

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE**

**UNIVERSITE FERHAT ABBAS-SETIF
UFAS (ALGERIE)**

THESE

Présentée à la Faculté des Sciences de l'Ingénieur

Département d'Electronique

Pour l'obtention du Diplôme de

DOCTORAT D'ETAT

OPTION : Contrôle

Par

Mr HACENE AZIZI

THEME

**Parallélisme et Logique Floue Appliqués
Aux outils de la CAO en Microélectronique**

Soutenue le :..... Devant la commission d'examen

MM.

A. MERZOUKI

Prof. à l'université. F. Abbas de Sétif

Président

F. DJAHLI

Prof. à l'université. F. Abbas de Sétif

Examineur

H. TEBBIKH

Prof. à l'université. 8 Mai 45 de Guelma

Examineur

M. S. AIDA

Prof. à l'université Mentouri de Constantine

Examineur

K. BENMAHAMMED Prof. à l'université. F. Abbas de Sétif

Rapporteur

REMERCIEMENTS

Ce travail a été réalisé au laboratoire d'optoélectronique et composants de l'université Ferhat Abbas de Sétif en collaboration avec le laboratoire de Conception et d'Intégration des Systèmes (LCIS) de Valence (FRANCE)

J'exprime ma vive reconnaissance à monsieur **K. BENMAHAMMAD** professeur à l'université de Sétif d'avoir accepté la lourde tâche d'être directeur de cette thèse. Je le remercie de m'avoir fait bénéficier de ses conseils fructueux et ses suggestions pertinentes qui m'ont permis de mener à bien ce travail sans oublier ses remarques finales qui ont contribué à améliorer ce manuscrit.

Je tiens à témoigner ma profonde gratitude envers Madame **O. AKTOUF**, Maître de Conférence à l'école d'ingénieurs ESISAR de l'institut National polytechnique de Grenoble et membre du laboratoire LCIS qui a donné une impulsion décisive à mon travail et qui m'a guidé attentivement tout le long de mes recherches.

Je remercie vivement Monsieur **A. MERZOUKI** professeur à l'université de Sétif d'avoir bien voulu me faire l'honneur d'accepter la présidence du jury de cette thèse.

Je tiens à ce que Monsieur **F. DJAHLI**, professeur à l'université de Sétif, sache que j'apprécie énormément sa présence dans mon jury.

C'est un réel plaisir que Monsieur **H. TEBBIKH**, professeur à l'université de Guelma et directeur du laboratoire d'automatique et informatique de Guelma (L.A.I.G), participe au jury. J'en suis très honoré et l'en remercie vivement.

J'exprime ma vive reconnaissance à Monsieur **M.S. AIDA** professeur à l'université de Constantine d'avoir accepté de participer à ce jury. J'en suis très honoré et l'en remercie vivement.

Je désire également remercier Mrs **L. ZOUAOUI**, **C. AKTOUF** et **S. MOKHNACHE** pour leurs aides et leurs disponibilités.

SOMMAIR E

Introduction	1
--------------	---

Chapitre 1 Flot de Conception

I.1 Introduction	5
I.2 Processus de Conception	6
I.2.1 Les Niveaux d'Abstraction	6
I.2.2 Conception Logique	8
I.2.3 Conception Physique	11
I.2.3.1 Style de Conception	12
I.2.3.2 Plan de Masse	18
I.2.3.4 Dessin de Masques	19
I.2.3.5 Placement	19
I.2.3.6 Routage	20
I.2.3.7 Compaction	20
I.3 Simulation	20
I.3.1 Simulateurs Electriques	21
I.3.2 Simulateurs Logiques	21
I.3.3 Simulateurs Analogique/Digitales	21
I.3.4 Simulateurs Fonctionnels	22
I.4 Fabrication et Test	23
Références	24

Chapitre II Les Réseaux de KOHONEN Appliqués au Placement de Cellules VLSI

II.1 Introduction	26
II.2 Classification des Algorithmes de Placement	27
II.3 Motivation	30
II.4 Enoncé du Problème	32
II.5 Algorithme	33
II.5.1 Prétraitement	33
II.5.2 Création de la séquence d'entrée	34
II.5.3 Initialisation de la Matrice des Poids Synaptiques	35
II.5.4 Itérations	35
II.5.4.1 Présentation d'un stimulus	35
II.5.4.2 Calcul des Réponses et Recherche d'un Neurone Gagnant	35
II.5.4.3 Adaptation des Poids Synaptiques	36
II.6 Conditions d'arrêt	37
II.7 Résultats et Tests	37
II.7.1 Influence de la Taille du Réseau	40
II.7.2 Influence du Choix de la Position de la Première Cellule	41
II.7.3 Influence du Choix de la Séquence d'Entrée	41
II.8 Analyse et Adaptation de l'Algorithme	41
II.8.1 Limites Géométriques	42
II.8.2. Croissance des Solutions	42
II.9 Amélioration de l'Algorithme par Recuit Simulé	43
II.9.1 Algorithme	44
II.9.2 Prétraitement Global	44
II.10 Autres Améliorations	45
II.10.1 Choix de la Taille du réseau	45
II.10.2 Choix de la Première Cellule et de sa Position	45
II.10.3 Initialisation de la Matrice des poids Synaptiques	45

II.11 Tests et Résultats	45
II.12 Structure des Données	47
II.13 Extrapolation de l'Algorithme	49
II.13.1 Les Connexions Entre Cellules	49
II.13.2 Les Cellules de Différentes Tailles	49
II.13.3 Le But Recherché	49
II.13.4 Modélisation du Problème	50
II.13.4.1 Modélisation Neuronale	50
II.13.4.2 Modélisation des Cellules	51
II.14 Adaptation de l'Algorithme	52
II.14.1 Enoncé du Problème	52
II.14.2 Prétraitement	53
II.14.3 Séquence d'Entrée	53
II.14.4 Calcul des Réponses	53
II.14.5 Recherche du Neurone Gagnant	56
II.14.6 Adaptation des Poids Synaptiques	57
II.14.6.1 Nouvelle Distance	57
II.14.6.2 Adaptation	60
II.15 Résultats et Tests	61
II.16 Conclusion	63
Références	65

Chapitre III Algorithme de Routage

III.1 Introduction	67
III.2 Formulation du Problème	68
III.3 Les approches du Routage	68
III.3.1 Routage Global	69
III.3.1.1 Définition des Zones	69
III.3.1.2 Assignement des Zones	71
III.3.1.3 Assignement des points de Connexion	71
III.3.2 Routage Détaillé	72
III.4 Etat de l'Art	72
III.4.1 Algorithme de LEE	73
III.4.2 Variations sur LEE	76
III.4.2.1 AKERS	76
III.4.2.2 RUBIN	77
III.4.2.3 HOEL	77
III.4.2.4 MIKAMI ET TABUSHI	78
III.4.2.5 BACKTACKING	78
III.4.2.6 SOUKUP	78
III.5 Routage Concurrent	80
III.6 Application	81
III.6.1 Algorithme de Routage	81
III.6.2 Fonction Heuristique et Génération des Nœuds	82
III.6.3 Grille	86
III.6.4 Connexion à Plusieurs Terminaisons	86
III.6.5 Utilisation de Plusieurs Niveaux	89
III.7 Conclusion	92
Références	93

Chapitre VI Application de La Logique Floue Dans la Conception VLSI

VI.1 Introduction	95
VI.2 Théories des Base des Ensembles Flous	97
VI.2.1 Opérations sur les Ensembles Flous	99
VI.2.2 Propriétés des Ensembles Flous	100
VI.2.3 Les Relations Flous	101
VI.2.4 Le Raisonnement Approximatif par Variables Linguistiques	102
VI.3 Application de la Logique Floue aux Outils de la CAO	103
VI.3.1 Application de la Logique Floue aux Algorithmes Constructifs	104
VI.3.2 Application de la Logique Floue aux Algorithmes Itératifs	104
VI.3.3 Algorithmes et Outils de Logique Floue pour la Synthèse Logique	106
VI.3.4 Algorithmes et Outils de Logique Floue pour la Synthèse Architecturale	107
VI.4 Application	107
VI.4.1 Application de La Logique Floue au Problème de Placement	107
VI.4.2 Adaptation de la Matrice de Connexion	108
VI.4.3 La Relation Flous et la Matrice de Connexion	108
VI.4.4 Modification de la Matrice de Connexion	109
VI.4.5 Résultats et Discussion	114
VI.4.6 Conclusion	113
VI.5 Application de la Logique Floue au Problème de Routage	113
VI.6 Conclusion	115
Références	116

Chapitre V Le Parallélisme et la Conception VLSI

V.1 Introduction	118
V.2 Concept Général	119
V.2.1 Programmation Parallèle	119
V.2.2 Parallélisme et Granularité	119
V.2.3 La Loi de AMDHAL	120
V.2.4 La Communication Entre Processus	121
V.2.5 La Synchronisation Entre les Processus	122
V.2.6 Motivation Pour le Parallélisme	122
V.3 Les Trois Phases de l'Exploitation du Parallélisme	123
V.3.1 Exploitation du Parallélisme de Contrôle	123
V.3.2 Exploitation du Parallélisme de Données	123
V.3.3 Exploitation du Parallélisme de Flux	124
V.4 Classification des Architectures Parallèles	124
V.5 Exploitation du Parallélisme dans la Conception VLSI	126
V.6 Application	127
V.6.1 Le Parallélisme Intrinsèque des Réseaux de Neurones	127
V.6.2 Le Parallélisme Pour le Routage	132
V.7 Conclusion	134
Références	135

Conclusion et Perspectives	137
----------------------------	-----

Introduction

La densité d'intégration des circuits VLSI (*Very Large Scale Integration*) atteinte au début des années 90 a permis de développer des circuits composés de plus d'un million de transistors. Une si grande complexité rend pour le moins très difficile la conception de tels systèmes en dessinant chaque transistor ou en définissant chaque signal en terme de portes logiques. De ce fait, et de par la nécessité de réduction de l'effort et du temps de conception des circuits, l'utilisation des outils de conception assistée par ordinateur (CAO) en microélectronique s'est avérée indispensable. L'expression de conception assistée par ordinateur (CAO) recouvrira un ensemble de systèmes matériels et logiciels où l'ordinateur intervient pour accomplir très rapidement et économiquement des tâches de calcul dans le but d'aider le concepteur à prendre des décisions logiques tout en lui laissant la liberté d'introduire son art, son expérience et son intuition.

Aujourd'hui l'ordinateur est devenu un outil indispensable pour la conception et la fabrication de nouveaux produits. La conception de circuits ne peut se faire maintenant sans l'ordinateur. Il intervient dans toutes les étapes de l'élaboration d'un produit.

Ces outils ont suivi l'évolution technologique en gagnant un niveau d'abstraction par décennie. Ainsi, ont été commercialisés successivement des outils de dessin de masques (niveau *layout*), puis des outils de synthèse logique (niveau porte logique) et enfin, à l'heure actuelle, des outils de synthèse travaillant au niveau transfert de registres (RTL en anglais pour *Register Transfert Level*) [1]. Le niveau RTL est un niveau intermédiaire entre le niveau le plus élevé (appelé niveau algorithmique) et le niveau porte logique.

La complexité des circuits implantés n'est plus limitée par la technologie, mais par le processus de conception. Un concepteur ne peut plus réaliser certains circuits sans l'aide d'outils de CAO tant la complexité des circuits est grande, et ceci pour plusieurs raisons : le temps de conception, l'optimisation et la stabilité des circuits. L'automatisation de la réalisation des circuits intégrés est donc aujourd'hui une nécessité.

Ce travail est une contribution à l'automatisation de la conception des circuits intégrés, à partir de la phase physique qui inclut le placement des cellules et le routage et l'application du parallélisme et de la logique floue. Le parallélisme est appliqué pour une optimisation purement temporelle (accélération de la vitesse d'exécution) alors que la logique floue est appliquée du fait de la présence de fonction multi-objective dans les problèmes d'optimisation et en particulier dans la phase de conception physique.

Le mémoire comporte cinq chapitres incluant une introduction générale et une conclusion. Le premier chapitre se concentre sur la description du processus de conception des circuits intégrés depuis le cahier des charges jusqu'au processus de fabrication. En effet, Il est démontré que la réalisation d'un circuit intégré, doit passer d'abord par la spécification de la fonction et les performances que doit remplir ce dernier, ainsi que son environnement.

Le deuxième chapitre étudie le placement relatif des cellules lors du dessin d'un circuit intégré. Dans ce chapitre, une solution au problème de placement des différentes cellules lors de la réalisation d'un circuit intégré est proposée en utilisant le réseau de KOHONEN [2]. En effet, Des études ont montré qu'un type particulier de réseaux de neurones artificiels, le réseau de KOHONEN, peut être utilisé pour positionner des cellules de taille et de forme identiques sur un plan de masse sans rails de connexions.

Un premier algorithme a été présenté avec des hypothèses simplificatrices. L'extrapolation de l'algorithme, introduit la notion de borne où chaque cellule est représentée par un ensemble de bornes, et où les connexions se faisaient entre ces mêmes bornes.

Le troisième chapitre se préoccupe essentiellement du problème de routage, où il s'agit d'implanter la totalité des connexions tout en respectant les règles de dessin et en optimisant certains critères tels que la longueur de connexion, le nombre de points de liaison entre deux niveaux... L'algorithme utilisé dans l'étape de routage utilise la recherche heuristique et pour l'appliquer aux circuits intégrés il a fallu procéder à quelques adaptations.

