamy, G. Wang, C. Ababei, K. Bazargan, R. Kastner, and E. Bozorgzadeh. Harp : Hardwired routing pattern fpgas. *International Symposium on Field Programmable Gate Arrays*, 20-22 Fevrier 2005.

[86] The Royal Society and The Royal Academy of Engineering. Nanoscience and nanotechnologies : Opportunities and uncertainties. Technical report, Nanoscience and Nanotechnologies, nano@royalsoc.ac.uk, 29 Juillet 2004.

[87] A. Takahara and al. More wires and fewer luts : A design methodology for fpgas. In *FPGA*, 1998.

[88] E. Tau, I. Eslick, D. Chen, J. Brown, and A. DeHon. A first generation dpga implementation. In *Proceedings of the Third Canadian Workshop on Field-Programmable Devices*, 1995.

[89] S. Tiwari, F. Rana, H. Hanafi, A. Hartstein, E. F. Crabbé, and K. Chan. A silicon nanocrystals based memory. *Applied Physics Letters*, 68(10) :1377–1379, 1996.

[90] P. D. Tougaw and C. S. Lent. Logical devices implemented using quantum cellular automata. *Applied Physics*, 28 Juillet 1993.

[91] J. D. Ullman. *Computational aspects of VLSI*. Cumputer Science Press, 1984.

[92] T. Wang, M. ben Naser, Y. Guo, and C. A. Moritz. Combining circuit level and system level techniques for defect-tolerant nanoscale architectures. *2nd IEEE International Workshop on Defect and Fault Tolerant Nanoscale Architectures*, Juin 2006. Boston.

[93] N. Weaver, J. Hauser, and J. Wawrzynek. The sfra : a corner-turn fpga architecture. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 3–12. ACM Press, Fevrier 2004.

[94] Dongmok Whang, Song Jin, and Charles M. Lieber. Nanolithography using hierarchically assembled nanowire masks. *Nano Letters*, 3(7) :951–954, 19 Juin 2003.

[95] C. T. White, D. H. Robertson, and J. W. Mintmire. Helical and rotational symmetry of nanoscale graphitic tubules. *Physic review*, B(47) :5485–5488, 1993.

[96] S. Wilton. Architectures and algorithms for field-programmable gate arrays with embedded memories. Thèse, Université de Toronto, 1997.

[97] V. V. Zhirnov and D. J. C. Herr. New froniters : Self-assembly and nanoelectronic. *IEEE Computer*, 34(1) :34–43, Janvier 2001.

[98] M. M. Ziegler and Mircea R. Stan. The cmos/nano interface from a circuits perspective. In *The International Symposium on Circuits and Systems (ISCAS)*, pages 904–907. IEEE, mai 2003.

Liste des publications

Conférence internationale avec comité de lecture

[25] Catherine Dezan, Erwan Fabiani, Loic Lagadec, Bernard Pottier, **Alix Poungou**, et Samar Yazdani. Abstract execution mechanisms in a synthesis framework.IN N. Carter et S. Goldstein, éditeurs, *Workshop Non-Silicon Computations (NSC3), (conjoint avec ISCA 2004, ACM et IEEE*, Munich (Allemagne), Juin 2004.

Conférence nationale avec comité de lecture

[75] **Alix Poungou**, Erwan Fabiani, Loic Lagadec, et Bernard Pottier. Du reconfigurable aux nano-fabriques : composants nano-électroniques et outils de modélisation. *Sympa'2005*, Pages 47-56. GDR ARP, Avril 2005.

[76] **Alix Poungou**, et Loic Lagadec. Conception d'un FPGA en nanotechnologie. *Sympa'2006*, Pages 198-210. GDR ARP, Avril 2006.

[74] Damien Picard, Loic Lagadec, **Alix Poungou**. Vers un FPGA paramétrique en nanotechnologie. *MAJESTIC'2006*, Novembre 2006.

Revue des sciences et technologies avec comité de lecture

[24] Catherine Dezan, Erwan Fabiani, Loic Lagadec, Bernard Pottier, Caaliph Andriamisaina, et **Alix Poungou**. Synthèse portable pour micro-architectures à grain fin : Application aux turbo décodeurs et nanofabriques. In Hermès-Lavoisier, éditeur, *Architecture des ordinateurs*, Volume 25, Pages 893-920, Octobre 2006.

Nanotechnologies et Architectures Reconfigurables

Résumé L'émergence des nanotechnologies souvent discrètes requière des méthodes particulières pour la conception des circuits intégrés. Une approche de conception dominante des architectures repose sur une structure matricielle à l'échelle nanoscopique. Celle-ci intégre des propriétés de simplicité, de regularité et de tolérance aux fautes pour un support technologique performant.

Cette évolution technologique implique la disponibilité d'outils de prospection pour explorer rapidement les topologies architecturales et évaluer leurs performances. Nous avons incorporé des règles de dessin basées sur la conception NASIC dans un outil existant pour synthétiser les nanofabriques. Une automatisation de la production des architectures de traitement s'appuie sur ces règles. Elle montre une rupture de méthodes par rapport à une conception artisanale : un petit processeur est illustré avec une considération de calcul d'impacts. Par ailleurs, nous avons défini une architecture reconfigurable s'appuyant sur un composant existant. Celle-ci est outillée d'une capacité de prospection pour caractériser son coût spatial.

Mots-clé nanotechnologies, FPGAs, outils de CAO, NASICs, architectures reconfigurables

Nanotechnologies and Reconfigurable Architectures

Abstract The evolution of nanotechnologies requires specific methodology for the design of the integrated circuits. A main stream design approach of the architectures is based on the nanoscale matrix structure of the nanowires. This structure must be simple, regular and must have defect tolerance for making the technological supports competitive.

The technological evolution requires available prospecting tools to explore architectural topologies quickly and to assess their performances. Layout generation regulations based on the NASIC design have been integrated in an existing tool to synthesize the nanofabrics. Automation of such physical structure generation for nanofabrics is based on the aforementioned rules. It shows a methodological divergence as compared to a manual design : a small processor has been presented with a calculation of physical impacts. Moreover, we have also defined a (nano)-reconfigurable architecture based on an existing component. This component is equipped with prospecting capacity to calculate its performance metrics.

Keywords nanotechnologies, FPGAs, CAD tools, NASICs, reconfigurable architectures