Le quatrième chapitre prend en considération, explicitement, deux aspects importants de problème de placement : présence de multiples objectifs contradictoires et utilisation de la connaissance experte dans la prise de décision. Ces aspects du problème de placement peuvent être traités par la technique basée sur la théorie des ensembles flous [3] [4] [5]. La matrice de connexion a été modifiée et alors la théorie des ensembles flous a été employée pour réaliser le placement [6] [7]. La présence de plus d'un critère d'optimisation a été exploitée pour résoudre le problème de routage, où la fonction coût ne correspond plus à un seul critère d'optimisation.

Enfin le cinquième chapitre introduit le concept de parallélisme et son application à la conception des circuits intégrés pour la phase de placement et routage [8]. En effet, dans la phase de placement nous avons utilisé l'aspect de parallélisme intrinsèque des réseaux de neurones. Quant à la phase de routage, nous avons utilisé la solution du partitionnement du circuit et qui consiste à considérer le circuit (espace de routage) non comme une entité entière, mais comme un ensemble de parties indépendantes. Chaque partie sera traitée par un processeur, sans interférences entre elles.

Les conclusions et perspectives présentent les résultats obtenus dans ce travail et décrivent les lignes à suivre pour les améliorer.

REFERENCES DE L'INTRODUCTION

- [1] V. P. VIJAYARAGHAVAN "Exploration des liens entre la Synthèse de Haut niveau (HLS) et la Synthèse au niveau des Registres (RTL)" Thèse de Doctorat INP Grenoble 1996
- [2] Kohonen T; Self-Organization and Associative Memory. 1989. 3rd Edition. Springer-Verlag. NY USA.
- [3] L. A. Zadeh, "Fuzzy sets," Info. & Contrc I
- [4] H. J. Zimmerman, "Fuzzy Set Theory and Its Applications", Kluwer Nijhoff, 1991. Va 8, 1965.
- [5] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-3, No.1, 1973, Proceedings of --Fuzz 1992 and 1993 conferences.
- [6] Sadiq M. Sait, Habib Youssef et al., "Iterative heuristics for multiobjective VLSI standard-cell placement", IEEE 2001
- [7] Junaid A. Khan, Sadiq M. Sait, "Fuzzy biasless simulated evolution for multiobjective VLSI placement", IEEE 2002.
- [8] J. A. Chandy "Parallel Algorithms For Standard Cell Placement Using Simulated annealing" PHD thesis, University of Illinois at Urbana-Champaign, 1996.

Conclusion et Perspectives

L'objectif des travaux était d'appliquer le concept du parallélisme et la logique floue aux outils de la CAO. Notre choix s'est porté sur la conception physique et plus précisément le problème du placement des cellules et le tracé des connexions.

En ce qui concerne le problème de placement, il faut dire que le fait d'utiliser les réseaux de neurones pour optimiser le placement de cellules s'avère une idée intéressante et "originale". Comme perspective, et pour mieux évaluer les performances de l'algorithme il faut prévoir, dans des travaux futurs, des tests sur des circuits de références (benchmarks).

Pour le problème de routage, l'utilisation de l'algorithme A*, qui est initialement utilisé dans les problèmes de l'intelligence artificielle, a donné de bons résultats comparativement à l'algorithme de LEE. Notre adaptation de cet algorithme avec, entre autre, l'utilisation de connexions à plusieurs terminaisons et deux niveaux de connexion. Comme perspectives, il faut prévoir aussi des tests sur des circuits de références.

Dans ce travail nous avons proposé développé et implémenté une nouvelle approche à base de logique floue pour le placement de cellules. La logique floue est connue pour sa capacité à traiter l'imprécision et l'incertitude des systèmes dynamiques et à représenter l'expertise humaine. Elle est salubre dans tous les niveaux hiérarchiques du processus de conception. L'utilisation de l'approche floue pour le problème de placement (modification de la matrice de connexion) a donné de bons de résultats ce qui n'est pas le cas avec l'incorporation d'un deuxième critère d'optimisation dans la fonction multi-objective pour le problème de routage. Comme perspectives, il faut étendre le choix des paramètres d'optimisation à la surface du circuit, le temps de propagation...pour placement de cellules.

Le parallélisme reste une bonne alternative pour l'accélération du processus de traitement surtout avec la grande quantité de calcul que nécessitent les algorithmes de placement et de routage. L'utilisation des réseaux de neurones pour le placement de cellules nous a permis d'exploiter le parallélisme intrinsèque de ceux-ci. Comme perspectives, il faut exploiter le recuit simulé pour paralléliser la phase de placement.

Cette vision perspective concerne l'amélioration des travaux déjà réalisés. Alors, si nous nous plaçons dans la vision générale du flot de conception, nous pouvons aussi prévoir d'autres travaux futurs tels que l'utilisation du langage de description de haut niveau (VHDL) pour concevoir des circuits en vue de tests (DFT) ; ce projet est en collaboration avec un laboratoire privé DEFACTO ou encore et avec l'évolution des systèmes embarqués, l'utilisation du codesign (exploitation combinée du matériel avec les programmes) ; ce projet est en collaboration avec le laboratoire LCIS de Valence (France).

Flot de Conception d'un circuit intégré

I.1. INTRODUCTION

Les systèmes et les composants microélectroniques sont de plus en plus intégrés dans différents systèmes dédiés ou au sein d'applications importantes telles que les télécommunications, l'avionique, l'automatique, la robotique, l'appareillage électronique, etc.

Certains de ces systèmes microélectroniques sont très complexes; ils peuvent être perçus comme étant une interconnexion de blocs contenant chacun des millions de transistors. Souvent, lorsque les concepteurs sont en train de concevoir un matériel pour une application spécifique, ils choisissent des composants préconçus, déjà testés et essayés [1] [2].

La réutilisation du design devient de plus en plus populaire puisqu'elle permet de réduire aussi bien le temps de réalisation d'un circuit que son coût. Les tâches de test et de vérification débutent maintenant au niveau comportemental où l'intégration des composants est une importante solution. Le concepteur dispose d'une spécification complète ou partielle du système global ou du composant spécifique qui doit être conçu à partir d'ingrédients microélectroniques [4][5].

Ce chapitre présente les étapes essentielles dans le processus de conception, depuis la spécification du cahier des charges et jusqu'à la fabrication et le test du circuit. Il est divisé en deux grandes parties, suivant le niveau d'abstraction, une partie avec un niveau d'abstraction plus élevé avec la conception logique et le deuxième niveau, un peu plus bas, concerne la conception physique avec les étapes de plan de masse, placement, routage et compaction. Les différents styles de conception des circuits intégrés sont présentés avec des tableaux présentant les avantages et les inconvénients de chaque style.

I.2 PROCESSUS DE CONCEPTION

Le processus de conception est un ensemble d'actions, ordonnées et bien définies, à exécuter par l'outil ou l'opérateur de conception pour satisfaire une spécification d'un système, des consignes ou des objectifs prédéfinis. Ces actions sont étroitement dépendantes de la nature du système en cours de conception. D'où, il n'y a pas une méthode unique pour concevoir des systèmes différents, cependant la philosophie y demeure globalement la même. Le processus de conception accueille à son entrée la spécification d'un produit et génère à sa sortie l'implémentation de celui-ci. Fournie par le client, la spécification englobe toutes ou quelques consignes que l'implémentation du produit doit satisfaire.

Les processus de conception sont classifiés en deux grandes catégories [3]:

- La conception personnalisée ("*custom design*") : elle est coûteuse et requiert des efforts colossaux, alors son utilisation est limitée à des applications spécifiques telles que la conception des unités de processeurs.
- La conception semi-personnalisée ("*semi-custom design*") : elle représente avec la méthodologie de la conception réutilisable une bonne alternative de la conception personnalisée. La conception semi-personnalisée peut être répartie en deux types : le design à base de cellules et le design à base de tableaux. Le premier s'effectue en utilisant des cellules de la librairie (des cellules); quant au second, il utilise des modules MPGAs et FPGAs [8].

I.2.1 LES NIVEAUX D'ABSTRACTION

L'abstraction est un concept basé sur le fait d'ignorer ou de supprimer dans un modèle certains détails du système original afin de simplifier sa taille (du modèle), d'en déduire des conclusions d'ordre général et aussi pour réduire le volume du travail requis. On distingue plusieurs niveaux d'abstraction :

- *Niveau système* : on essaye à ce niveau d'identifier l'architecture globale du système en spécifiant ses principaux éléments et leurs communications bilatérales et avec l'environnement. À ce niveau, on ne s'intéresse pas aux détails des composants du système, une vue globale suffit.

- *Niveau comportemental* : la fonctionnalité d'un composant matériel à ce niveau est décrite depuis ses ports d'entrée jusqu'à ses ports de sortie. Le « timing » n'est pas requis dans le modèle; cependant, il peut y être considéré afin de servir de repère pour certaines exécutions.

- *Niveau transfert de registres (RTL)* : à ce niveau, on suppose que le système est synchrone. Des contraintes sur le « timing » et les ressources du système sont définies. Le comportement de chaque composant est donc décrit pour chaque cycle d'horloge. Les opérations des flux de contrôle et de données sont accomplies à l'aide des registres.

- *Niveau porte* : le système à ce niveau est globalement vu comme étant un circuit séquentiel. Ce dernier pourrait comprendre des bascules et des portes logiques. Un tel modèle peut être dérivé d'une description RTL tout en faisant usage des bibliothèques des différentes technologies (TTL, CMOS, etc.). Ces dernières, créées sous une optique de réutilisation, regroupent des modèles « clé en main » de bascules et de portes.

- *Niveau transistor et mise en carte* : à ce niveau, le système paraît comme étant une carte portant plusieurs composants actifs (transistors) et passifs (résistances, capacités, etc.). Une optimisation des emplacements des composants sur la surface offerte s'en suit.

La Figure I.1 illustre le processus de conception typique pour un circuit complexe. Des outils de CAO sont utilisés à chaque étape du processus de conception. Un modèle de circuit est une abstraction, c'est-à-dire une représentation qui donne les caractéristiques d'un circuit sans entrer dans les détails. Les modèles peuvent être classés suivant différents niveaux d'abstraction c'est-à-dire suivant différents niveaux de détail. La figure I.1 illustre les différents types de synthèse et les différents niveaux d'abstraction. Les axes représentent les trois domaines (ou vues) selon lesquels on peut modéliser un système:

- La **vue physique** qui permet de spécifier les caractéristiques physiques et géométriques du système. Dans le domaine physique on décrit les objets réels, par exemple les transistors d'un circuit.
- La **vue structurelle** dans laquelle un composant est décrit comme une interconnexion avec d'autres composants (structure potentielle du système physique).
- La **vue comportementale** dans laquelle un composant est décrit en définissant comment ses sorties réagissent par rapport aux entrées.

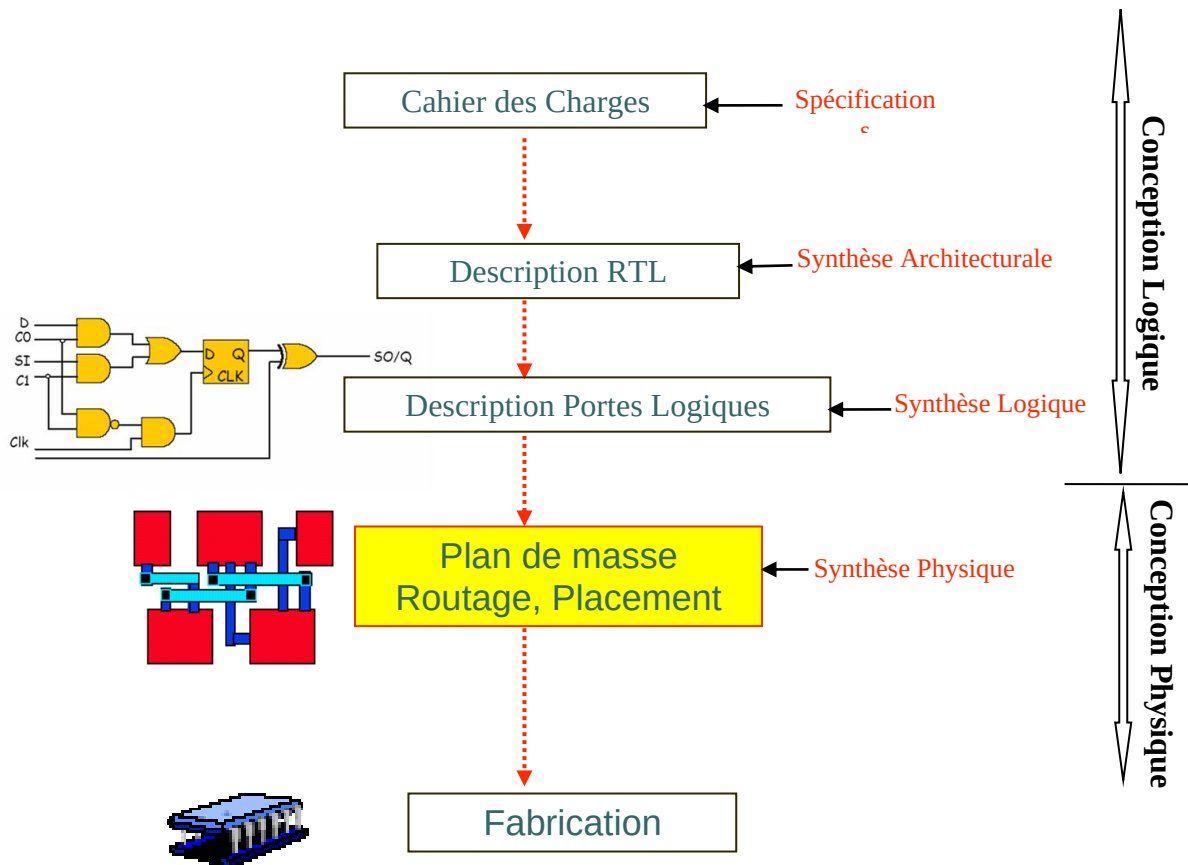


Figure I.1 Flot de Conception d'un circuit intégré

I.2.2 CONCEPTION LOGIQUE

Typiquement, les concepteurs de circuits commencent par l'écriture, à partir de spécifications textuelles (sous la forme d'un cahier des charges), d'une description au niveau algorithmique ou RTL [10] de la fonctionnalité du système en utilisant un langage de description de matériel (HDL en anglais pour *Hardware Description Language*) [13][14]. Ensuite, les outils de synthèse sont utilisés pour transformer la description HDL de niveau RTL en une description au niveau porte logique, puis au niveau transistor, et produisent finalement le masque au niveau *layout* qui est employé pour fabriquer le circuit.

Des outils de CAO sont utilisés intensivement pour exécuter les transformations entre les différents niveaux d'abstraction. Ils aident également à optimiser la description afin de prendre en compte les contraintes de vitesse, de consommation de puissance, et de surface.

Au niveau RTL, le temps est divisé en intervalles appelés étapes de contrôle. À chaque étape de contrôle, on spécifie les conditions qui doivent être testées, tous les transferts entre registres qui doivent être exécutés et l'étape suivante.

Au plus haut niveau d'abstraction, c'est-à-dire au niveau algorithmique, on utilise des variables et des opérateurs (de type mathématique) pour exprimer la fonctionnalité des composants du système. Les variables et les structures de données ne sont pas liées à des registres ou à des mémoires particuliers pas plus que les opérateurs ne le sont à des unités fonctionnelles (de types ALU ou MUX) ou à des étapes de contrôles. À ce niveau, le temps est représenté par l'ordre dans lequel les affectations des variables sont effectuées.

Une représentation structurelle permet de passer d'une représentation comportementale à une représentation physique. Une représentation structurelle (comme un schéma logique) peut éventuellement servir de représentation fonctionnelle. D'un autre côté une représentation comportementale, par exemple une expression booléenne, suggère une implantation directe, comme une structure composée de portes NOT, AND et OR. C'est pourquoi la différenciation entre niveaux et domaines n'est pas toujours claire. De même, on retrouve souvent dans la littérature le plus haut niveau d'abstraction d'une description sous le nom de niveau comportemental, ce qui ajoute à la confusion avec la vue comportementale.

La Figure I.2 illustre le flot de conception de haut niveau d'un circuit complexe. Le concepteur du circuit dispose de spécifications textuelles du circuit (cahier des charges) qui lui permettent de décrire, à l'aide d'un langage de description de matériel (tel que Verilog ou VHDL) [13] [14], son comportement.

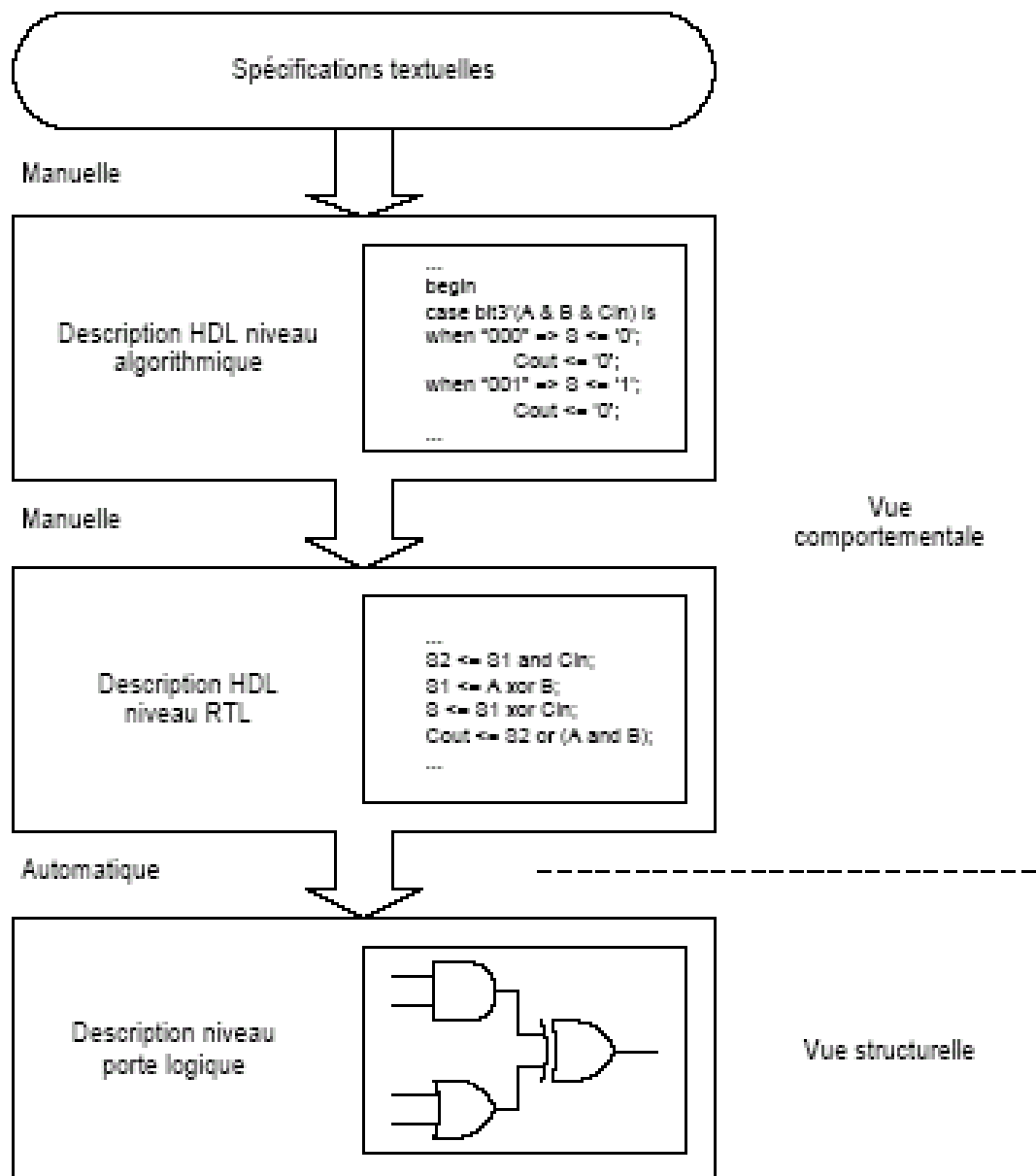


Figure I.2 Flot de conception de haut niveau

Cette description se présente sous la forme d'une description algorithmique. La transformation de cette première description en une description au niveau RTL est appelée synthèse architecturale (ou comportementale) et est effectuée manuellement par le concepteur.

L'étape suivante consiste à transformer cette description comportementale au niveau RTL en une description structurelle au niveau porte logique. Ce passage est appelé synthèse RTL. Des outils commerciaux bien connus de l'industrie (outils de synthèse logique) permettent de produire automatiquement de telles descriptions structurelles au niveau porte logique.

I.2.3 CONCEPTION PHYSIQUE

La réalisation d'un circuit intégré, doit passer d'abord par la spécification de la fonction et les performances que doit remplir ce dernier, ainsi que son environnement: brochage, conditions de fonctionnement. On définira tout ça sans ambiguïté dans le cahier des charges.

Pour déterminer la technologie appropriée pour le circuit, on consultera le cahier des charges suivant les critères de complexité, de vitesse de consommation, de coût et de disponibilité. Le choix des méthodes de conception et de la fabrication peut être lui aussi déterminant dans le choix de la technologie.

Les outils d'aide à la conception ont permis un gain dans le temps de conception et dans le coût final du produit, même si le délai de fabrication n'a pas beaucoup évolué. Actuellement, on est appelé à réaliser des circuits intégrés de plus en plus complexes avec une occupation spatiale de plus en plus petite, c'est là qu'intervient la conception assistée par ordinateur (CAO), elle est indispensable dans le domaine des circuits intégrés VLSI . La complexité de ces circuits rend leur réalisation manuelle difficile. Il a donc fallu développer des outils informatiques propres à l'élaboration de circuits intégrés.

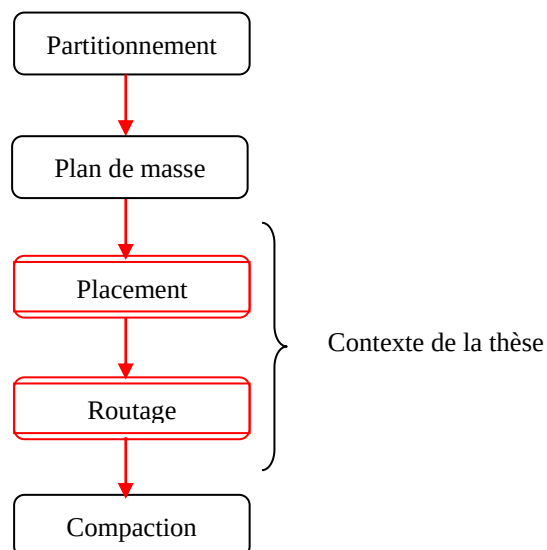


Figure I.3 Phase de conception physique

On peut concevoir un circuit intégré grâce à ces différentes phases [6]:

- 1- La composition d'un schéma formé de pièces élémentaires effectuant les fonctions désirées. Pour ce faire, on aura recours aux simulateurs. Cet outil calcule les caractéristiques réelles (tensions, courants . . .) du circuit qui lui est fourni sous forme symbolique .
- 2- Réalisation du plan de masse (vue d'ensemble du circuit) et le dessin de masques (dessins devant reproduire les divers masques lors des opérations de photogravure). Dans cette phase on fera appel à des outils informatiques de dessin interactif ou automatique.
- 3- Examen du dessin de masques : C'est-à-dire voir s'il est conforme aux règles de dessin de la technologie choisie. S'assurer que le circuit est conforme aux exigences du cahier des charges en faisant une simulation électrique ou logique, pour ce faire, on extrait un schéma électrique ou logique à partir du dessin de masque.
- 4 - Le dessin de masque est reporté sur le silicium grâce à une suite d'étapes technologiques
- 5- Le test du circuit.

I.2.3.1 STYLE DE CONCEPTION

La méthode de conception est la façon de réaliser la fonctionnalité d'un circuit à l'aide d'instances placées et routées. Elle définit le coût de développement du circuit qui dépend du nombre d'étapes informatiques nécessaires à sa traduction.

Le style de conception est la façon d'implanter la fonctionnalité d'un circuit dans le silicium. Il définit le coût de fabrication du circuit car il détermine le nombre d'étapes technologiques nécessaires à son intégration.

Avant l'étape de conception physique, le concepteur doit choisir la technologie et la méthodologie de disposition physique du circuit. Une technologie est un processus particulier de fabrication de circuit intégré. *Le modèle de méthodologie ou de conception* de la disposition de cellules détermine la construction de masques.

Des technologies sont normalement définies par la taille minimum de dispositif (typiquement plus petite largeur permise de couche), nombre et type de couches de routage, ainsi que le type des dispositifs possibles au processus [7][9].

Actuellement, les modèles physiques populaires de conception de VLSI sont les circuits prédiffusés (Gate Array), les circuits précaractérisés (Standard Cell) et les circuits sur mesure (Full Custom) comme représentés sur la figure I.4.

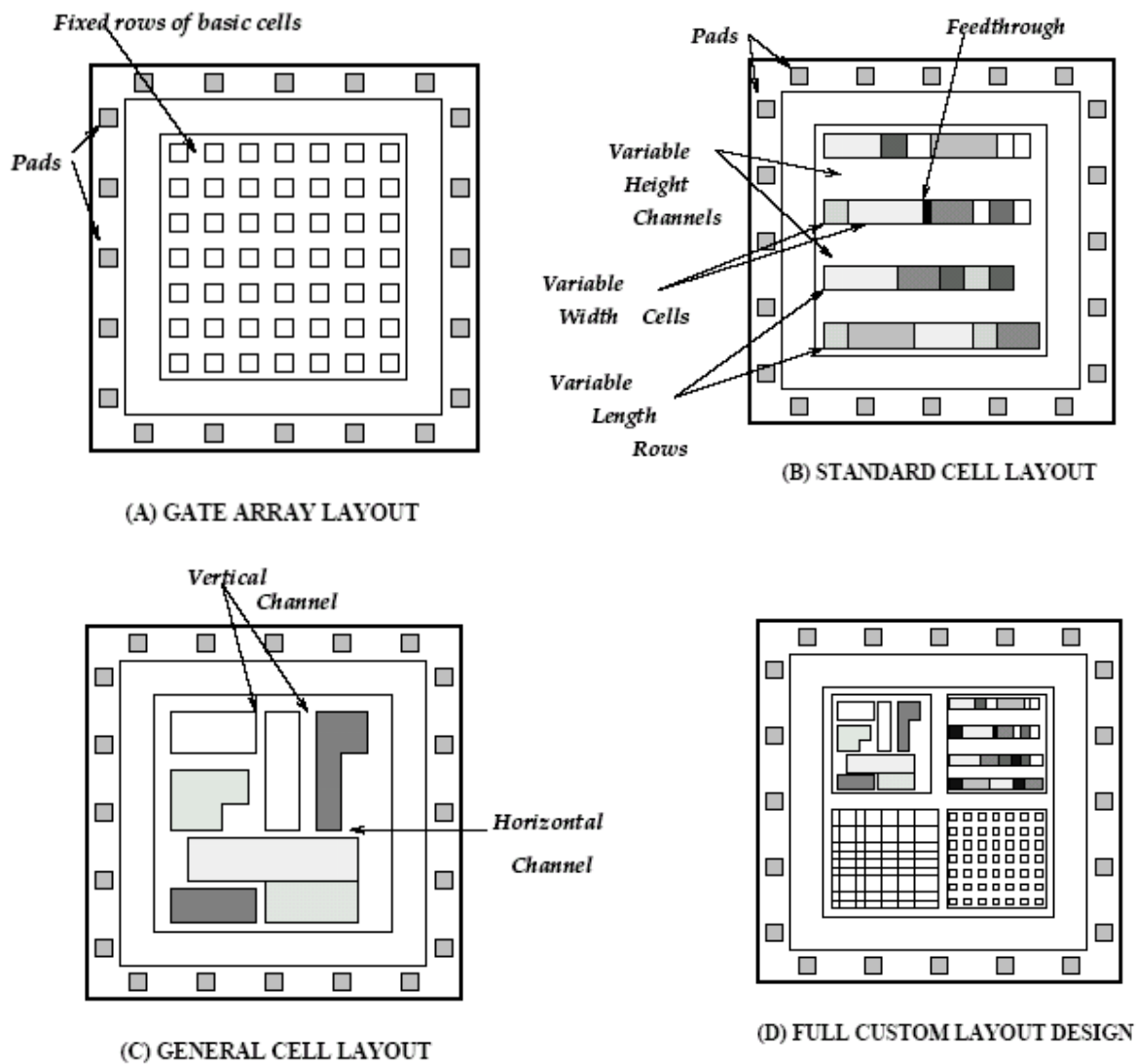


Figure I.4 Les différents styles de conception

a. Conception des circuits prédiffusés (gate array)

Dans la conception des circuits prédifusés (Réseaux de transistors déjà implantés dans le silicium, mais non interconnectés. C'est l'interconnexion des transistors qui personnalise le circuit en fonction de l'application visée), la surface entière est préfabriquée avec un choix de portes ou de cellules identiques. Comme représenté sur la figure I.4a, les cellules sont séparées par les espaces verticaux et horizontaux appelés les canaux.

Style de conception	Prédifusés
Caractéristiques	• Une préfabrication partielle des plaquettes par le fondeur (seul manque la métallisation qui permet de réaliser l'interconnexion entre les transistors) • Une matrice est caractérisée par son nombre de plots (E/S) et de transistors. La surface est figée. • Conception à partir d'une bibliothèque. Résultat: Dessin des niveaux de connexion (contacts, métaux)
Avantages	• Faible investissement CAO (éditeur de schémas, simulateur, routeur auto.) • Rapidité de conception • Faible coût de fonderie • Rapidité d'obtention de prototypes
Inconvénients	• Choix limité de cellules • Faible densité d'intégration (car faible taux de remplissage) • Faible complexité • Faible vitesse (due aux capacités d'interconnexion)

Tableau I.1 Conception des circuits prédifusés

Les prédifusés sont de moins en moins utilisés, et sont remplacés par les FPGAs [8].

b. Conception des circuits précaractérisés (Standard Cell)

Dans la disposition de circuits précaractérisés (figure I.4b), toutes les cellules ont la même taille mais différentes largeurs; et les cellules ont fixé les raccordements du côté gauche et droit (des horloges et/ou puissance) qui aboutent l'un avec l'autre.

Style de conception	Précaractérisés
Caractéristiques	• Assemblage de fonctions prédéfinies et précaractérisées électriquement par le fabricant (éléments de bibliothèques) • La personnalisation du circuit se fait par placement des cellules et optimisation du routage • Conception de tous les masques de fabrication (nombre de transistors et d'E/S non figé) • Grande variété de cellules (compteurs, ALU, mémoires, ...) plus ou moins complexes. • Possibilité d'exploiter toute la surface de la puce • Surface 2 fois inférieure à un prédiffusé • Surface 3 fois supérieure à un full custom
Avantages	• Investissement CAO modéré (éditeur de schémas, simulateur logique uniquement, placement/routeur automatique) • Rapidité de conception • Pas de perte de silicium • Rapidité d'obtention de prototypes
Inconvénients	• Choix limité de cellules • Complexité moyenne • Début de concurrence des FPGAs

Tableau I.2 Conception des circuits précaractérisés

c. Conception des circuits cellules générales

Le modèle de conception de cellules générales présenté dans la figure I.4c est une généralisation du modèle de conception de cellules standard. Les cellules (fournies par une bibliothèque ou construites selon les exigences du système de conception) peuvent être grandes et ont une forme irrégulière.

d. Conception des circuits sur mesure (Full Custom)

La conception de circuits sur mesure (figure I.3d) exige habituellement des concepteurs expérimentés puisque le manque de contraintes rend des outils de synthèses difficiles à se développer. Cependant, cette conception est largement répandue pour de plus petites cellules qui sont entrées dans des outils de synthèses.

Style de conception	Sur mesure
Caractéristiques	• Recherche d'une solution optimale en termes de densité et de performance: la totalité des opérations de conception est effectuée de façon spécifique, adaptée aux exigences de l'utilisateur. • Justification : grandes séries (démarche effectuée pour la plupart des produits catalogues des fabricants, réduction des coûts) • Le placement des blocs fonctionnels et le routage des interconnexions est effectué avec beaucoup plus d'interventions manuelles (éditeur de layout) pour atteindre l'optimisation au niveau de chaque transistor. • Conception de tous les masques de fabrication
Avantages	• Performances élevées • Haute densité d'intégration • Grande complexité
Inconvénients	• Temps de conception élevé • Fort investissement CAO (éditeur de schémas, simulateur électrique, règles de dessin, placeur/routeur automatique, extracteur de masques) • Coût de fonderie élevé

Tableau I.3 Conception des circuits sur mesure

Le tableau I.4 donne une comparaison entre les différents styles de conception.

	Style			
	Sur mesure	Cellule standard	Porte logique	FPGA
Taille des cellules	Variable	Hauteur fixe	Fixe	Fixe
Type de cellules	Variable	Variable	Fixe	Programmable
Placement de cellules	Variable	Dans des rangées	Fixe	Fixe
Interconnexions	Variable	Variable	Variable	Programmable

	Style			
	Sur mesure	Cellule standard	Porte logique	FPGA
Surface	Compacte	Compacte à modérée	Modérée	Large
Performance	Elevée	Elevée à modérée	Modérée	Faible
Couches fabrication	Tout	Tout	Routage couche	Aucune

Tableau I.4 Comparaison des différents styles de conception

La figure I.5 donne le coût du système en fonction du volume de production pour les différents styles de conception.

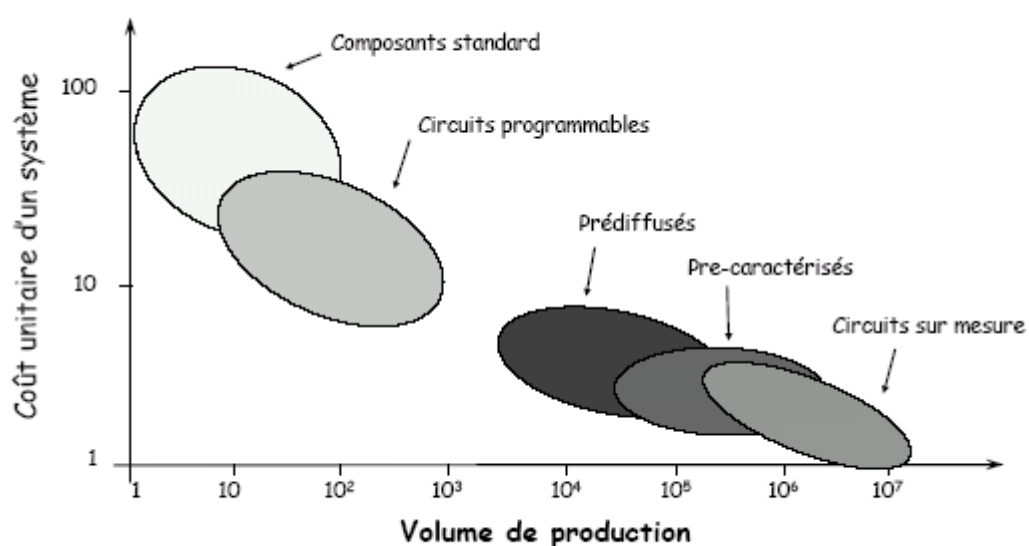


Figure I.5 Complexité de la conception des circuits intégrés

Dans le cadre de notre thèse, les algorithmes utilisent la disposition de cellules standard.

I.2.3.2 PLAN DE MASSE

C'est l'estimation grossière de la surface et de la forme de chaque bloc. Cette estimation topologique permet un bon assemblage des blocs, prévoir les interfaces internes et externes du circuit et de préciser les peignes d'alimentation.

Pour faciliter l'obtention du plan de masse, on utilise des blocs préalablement testés sur d'autres circuits.

En d'autres termes, le problème du plan de masse c'est de prévoir *les positions* et *les formes* des modules au début de la conception pour optimiser certains paramètres liés aux performances des circuits tels que, la surface du circuit, la longueur totale de connexion ...

Pour comparer le placement du plan de masse, nous pouvons dire que tous les deux déterminent des positions de bloc pour optimiser l'exécution de circuit, alors pour le plan de masse: les détails comme la forme des blocs, la position des peignes d'E/S, etc. ne sont pas encore fixes (des blocs avec la forme flexible s'appellent les blocs doux). Alors que pour le placement : les détails comme la forme de module et la position des peignes d'E/S sont fixes (des blocs sans la flexibilité dans la forme s'appellent les blocs durs).

La problématique du plan de masse peut être définie comme suit:

-Entrée:

n Blocs avec une surface de $A_1, \dots, A_n$ et des limites r_i et s_i

-Sortie:

Coordonnées (x_i, y_i) , avec une largeur w_i et une hauteur h_i pour chaque bloc tels que

$$h_i * w_i = A_i \text{ et } r_i \leq h_i / w_i \leq s_i$$

-Objective:

Pour optimiser les performances du circuit.

La fonction objective utilisée est généralement une somme pondérée de la surface du circuit avec la longueur totale de connexion :

$$\text{Coût} = \alpha A + \beta L$$

où A est la surface totale, L étant la longueur totale de connexions, α et β sont des constantes.

Puisque la longueur totale de connexions ne sera connue qu'après l'étape de routage, de même que la position des blocs, alors quelques évaluations de la longueur sont utilisées :

- évaluation de centre à centre
- évaluation de moitié du périmètre

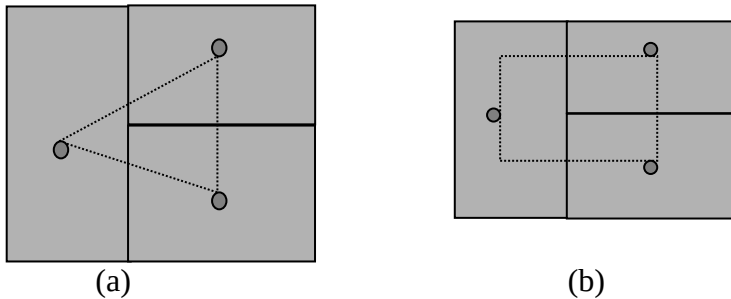


Fig. I.6 Evaluation de la longueur de connexion (a) centre à centre (b) moitié du périmètre

I.2.3.4 DESSIN DE MASQUES

Dans cette phase de conception, on dessine les masques de chaque élément du circuit, et on les assemble conformément au plan de masse. On distingue plusieurs méthodes telles que :

- Le dessin à partir de cellules précaractérisées (cellules testées sur d'autres circuits) .
- Les dessins symboliques où chaque élément de base est représenté par un symbole graphique.
- Le dessin symbolique où chaque " site " technologique (connexion, transistor . . .) est représenté par un symbole graphique simple.
- Le " dessin au micron " où chaque rectangle de masque est décrit et placé.

I.2.3.5 PLACEMENT

Dans cette étape de la conception du circuit intégré, il s'agit de trouver un placement optimal pour un nombre donné d'éléments, soit sur un certain nombre de positions, soit sur une surface S , où ($S \geq S_1, S_2, \dots, S_n$, où les S_i sont les surfaces de chaque élément). On jugera qu'un placement est de bonne qualité si le tracé de connexion soit possible. Différentes méthodes sont utilisées dans le placement et qui sont présentées en détail dans le chapitre 2.

I.2.3.6 ROUTAGE

Une fois toutes les cellules placées sur une surface, on procède à la phase du routage, c'est à dire l'implantation de la totalité des interconnexions et on se préoccupera d'optimiser la longueur et la densité des connexions (plus de détails, voir chapitre 3).

I.2.3.7 COMPACTION

La compaction est une étape facultative pour réduire la surface du circuit intégré. Si une conception a des violations initiales de règle de conception, l'espacement pourrait augmenter réellement la taille du circuit. La compaction réduit au minimum la surface d'un circuit intégré sans changer sa topologie.

I.3 SIMULATION

La simulation est essentielle dans la conception des circuits intégrés VLSI en tant qu'outil de validation des choix du concepteur. Pour des raisons de compétitivité, elle doit être la plus rapide et la plus fiable possible. On distingue généralement trois types de simulation:

- La **simulation analogique ou électrique**: elle traite des signaux continus dans le temps et est utilisée pour déterminer les performances électriques des circuits. Le niveau de précision demandé est généralement élevé et de ce fait, le temps de simulation des circuits comportant un grand nombre de transistors est souvent critique.
- La **simulation digitale**: couramment utilisée pour les circuits digitaux VLSI, elle manipule des signaux discrétisés et quantifiés (0,1, indéterminé ou X,...) et se caractérise par une très grande rapidité.
- La simulation **mixte analogique-digitale**: elle permet d'accélérer l'étude temporelle des circuits mixtes complexes.

I.3.1 SIMULATEURS ELECTRIQUES

Ce sont les premiers à être utilisés. Ils permettent d'étudier la réponse des circuits intégrés à la suite de différentes excitations.

Tous les programmes développés utilisent les mêmes éléments de base pour la description des composants des circuits. Chaque programme reconnaît un nombre limité d'éléments de base. Plus le programme reconnaît d'éléments de base plus son spectre d'utilisation sera large.

Il est important de toujours connaître les limitations des modèles utilisés, les dispersions possibles des paramètres d'entrée et d'estimer ainsi le degré de fiabilité des résultats de simulation.

La référence en matière de simulateur analogique de circuits intégrés est le programme SPICE, développé à l'université de Berkeley. Des simulateurs plus récents, tels que Saber, Eldo, Spectre proposent, quant à eux, des langages de description comportementale, qui facilitent l'écriture de nouveaux modèles. Ils possèdent d'autre part une bibliothèque étendue de primitives comportant des modèles comportementaux de macro-blocs (amplificateurs, comparateurs, commutateurs analogiques, portes logiques, etc...).

I.3.2 SIMULATEURS LOGIQUES

Ce sont des logiciels qui permettent d'analyser le fonctionnement logique d'un bloc ou d'un circuit complet. L'ensemble à analyser peut-être décrit à partir d'opérateurs de base (NAND, NOR, ...) ou de blocs complexes. On utilise en général une description hiérarchisée, ce qui permet de définir des blocs de base qui pourront être utilisés directement dans d'autres blocs plus importants.

I.3.3 SIMULATEURS ANALOGIQUE/DIGINALE

La simulation mixte analogique-digitale permet d'étudier le comportement temporel de systèmes complexes en un temps extrêmement réduit par rapport à une simulation uniquement électrique. Ce type de simulation est en effet basé sur l'abstraction de la partie digitale à un niveau fonctionnel logique. Pour cette partie, les grandeurs étudiées ne sont donc plus électriques mais numériques et sont caractérisées par leurs changements d'état.

Une simulation mixte peut être décomposée en trois phases distinctes:

- Une phase d'**élaboration**: elle correspond au partitionnement du circuit mixte en blocs distincts analogiques et digitaux, chaque partie étant traitée par les algorithmes concernés. Aux interfaces entre les deux parties, doivent être placés des modèles plus ou moins élaborés de **convertisseurs** A/D et D/A, qui assurent la correspondance des données entre les algorithmes analogiques et digitaux.
- Une phase d'**initialisation**: il s'agit de déterminer le point de fonctionnement du système, c'est à dire l'état initial de toutes les grandeurs mises en jeu (tensions, courants, états logiques). Cette recherche est indispensable au simulateur analogique et correspond à une analyse DC. Pour la partie digitale, cette notion dépend du simulateur: cela peut correspondre soit à une initialisation (solution au temps 0), soit à une certaine durée, appelée temps de *setup*, au bout de laquelle un état stable est trouvé.
- Une phase de **simulation**: elle doit résoudre les problèmes de **synchronisation** des algorithmes électriques et digitaux, qui ont des gestions différentes du pas de temps.

I.3.4 SIMULATEURS FONCTIONNELS

Ils sont destinés à vérifier les algorithmes proposés pour réaliser une fonction à un niveau d'abstraction le plus élevé possible.

La simulation fonctionnelle pour les circuits analogiques peut aller du simulateur électrique possédant des modèles élaborés (amplificateurs, comparateurs, . . .) jusqu'à des logiciels analysant la fonction de transfert d'un filtre de façon symbolique.

I.4 FABRICATION ET TEST

Cette phase sert à la validation de la conception : le circuit fonctionne-t-il conformément aux simulations effectuées ?, le circuit est-il bien conforme au schéma de départ ?.

La simulation logique, ou logico-électrique réalisée pendant la conception permet en chaque point de mesure de comparer l'état de l'information réelle avec celle espérée. On effectue le cycle fabrication-validation, puis correction (si nécessaire) jusqu'à l'obtention d'un circuit conforme au cahier des charges.

Étant donné la complexité des circuits actuels, il n'est pas possible de les tester efficacement si le test n'a pas été prévu dès leur conception. Le terme Conception en Vue du Test (*Design For Testability*- DFT) [12] recouvre toutes les techniques utilisées en vue de rendre le test des circuits économiquement viables. Parmi les DFT, la technique dites de « *Boundary-Scan* » est la plus récente et la plus efficace. Cette technique consiste à ajouter un certain nombre de composants électroniques au circuit initial afin d'augmenter sa contrôlabilité et son observabilité.

Ainsi, développer une approche efficace pour tester des programmes VHDL [11] au niveau algorithmique qui intègre la technique du « *Boundary-Scan* », et créer un logiciel qui automatise le processus de test serait une contribution significative pour les utilisateurs d'outils de CAO.

REFERENCES DU CHAPITRE 1

- [1] Z. Juneidi "Outils CAO pour microsystemes", thèse de Doctorat INP Grenoble 2003
- [2] K.R. Larker, W.M.C Sansen "Design of Analog Integrated Circuits and Systems" Mac Graw Hill 1994.
- [3] William P. Swartz, Jr. "Automatic Layout of Analog and Digital Mixed Macro/Standard Cell Integrated Circuits" PHD thesis, 1993.
- [4] C. Mead and L. Conway, "Introduction to VLSI Systems", Addison-Wesley, 1980.
- [5] Dillinger, Thomas E. "VLSI Engineering". Englewood Cliffs, NJ: Prentice Hall, 1988.
- [6] B.T. Preas, and M. J. Lorenzetti, editors, "Physical Design Automation of VLSI Systems", (Menlo Park: The Benjamin/Cummings Publishing Company, Inc., 1988.
- [7] W. Swartz, H. Khan, D. A. Thomas, C. R. Giuffre, M. deWit, T. Pavey, C. McIntosh, and W. H. Banzhaf, "CMOS RAM, ROM, and PLA Generators for ASIC Applications," Proc. Custom Integrated Circuits Conference, 1986, pp. 334-338.
- [8] Steven Derriern, "Étude quantitative des techniques de partitionnement de réseaux de processeurs pour l'implantation sur circuits FPGA", Thèse de doctorat, Université de Rennes 1, 2002

- [9] Fabrice PICOT, "Contribution à l'Optimisation d'un Flot de Conception Submicronique à base de Cellules Précaractérisées", Thèse de doctorat, Université de Montpellier II Sciences et techniques du Languedoc, 2002
- [10] M. Aichouchi "Etude des liens entre la synthèse architecturale et la synthèse au niveau transfert de registres". Thèse de Doctorat, INP Grenoble, 1994.
- [11] Kevin Skahill, Jay Legenhausen, Ron Wade, Corey Wilner, Blair Wilson "VHDL for programmable logic",. 1996.
- [12] Crouch, Alfred L., "Design for Test for Digital Ic's and Embedded Core Systems", Publisher: Prentice Hall PTR; 1st edition (June 15, 1999)
- [13] J.R. Armstrong and F. Gail Gray, "VHDL Design: Representation and Synthesis", Prentice Hall, 2000.
- [14] Ludovic Jacomme, "Analyse sémantique de descriptions VHDL synchrone en vue de la synthèse", Thèse de Doctorat, Paris VI 1999.

Les Réseaux de KOHONEN Appliqués au Placement de Cellules

II.1 INTRODUCTION

Le développement des outils de la CAO a rendu possible l'automatisation entière du processus de conception. Ceci est possible avec l'utilisation du style de conception (portes logiques, cellules standards, ...) couplé avec un ensemble efficace de programmes (Software) pour les opérations de placement et de routage automatiques.

Les réseaux de neurones artificiels sont étudiés depuis plusieurs années. Leur efficacité permet d'espérer de hautes performances. Les domaines privilégiés de ces techniques restent la reconnaissance et la classification. Différentes applications d'optimisation sont également étudiées sous l'angle des réseaux de neurones artificiels. Ils permettent d'appliquer des algorithmes heuristiques répartis.

Dans ce chapitre, une solution au problème de placement des différentes cellules lors de la réalisation d'un circuit intégré est proposée en utilisant le réseau de KOHONEN [1]. En effet, des études ont montré qu'un type particulier de réseaux de neurones artificiels, le réseau de KOHONEN, peut être utilisé pour positionner des cellules de taille et de forme identiques sur un plan de masse sans rails de connexions.

Dans ce chapitre, nous avons présenté un premier algorithme, avec des hypothèses simplificatrices, qui n'a pas donné des résultats satisfaisants, ce qui nous a poussé à l'améliorer pour devenir un algorithme très performant. Ensuite, nous avons abordé la phase de l'extrapolation de l'algorithme, où les cellules étaient de taille différente. Nous avons introduit la notion de borne où chaque cellule est représentée par un ensemble de bornes, et où les connexions se faisaient entre ces mêmes bornes.

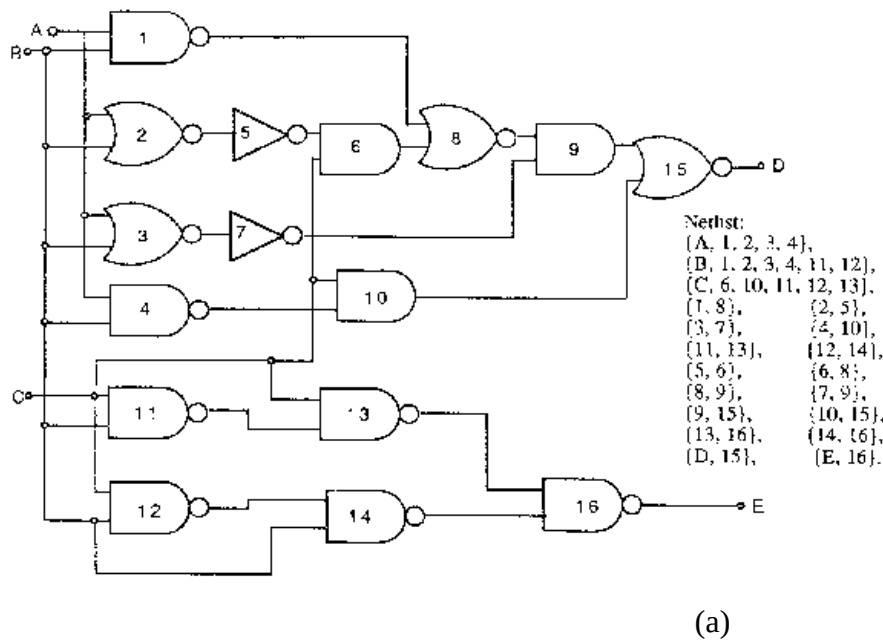
Nous avons présenté le cas fondamental du problème de placement ainsi que la méthode proposée par A. HEMANI et A. POSTULA [2] pour le résoudre en utilisant les réseaux de KOHONEN. Nous consacrons une partie à une critique de cette méthode, une adaptation et une amélioration de l'algorithme est présentée, ainsi qu'une large expérimentation. Enfin, Nous extrapolons cet algorithme pour résoudre le problème de placement des cellules de tailles différentes.

II.2 CLASSIFICATION DES ALGORITHMES DE PLACEMENT

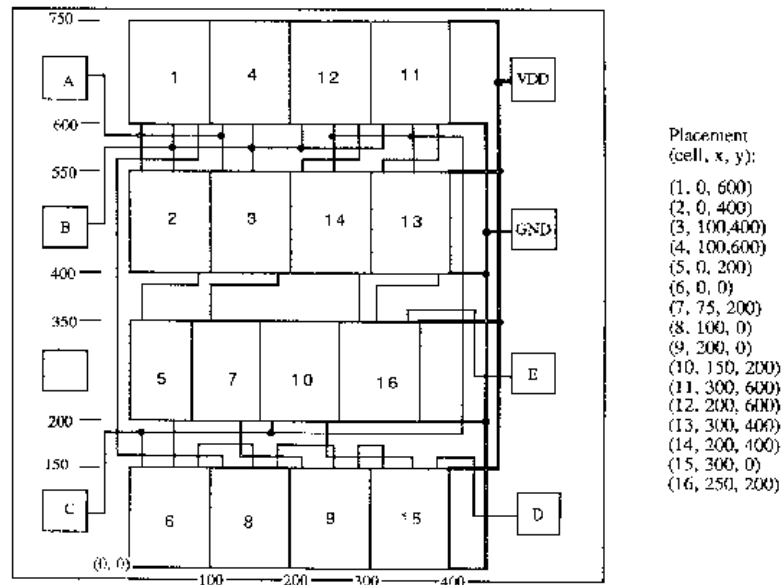
Le problème du placement peut être défini comme suit [3]: soit un circuit électrique constitué de modules avec des entrées/sorties prédéfinies, il faut construire un layout indiquant les positions des modules de telle sorte que l'estimation de la longueur des connexions et la surface du circuit soit minimisée. L'entrée du problème correspond à la description des modules (taille) et le fichier de connexion (netlist) décrivant les interconnexions entre les bornes de ces modules. La sortie est une liste de coordonnées (x, y) de tous les modules. La figure II.1 [3], donne un exemple de placement où le schéma du circuit de la figure II.1 (a) est placé, suivant le style de conception cellules standard, comme montré dans la figure II.1 (b).

Le but principal des algorithmes de placement est de minimiser la surface globale du circuit ainsi que l'estimation de la longueur des connexions. Dans le style de conception cellules standard et portes logiques, la surface totale est approximativement égale à la surface des modules plus la surface occupée par les connexions. Dans ce cas, minimiser la longueur de connexion revient à minimiser la surface du circuit.

Dans le style de conception macro-cellules, l'irrégularité des formes des blocs qui ne sont pas alignés l'un à côté de l'autre fait apparaître des surfaces perdues. Ceci joue un rôle important dans la détermination de la surface globale du circuit.



(a)



(b)

Figure II.1 Problème de placement de cellules

(a) Entrée: fichier de connexion

(b) Sortie : Cordonnées des modules.

Pour un placement acceptable, un autre critère est à prendre en considération est le fait qu'il devrait être réalisable physiquement; c'est-à-dire, (1) les modules ne devraient pas recouvrir, (2) ils

ne devraient pas dépasser les frontières du circuit, (3) les cellules standard devraient être confinées aux rangées dans des positions prédéterminées, et (4) les portes devraient être affectées aux rangées dans des points de grille.

En pratique, il s'agit de définir une fonction de coût ou une fonction objective, qui comprend la somme de tous ces critères, l'estimation de la longueur de connexion, surface totale du circuit, et ainsi de suite. Le but de l'algorithme de placement est de déterminer un placement avec un minimum de coût possible.

Les algorithmes de placement peuvent être divisés en deux catégories : méthodes constructives et méthodes itératives [3][4][5]. Un algorithme constructif accumule un placement complet basé sur un placement partiel ou vide. Un algorithme d'amélioration itératif commence par un placement initial et le modifie à plusieurs reprises pour l'amélioration d'une fonction coût.

Un algorithme constructif est généralement très rapide, mais souvent les résultats donnent un placement de qualité faible. Typiquement, une première cellule est choisie et placée. Ensuite, les cellules sont choisies une par une, selon leur connectivité aux cellules placées. La cellule choisie est alors placée à un endroit vide près des cellules déjà placées.

Un algorithme itératif produit typiquement le bon placement aux dépens d'énormes quantités de temps de calcul. Il débute avec un placement complet qu'il modifie à plusieurs reprises. Une modification peut mener à un meilleur ou à un plus mauvais placement. L'algorithme est basé sur un mécanisme de commande pour décider si l'original ou le placement modifié devrait être employé comme point de départ pour la prochaine itération. Aujourd'hui, Le recuit simulé est probablement la méthode la plus connue pour ce genre de placement. Il s'est avéré que le recuit simulé peut converger à une quantité suffisante donnée par résultat globalement optimal de temps de calcul. Actuellement, l'amélioration itérative inclut les algorithmes génétiques et quelques techniques "force dirigée" de placement.

Il existe, principalement, six techniques principales d'algorithmes de placement:

- 1) recuit simulé [6][7];

- 2) force dirigée [8][9];
- 3) partitionnement [10];
- 4) optimisation numérique [3];
- 5) algorithme génétique [12][13];
- 6) réseaux de neurones [2][3][14][15].

Les deux premières classes des algorithmes doivent leur origine aux lois physiques, la troisième et la quatrième aux techniques analytiques, alors que les deux dernières classes des algorithmes sont dérivées de phénomènes biologiques. Chacune de ces techniques a ses propres avantages et inconvénients.

II.3 MOTIVATION

Le problème du placement des cellules est posé lors de la conception de circuit intégré. Une fois les fonctions du circuit seront définies, on peut déduire un schéma électrique remplissant la fonction souhaitée. Le schéma est composé d'un ensemble d'unités fonctionnelles interconnectées.

Ces unités sont elles même composées d'éléments de base comme des transistors ou des lignes de connexions, elles sont dessinées et optimisées individuellement. Elles deviennent donc des unités élémentaires que nous appellerons cellules. Maintenant que le circuit intégré est construit en reliant physiquement des cellules, le problème du placement est de définir comment les cellules seront placées pour maximiser les performances de ce circuit. Il s'agit de minimiser la surface requise.

Les algorithmes et les méthodes connues pour résoudre ce problème souffrent de l'accroissement du temps de calcul en fonction du nombre des cellules. Pour résoudre ce problème, on doit imposer des contraintes pour la recherche d'une solution.

Dans cette partie, nous analyserons un cas élémentaire du problème de placement en nous basant sur l'article de A HEMANI et POSTULA [2]. Ainsi, nous admettrons les hypothèses simplificatrices suivantes:

- Toutes les cellules sont de taille et de forme identiques.
- Il n'existe pas de rails de connexion: les liaisons intercellulaires se font en utilisant l'espace libre entre les cellules.
- Les cellules sont disposées sur une grille de pas unitaire.
- La fonction coût minimisée ne prend en compte que la longueur des connexions.

La surface réservée aux connexions sera donc aussi minimisée. Ceci devrait avoir un effet bénéfique sur la surface totale du circuit. Un circuit composé de 9 cellules aura, par exemple, l'allure décrite dans la figure II.2.

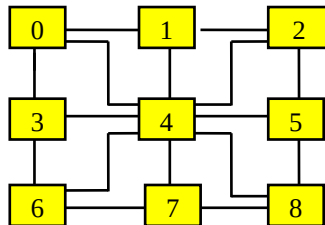


Figure II.2 Interconnexion dans un circuit à Neuf cellules

La matrice de connexions qui regroupe les connexions entre ces cellules sera donc symétrique, l'intensité des connexions est une fonction de divers paramètres tels que le nombre de lignes reliant deux cellules. Une valeur importante signifie qu'il y'a une forte connexion entre ces deux cellules, une valeur nulle signifie qu'il n'y a pas de connexion entre ces deux cellules. La matrice de connexions du circuit de la figure II.2 a, par exemple, la forme suivante:

	0	1	2	3	4	5	6	7	8
0	0	5	0	5	2	0	0	0	0
1	5	0	5	0	5	0	0	0	0
2	0	5	0	0	2	5	0	0	0

3	5	0	0	0	5	0	5	0	0
4	2	5	2	5	0	5	2	5	2
5	0	0	5	0	5	0	0	0	5
6	0	0	0	5	2	0	0	5	0
7	0	0	0	0	5	0	5	0	5
8	0	0	0	0	2	5	0	5	0

Tableau II.1: Matrice de connexions (figure II.2).

II.4 ENONCE DU PROBLEME

On peut résumer l'énoncé du problème de placement de la façon suivante [2]:

Etant donné $K = \{0, \dots, C-1\}$, ensemble de cellules C_k , $k \in K$.

$I = \{0, \dots, N-1\}$, ensemble de positions n_i , $i \in I$.

Une matrice de connexions $CM = (cm_{kl})$, $k, l \in K$.

On cherche un ensemble C de couples $(k,i) \forall k \in K$ avec $i \in I$ tel que

$$\sum_{k \in K} \sum_{l \in K} d_{kl} \cdot cm_{lk} \quad (II.1)$$

Soit minimum $\forall k, l \in K, \forall i, j \in I : d_{kl} = d_{ij}$ avec $(k,i) \in C$ et $(l,j) \in C$.

On utilisera les réseaux de KOHONEN pour résoudre ce problème et on va procéder comme suit:

- Un réseau de N neurones disposés sur un maillage de dimension 2 où chaque neurone représente un emplacement possible pour une cellule.
- Un neurone d'entrée e connecté à chaque neurone du réseau.
- Un vecteur stimulus au temps t $x(t) = (x_1(t), x_2(t), \dots, x_c(t))$
- Un vecteur de poids synaptiques qui caractérise un neurone i au temps t $w_i(t) = (w_{i1}(t), w_{i2}(t), \dots, w_{ic}(t))$.

II.5 ALGORITHME

Le problème est de minimiser la fonction d'énergie, en minimisant la contribution de chaque cellule C_k . La Fonction d'énergie est :

$$\sum_{k \in K} \sum_{l \in K} d_{kl} cm_{lk} \quad (\text{II.2})$$

Contribution de la cellule est

$$c_k = \sum_{l \in K} d_{kl} cm_{lk} \quad (\text{II.3})$$

L'idée générale de l'algorithme est de placer chacune de ces cellules à la meilleure position possible, pour optimiser cette somme. Tout d'abord, on a un réseau de neurones vide, après un prétraitement et des initialisations, une cellule est présentée au réseau pour être placée.

Une boucle itérative est entamée, avec chaque itération, une cellule qui se présente jusqu'à ce qu'on atteigne un état stable, c'est à dire jusqu'à ce que chaque cellule soit associée à un neurone qui produit la plus petite contribution.

Donc l'algorithme de A. HEMANI et A. POSTULA qui est obtenu par affinage des étapes correspondantes de l'algorithme général de KOHONEN, procède en les 4 étapes suivantes:

II.5.1 PRETRAITEMENT

Le but du prétraitement est de transformer la matrice de connexions afin de faire apparaître les connexions indirectes qui existent entre des cellules qui ne sont pas connectées directement.

Supposons qu'on a une cellule c_1 qui est fortement connectée à deux autres cellules c_2 et c_3 qui ne sont pas connectées entre elles (figure II.3), alors par transitivité si les cellules sont si proches de c_1 alors elles sont forcément proches l'une de l'autre. Il existe donc un lien implicite entre la cellule c_2 et c_3 .

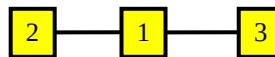


Figure II.3 Lien implicite entre les cellules

A. HYPOTHESE

Deux cellules connectées à une même cellule c_k mais n'étant pas connectées entre elles seraient placées à une distance inversement proportionnelle à leur intensité de connexion à c_k . En prenant comme unité de distance la distance à laquelle se trouve la cellule la plus fortement connectée à la cellule c_k par une intensité cm_{lk} se trouve à une distance :

$$\text{dist}(k,l) = \max(cm_{mk})/cm_{kl}$$

B. ALGORITHME

Begin

For $k=0$; $k < \text{nmb_cel}$; $k++$

For $l=k+1$; $l < \text{nmb_cel}$; $l++$

$\text{max_d} = 0$

If $\text{cm}_{kl} = 0$ **Then**

For $m=0$; $m < \text{nmb_cel}$; $m++$

If $\text{cm}_{kl} \neq 0$ and $\text{cm}_{ml} \neq 0$ **Then** $\text{dist}(k,l) = \max(\text{cm}_k)/\text{cm}_{km} + \max(\text{cm}_m)/\text{cm}_{ml}$

If $\text{max_d} < \text{dist}(k,l)$ **Then** $\text{max_d} = \text{dist}(k,l)$

End_If

End_If

End_For

$\text{cm}_{kl} = \max(\text{cm}_k)/\text{max_d}$

End_If

End_For

End_For

End.

II.5.2 CREATION DE LA SEQUENCE D'ENTREE

La création de la séquence d'entrée consiste à placer toutes les cellules les unes après les autres en présentant après chaque cellule, les cellules qui y sont connectées dans l'ordre décroissant de l'intensité de connexion. L'algorithme de la séquence d'entrée d'une cellule est le suivant :

ALGORITHME

Begin

$k = \{1, \dots, c\}$ $k = \text{Random}(\text{nmb_cel})$ Pour chaque cellule K faire

$S_k = \langle k, l_1, l_2, l_3, \dots, l_L \rangle$ tel que $\text{cm}_{kl1} \geq \text{cm}_{kl2} \geq \dots \text{cm}_{klL} > 0$

Séquence = Séquence $\subset$ S_k

End.

Où le symbole $\subset$ représente la concaténation de deux séquences.

II.5.3 INITIALISATION DE LA MATRICE DES POIDS SYNAPTIQUES

La matrice des poids synaptiques est initialisée par de faibles valeurs positives qu'on appellera Rnd_val .

II.5.4 ITERATION

C'est l'étape la plus importante dans cet algorithme, on a trois phases principales:

II.5.4.1 PRESENTATION D'UN STIMULUS

Nous définirons époque comme étant un traitement complet de la séquence d'entrée. La présentation d'un stimulus se fait comme suit :

$k \leftarrow \text{seq}[1]$; début d'une époque

$k \leftarrow \text{seq}[t]$; dans une époque

II.5.4.2 CALCUL DES REPONSES ET RECHERCHE D'UN NEURONE GAGNANT

La minimisation de la fonction d'énergie est obtenue par effet combiné du calcul des réponses et de la loi d'adaptation des poids synaptiques.

Un bon placement pour une cellule c_k minimise la contribution de cette cellule c'est à dire contribution de c_k

$$\sum_{l \in K} d_{kl} cm_{lk} \quad (\text{II.4})$$

La réponse d'un neurone doit avoir la forme suivante:

Réponse du neurone n_i au stimulus c_k est

$$\sum_{l \in K} w_{il} cm_{lk} \quad (\text{II.5})$$

Avec w_{il} inversement proportionnel à d_{il}

La réponse de chaque neurone s'obtient alors par produit scalaire entre le vecteur des poids synaptiques du neurone et le vecteur stimulus présenté en entrée.

Un neurone n_i peut donner la même réponse pour deux cellules différentes c_k et $c_{k'}$, donc il faut déterminer laquelle de ces deux cellules occupera ce neurone. Pour se faire, on mettra ces deux cellules en compétition. La cellule gagnante est déterminée en comparant la réponse du neurone n_i au stimuli c_k et $c_{k'}$, abstraction faite des interconnexions entre ces deux cellules.

La cellule qui donne la plus forte réponse est placée en n_i , la perdante est enlevée du réseau, et la matrice des poids synaptiques est réinitialisée à rnd_val .

ALGORITHME

Calcul de la réponse de chaque neurone:

For $i = 1$ to N **Do**

$$\text{rik} = \sum_{l \in K} w_{ik} cm_{lk}$$

End_For

La file des neurones est ordonnée par ordre décroissant de la réponse de chaque neurone i à la cellule k . Dans le cas de plusieurs neurones ayant la même réponse pour la cellule k , on prend d'abord le neurone occupé par la cellule k , puis les neurones libres, et enfin les neurones occupés par les autres cellules, pour éviter que la cellule ne change inutilement de position qu'elle occupe ou de pourchasser des cellules, alors qu'il y a une position libre ayant une réponse égale.

II.5.4.3 ADAPTATION DES POIDS SYNAPTIQUES

Les neurones proches du neurone gagnant, auront une valeur de poids supérieur à celle des neurones éloignés de celui-ci, et ce après l'adaptation.

Après chaque détermination du neurone gagnant, le vecteur des poids est mis à jour de manière à augmenter la réponse du neurone gagnant au stimulus, ainsi que celle de ses voisins.

L'adaptation des poids se fait comme suit :

ALGORITHME

Begin

i : ensemble des neurones

For $i=1$ to N **Do**

If $i \neq \gamma$ **Then**

$w_{ik} = w_{ik} + \text{cost } \Gamma_{i\gamma} (\max(c_{mk})/d_{i\gamma} - w_{ik});$ /* i n'est pas un neurone gagnant */

If $i = \gamma$ **then** /* i est un neurone gagnant */

$w_{ik} = w_{ik} + \text{cost } \Gamma_{i\gamma} \text{nb_p/nmb_cel} (\max(CM_k) - w_{ik});$

End.

Nb_p : Nombre de cellules placées.

Cost : Gain statique (paramètre de faible valeur de l'ordre de 0.001).

Γ : Fonction de voisinage topologique.

1, si $\text{dis}(n_i, n_j) \leq \text{vois_min}$

Γ_{ij} : {

0, si $\text{dis}(n_i, n_j) > \text{vois_min}$

vois_min : voisinage minimum d'un neurone n_i auquel est associé la cellule c_k comme étant le plus petit voisinage pouvant contenir toutes les cellules connectées à la cellule k , il est égal en général à 2.

II.6.CONDITIONS D'ARRET

La boucle itérative s'arrête dès que le placement se stabilise, après avoir essayer de placer chaque cellule au moins une fois. Cette situation est atteinte avant que la convergence de la matrice des poids ne soit totale. Dès qu'une position fixe est attribuée à chaque cellule, la solution du problème du placement est trouvée. L'effet des itérations ultérieures serait de modifier le niveau d'attraction d'une cellule sur ses voisines.

II.7 RESULTATS ET TESTS

Considérons un exemple de 9 cellules illustrées par la figure II.4, et voyons le comportement de l'algorithme avec cet exemple.

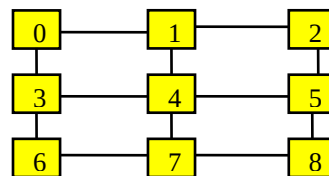


Figure II.4 Circuit à neuf cellules

Les interconnexions entre cellules sont données par la matrice de connexions suivante (tableau 2) :

	0	1	2	3	4	5	6	7	8
0	0	1	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0
2	0	1	0	0	0	1	0	0	0
3	1	0	0	0	1	0	1	0	0
4	0	1	0	1	0	1	0	1	0
5	0	0	1	0	1	0	0	0	1
6	0	0	0	1	0	0	0	1	0
7	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	1	0	1	0

Tableau II.2: Matrice de connexion pour le circuit de la fig. II.4

La solution optimale est connue et correspond au schéma de la figure II.4 La matrice de connexions après le prétraitement devient (tableau 3).

0.0	1.0	0.5	1.0	0.5	0.0	0.5	0.0	0.0
1.0	0.0	1.0	0.5	1.0	0.5	0.0	0.5	0.0
0.5	1.0	0.0	0.0	0.5	1.0	0.0	0.0	0.5
1.0	0.5	0.0	0.0	1.0	0.5	1.0	0.5	0.0
0.5	1.0	0.5	1.0	0.0	1.0	0.5	1.0	0.5
0.0	0.5	1.0	0.5	1.0	0.0	0.0	0.5	1.0
0.5	0.0	0.0	1.0	0.5	0.0	0.0	1.0	0.5
0.0	0.5	0.0	0.5	1.0	0.5	1.0	0.0	1.0
0.0	0.0	0.5	0.0	0.5	1.0	0.5	1.0	0.0

Tableau II.3 Matrice de connexion après le prétraitement.

Quant à la séquence d'entrée des cellules elle prendra la forme suivante :

séquence d'entrée = { **4** , 1 , 3 , 5 , 7 , 0 , 2 , 6 , 8 ,
 2 , 1 , 5 , 0 , 4 , 8 ,
 1 , 0 , 2 , 4 , 3 , 5 , 7 ,
 7 , 4 , 6 , 8 , 1 , 3 , 5 ,
 0 , 1 , 3 , 2 , 4 , 6 ,
 5 , 2 , 4 , 8 , 1 , 3 , 7 ,
 6 , 3 , 7 , 0 , 4 , 8 ,
 3 , 0 , 4 , 6 , 1 , 5 , 7 ,
 8 , 5 , 7 , 2 , 4 , 6 . }

Nous allons présenter maintenant les différentes itérations effectuées pour arriver à une solution optimale. Les neurones libres seront représentés par un trait (-), les différentes cellules par leur numéro, les chiffres représentés en gras indiquent le stimulus présenté en entrée (figure II.5).

N°	Placement	N°	Placement	N°	Placement	N°	Placement
1	4 -	10	4 1 2 - - - 3 0 5 - - - 7 -	19	4 5 2 - - - 3 8 1 - - - 6 7 0 - - - - - - - - - - - - - - - - - -	28	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
2	4 1 -	11	4 1 2 - - - 3 0 5 - - - 6 -	20	- 5 2 - - - 3 4 1 - - - 6 7 0 - - - - - - - - - - - - - - - - - -	29	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
3	4 1 - - - - 3 -	12	4 1 2 - - - 3 0 5 - - - 6 7 - - - - - - - - - - - - - - - - - - -	21	- 5 2 - - - 3 4 1 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	30	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
4	4 1 - - - - 3 5 - - - - - - - - - - - - - - - - - - -	13	4 1 2 - - - 3 8 5 - - - 6 7 - - - - - - - - - - - - - - - - - - -	22	0 5 2 - - - 3 4 1 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	31	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
5	4 1 - - - - 3 7 - - - - - - - - - - - - - - - - - - -	14	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - -	23	0 5 2 - - - 3 4 1 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	32	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
6	4 1 5 - - - 3 7 - - - - - - - - - - - - - - - - - - -	15	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - -	24	0 1 2 - - - 3 4 - - - - 6 7 8 - - - - - - - - - - - - - - - - - -	33	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
7	4 1 5 - - - 3 0 - - - - - - - - - - - - - - - - - - -	16	4 1 2 - - - 3 8 5 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - -	25	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	34	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
8	4 1 5 - - - 3 0 - - - - 7 - - - - - - - - - - - - - - -	17	4 5 2 - - - 3 8 - - - - 6 7 - - - - 0 - - - - - - - - - - - - - - -	26	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	35	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -
9	4 1 2 - - - 3 0 - - - - 7 - - - - - - - - - - - - - - -	18	4 5 2 - - - 3 8 1 - - - 6 7 - - - - 0 - - - - - - - - - - - - - - -	27	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -	36	0 1 2 - - - 3 4 5 - - - 6 7 8 - - - - - - - - - - - - - - - - - -

Figure II.5 Evolution des itérations pour arriver à une solution optimale

COMMENTAIRES

La première cellule présentée au réseau de neurones est la cellule 4, nous l'avons forcé à se positionner au neurone N°1. Ensuite, c'est les autres cellules qui se présentent à leur tour. Le premier pourchassement se produit à l'itération N°5, lorsque la cellule 7 pourchasse la cellule 5. D'autres cellules seront présentées, et d'autres pourchassements vont se produire jusqu'à l'itération N°25, où on constate qu'on est arrivé à une solution optimale. La boucle itérative est terminée quand chacune des cellules est présentée sans qu'un changement ne se produise dans le placement.

Nous allons effectuer une série de tests sur un circuit, pour évaluer les performances de notre algorithme et en tirer des conclusions. Nous allons évaluer les différents résultats en fonction de la taille du réseau de neurones, de la première cellule présentée, de sa position et des différents paramètres utilisés dans l'algorithme.

Tout d'abord, un circuit de 25 cellules aura pour solution optimale le circuit donné par la figure II.6.

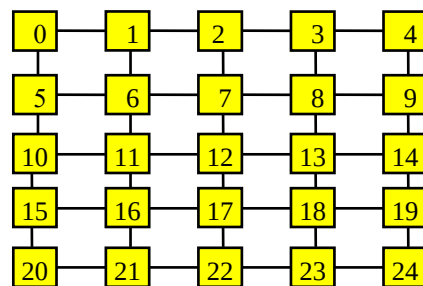


Figure II.6 solution optimale pour un problème à 25 cellules.

II.7.1 INFLUENCE DE LA TAILLE DU RESEAU

On a effectué une série de tests avec deux réseaux différents, le premier de même taille que le problème, c'est à dire 5 lignes sur 5 colonnes et l'autre 4 fois plus grand que la taille du circuit c'est à dire 10 lignes sur 10 colonnes.

En gardant la même séquence d'entrée et la même position initiale de la cellule en entrée, qui sont toutes les deux déterminées aléatoirement, on est arrivé à

- Un placement optimal sur 10 essais pour le premier réseau.
- Deux placements optimaux sur 10 essais pour le deuxième réseau.

COMMENTAIRES

On constate que le fait d'augmenter la taille du réseau à elle seule n'a pas vraiment donné des résultats encourageants.

II.7.2 INFLUENCE DU CHOIX DE LA POSITION DE LA PREMIERE CELLULE

Dans cette série de tests, on prend toujours les mêmes réseaux que précédemment, mais cette fois on fixe la séquence d'entrée, c'est à dire qu'on va effectuer ces tests sur une même séquence d'entrée. Quant à la position de la première cellule, elle sera déterminée aléatoirement comme lors de la première série de tests. Donc on a obtenu les résultats suivants:

- Un placement optimal sur 10 essais pour le premier réseau
- Trois placements optimaux sur 10 essais pour le deuxième réseau.

II.7.3 INFLUENCE DU CHOIX DE LA SEQUENCE D'ENTREE

On prend toujours les mêmes réseaux, la séquence d'entrée sera déterminée aléatoirement, et on va positionner la première cellule au centre du réseau, quelque soit la séquence d'entrée. Les résultats des tests qu'on a effectués sont les suivants:

- 0 placement optimal sur 10 essais pour le premier réseau.
- 8 placements optimaux sur 10 essais pour le second réseau.

II.8 ANALYSE ET ADAPTATIONS DE L'ALGORITHME

Dans cette partie, on va essayer d'analyser cet algorithme grâce à l'étude de cas pratiques, et de l'améliorer en fonction des résultats obtenus. En effet d'après les résultats obtenus avec l'algorithme présenté, on a constaté par exemple que la taille du réseau, le choix de la première cellule et sa position dans le réseau influent énormément sur la qualité du placement. On a rencontré aussi, après plusieurs tests effectués, des problèmes que nous allons détailler dans ce qui suit et que nous tenterons de résoudre pour améliorer cet algorithme.

II.8.1 LIMITES GEOMETRIQUES DU RESEAU DE NEURONES

On remarque que l'algorithme procède de la manière suivante: un début de solution, c'est à dire un sous-ensemble de cellules correctement placées les unes par rapport aux autres, est amorcé puis complété jusqu'à l'obtention de la solution complète. Si la solution s'amorce trop près d'un des bords du réseau, puis elle est complétée en direction de ce bord, il ne restera pas assez de place pour terminer le placement des autres cellules. La figure II.7 démontre bien le problème des limites géométriques.

-	98	39	19	79	59	69	49	29	89	9
-	95	0	1	2	3	4	5	6	7	8
-	93	10	11	12	13	14	15	16	17	18
-	96	20	21	22	23	24	25	26	27	28
-	94	30	31	32	33	34	35	36	37	38
-	99	40	41	42	43	44	45	46	47	48
-	92	50	51	52	53	54	55	56	57	58
-	97	60	61	62	63	64	65	66	67	68
-	91	70	71	72	73	74	75	76	77	78
-	90	80	81	82	83	84	85	86	87	88
-	-	-	-	-	-	-	-	-	-	-

Figure II.7 Limites géométriques du réseau de neurones.

II.8.2 CROISSANCE DES SOLUTIONS

Malgré que le problème des limites géométriques peut être évité en augmentant suffisamment la taille du réseau, la qualité des solutions reste faible à cause d'une autre difficulté: Lorsqu'une solution s'amorce en un point du réseau de neurones, le sous-ensemble de cellules correctement placées va croître selon deux axes particuliers pour donner par exemple une solution telle qu'elle est représentée dans la figure II.8

-	-	-	-	-	-	-	-	-	-
-	-	50	40	30	20	10	0	-	-
-	-	-	41	31	21	11	1	-	-
-	-	-	42	32	22	12	2	-	-
-	-	-	-	33	23	13	3	-	-
-	-	-	-	-	24	14	4	-	-
-	-	-	-	-	-	-	5	-	-
-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-

Figure II.8 Croissance d'une solution

Donc lorsque deux solutions partielles s'amorcent à deux endroits différents c'est à dire au mauvais endroit du réseau, soit ces deux solutions entrent en collision, soit elles sont assez

éloignées l'une de l'autre, et la solution totale est disjointe (figure II.9). Pour remédier à ce problème, on a recours au principe du recuit simulé.

-	-	9	19	29	39	49	59	69	93	83	-	-
-	-	8	18	28	38	48	58	68	94	84	-	-
-	-	7	17	27	37	47	57	67	95	85	-	-
-	-	6	16	26	25	24	23	22	21	-	-	-
-	-	5	15	36	35	34	33	32	31	-	-	-
-	-	4	14	46	45	44	43	42	41	40	-	-
-	-	3	13	56	55	54	53	52	51	50	-	-
-	-	2	12	66	65	64	63	62	61	60	-	-
-	-	1	11	76	75	74	73	72	71	70	-	-
-	-	0	10	20	77	98	99	87	86	80	-	-
-	-	-	-	30	78	88	89	97	96	90	-	-
-	-	-	-	-	79	81	82	91	92	-	-	-

Figure II.9 Croissance et collision de plusieurs solutions partielles

II.9 AMELIORATION DE L'ALGORITHME PAR RECUIR SIMULE

Lorsqu'un système est à l'équilibre, l'expression de la probabilité que ce système soit dans un état particulier comporte un facteur de la forme $e^{-E/KT}$, où E est l'énergie totale du système, K la constante de BOLTZMANN, et T la température [16].

Lorsqu'on abaisse la température d'un système constitué d'un ensemble de particules, ce système tend à se retrouver dans un état d'énergie minimale. Or le fait d'abaisser la température d'un système n'est en général pas suffisant pour que l'état d'énergie minimal soit atteint.

On va utiliser le concept du recuit simulé lors de la conception d'un circuit VLSI. Les problèmes d'optimisation peuvent souvent être formulés sous une forme analogue à ceux de la recherche d'un état d'énergie minimale.

L'idée de base de l'algorithme du recuit simulé consiste à simuler le comportement de la nature qui tend naturellement à se réarranger pour que soit minimisée une certaine fonction objective, le niveau d'énergie.

II.9.1 ALGORITHME

Ces deux forces sont approximativement de même intensité, donc la cellule 28 ne va pas changer de place. Avec l'introduction du prétraitement complet, c'est toutes les autres cellules qui vont influencer la cellule 28, et puisque leurs forces sont nettement plus grandes que celles des cellules 29, 33, 34 et 35, la cellule 28 va changer de place. Cette amélioration a donné des résultats très satisfaisants.

II.10 AUTRES AMELIORATIONS

II.10.1 CHOIX DE LA TAILLE DU RESEAU

Pour éviter le problème des limites géométriques du réseau de neurones, on prend un réseau suffisamment grand, c'est à dire, en générale quatre fois plus grand que la taille du problème.

II.10.2 CHOIX DE LA PREMIERE CELLULE ET DE SA POSITION

La première cellule présentée au réseau pour être placée sera choisie comme étant la cellule qui a une connexion avec le plus grand nombre de cellules possibles et elle sera placée dans le neurone centre du réseau.

II.10.3 INITIALISATION DE LA MATRICE DES POIDS SYNAPTIQUES

L'initialisation de la matrice des poids synaptiques par la valeur `rnd_val` est supprimée.

II.11 TESTS ET RESULTATS

Nous allons tester notre algorithme amélioré et adapté sur plusieurs problèmes, pour évaluer ses performances. On va effectuer ces tests sur des problèmes de 25 cellules (figure II.6), 36 cellules (figure II.11) et enfin sur un problème de 100 cellules analogue à celui de 36 cellules. Ensuite, nous effectuerons une série de tests sur un problème qui a une forme d'hypercube (figure II.12), et un circuit asymétrique (figure II.13)

La taille du réseau sera prise comme convenu quatre fois plus grand que la taille du problème

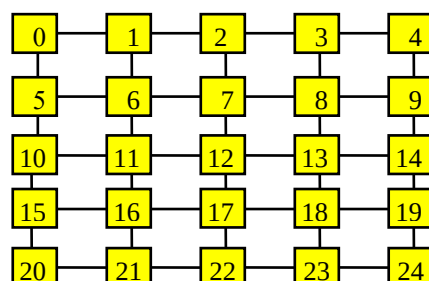


Figure II.11 Solution optimale pour un problème de 36 cellules

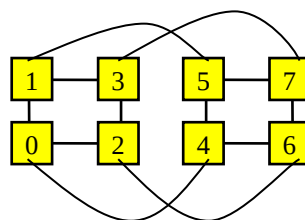
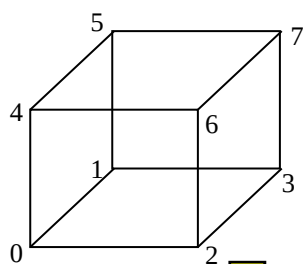


Figure II.12 Hypercube de dimension 3

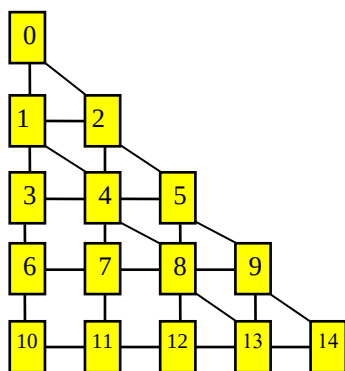


Figure II.13 Problème triangulaire

Nombre de cellules	Première cellule en entrée	Placement	Nombre d'itérations
25	12	Neurone centre	54
36	14	Neurone centre	115
100	44	Neurone centre	211
Hyper cube (8)	2	Neurone centre	34
Asymétrique (15)	5	Neurone centre	113

Tableau II.3 Résultats des tests

Pour le problème de 25 cellules, la première cellule en entrée est la cellule 12, elle a été placée dans le neurone centre du réseau, et la solution optimale est obtenue après 54 itérations.

Pour le problème de 36 cellules, la première cellule en entrée est la cellule 14, elle a été placée dans le neurone centre du réseau, et la solution optimale est obtenue après 115 itérations.

Pour le problème de 100 cellules, la première cellule en entrée est la cellule 44, elle a été placée dans le neurone centre du réseau, et la solution optimale est obtenue après 211 itérations.

Pour le problème d'hypercube, on a utilisé un circuit de 8 cellules, la première cellule en entrée est la cellule 2, elle a été placée dans le neurone centre du réseau, et la solution optimale est obtenue après 34 itérations

Pour le problème asymétrique, on a utilisé un circuit de 15 cellules, la première cellule en entrée est la cellule 5, elle a été placée dans le neurone centre du réseau, et la solution optimale est obtenue, en utilisant un nombre réduit de connexions implicites pour le prétraitement, après 113 itérations.

II.12 STRUCTURE DES DONNEES

Une cellule est un enregistrement à deux champs, le premier champ indique le nombre de connexions réelles de la cellule, et le deuxième est un pointeur vers réel qui constitue son vecteur de connexion.

```
# typedef struct cellule
```

```
int nmb_cel;
```

```
float *ligne_con;
```

Un neurone est constitué de quatre champs, son numéro, son état représente par le numéro de cellule qu'elle l'occupe, sa réponse pour le stimulus en entrée et enfin, le quatrième champs, un pointeur vers un autre neurone.

```
# typedef struct neurone
```

```
int numéro;
```

```
int état;
```

```
float réponse;
```

```
neurone *neur_prochain;
```

La matrice de connexions est un tableau de cellules : cellule mc_c[N]; le réseau de neurone est une liste chaînée de la structure neurone: neurone *réseau, la matrice des poids synaptiques est un pointeur vers réel: float * n_poids

La taille de ces trois structures est proportionnelle à la taille du problème (le nombre de cellules à placer) et la taille du réseau (nombres de neurones) utilisés pour le placement.

Soit nmb_cel le nombre total des cellules, $li_réseau$ et $cl_réseau$ le nombre de lignes et de colonnes du réseau de neurones. On a besoin d'allouer $(nmb_cel * nmb_cel)$ réels pour la matrice de connexions, et $(nmb_cel * li_réseau * cl_réseau)$ réels pour la matrice des poids synaptiques, et $(li_réseau * cl_réseau)$ neurones pour le réseau.

Etant donné que la matrice de connexions est symétrique et de diagonale nulle. Cela nous a permis d'optimiser sa demande d'allocation. Pour chaque cellule i , au lieu d'allouer un tableau de nmb_cel réels, on alloue seulement $(nmb_cel - i - 1)$ réels, qui indique sa valeur de connexion avec les cellules suivantes ($i+1, i+2, \dots, nmb_cel$), sa valeur de connexion avec les cellules précédentes ($0, 1, \dots, i-1$) se trouve dans leurs lignes de connexions

Au terme de l'étude du cas fondamental, une première conclusion peut-être tirée, l'approche neuronale est applicable pour le placement de cellules V.L.S.I. La propriété de classification des réseaux de KOHONEN est en effet adaptée à ce type de problèmes.

Les résultats obtenus sont maintenant acceptables, quoique le temps de calcul soit trop important pour des problèmes de grandes tailles (tableau II.4)

Nombre de cellules	Taille du réseau	Nombre d'itérations	Temps de calcul
100	20 x 20	211	1' 15"
144	24 x 24	302	4' 9"
256	32 x 32	532	38' 30"
289	34 x 34	600	57' 35"

Tableau II.4 Le temps de calcul pour les différents circuits

II.13 EXTRAPOLATION DE L'ALGORITHME

II.13.1 LES CONNEXIONS ENTRE CELLULES

Dans un circuit réel, les cellules sont alignées les unes à côtés des autres, pour former des bandes ininterrompues. Ces bandes sont placées les unes au dessus des autres, et elles sont séparées par un canal laissé libre pour les rails de connexions (figure II.14).

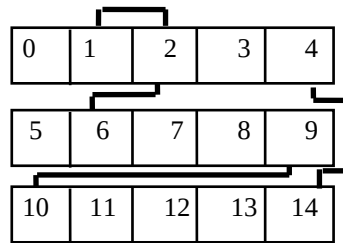


Figure II.14 Cellules par bandes.

Seul les côtés supérieurs et inférieurs de la cellule sont accessibles aux connexions.

II.13.2 LES CELLULES DE DIFFERENTES TAILLES

Une cellule est une entité fonctionnelle, elle peut aller d'une simple porte logique jusqu'à une unité de multiplication. Plus la fonction réalisée est complexe, plus elle nécessite une grande surface. C'est pourquoi, dans un circuit, on peut trouver de grandes et de petites cellules.

Les petites et moyennes cellules sont regroupées en **bibliothèque**. Dans ce cas, le rapport de surface entre la plus petite et la plus grande cellule d'une même bibliothèque reste limité à un ordre de grandeur. Etant donné que les cellules sont placées par bandes, chaque bibliothèque est souvent dessinée de manière à ce que la hauteur de toutes ses cellules soit identique. Seule la largeur des cellules varie d'une cellule à l'autre. Les très grosses cellules sont dessinées indépendamment de toute bibliothèque. Elles sont optimisées séparément, elles sont généralement désignées sous le terme de macro-cellules.

II.13.3 LE BUT RECHERCHE

Le but de l'algorithme est de minimiser la longueur de connexion et par conséquent la surface du circuit.

La surface des bandes est une constante égale à la largeur des bandes, quelque soit le placement, et la largeur des zones de connexions entre bandes.

Elle est donc indépendante du placement. Seule la hauteur des zones de connexions est influencée par le placement.

Dans un circuit les connexions sont réalisées en utilisant deux couches métalliques. L'une aux lignes horizontales, l'autre, aux lignes verticales. La hauteur de la zone qui sépare deux bandes est déterminée uniquement par le nombre maximum des lignes horizontales qui passent au même endroit.

Deux méthodes permettent de minimiser cette hauteur:

- Minimiser la largeur des connexions pour limiter ce nombre maximum.
- Favoriser les connexions qui n'utilisent pas de lignes horizontales.

II.13.4 MODELISATION DU PROBLEME

II.13.4.1 MODELISATION NEURONALE

Pour des cellules de tailles différentes, on leur attribue un nombre différent de neurones, chaque cellule est représentée sur le réseau par un ensemble de neurones.

Le plan neuronal sera divisé en une superposition de bandes de neurones (figure II.15).

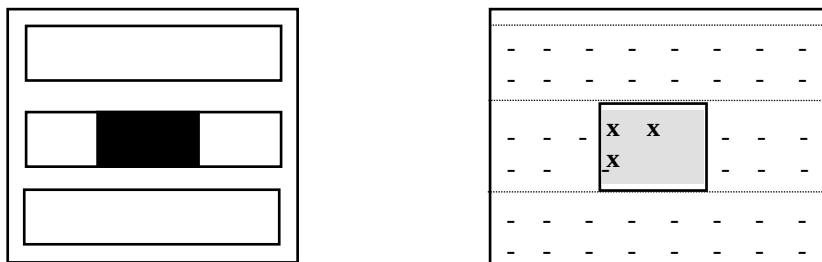


Figure II.15 Modélisation d'une cellule dans une bande.

Chaque cellule qui occupe une portion d'une bande dans un circuit, voit l'ensemble des neurones qu'elle recouvre, occuper une portion de la bande de neurones qui représente la bande à laquelle cette cellule appartient.

La structure par bandes introduit une contrainte quant aux positions que peut occuper une cellule dans le circuit.

Seuls les emplacements situés dans une bande sont acceptés :

- Nous n'autorisons pas de mélange de bibliothèque. Toutes les cellules ont la même hauteur, propre à la bibliothèque d'où elles sont issues. Toutes les cellules sont rectangulaires. Elles disposent d'accès pour les connexions sur leur partie supérieure et inférieure.

- Nous n'acceptons pas de macro-cellules.

II.13.4.2 MODELISATION DES CELLULES

Dans l'extrapolation de l'algorithme, on va introduire la notion de **borne**. Chaque cellule est divisée en un certain nombre de bornes de connexion, il existe autant de bornes que de neurones occupés par la cellule. Comme le nombre de bornes est déterminé par la taille de la cellule, et que le nombre d'accès est déterminé par sa fonction, il peut y avoir plus ou moins de bornes que d'accès, donc une borne peut représenter plusieurs accès, certaines peuvent n'en représenter aucune.

Ces bornes de connexion représentent toutes les entrées ou les sorties possibles de la cellule. Si toutes les cellules sont dessinées avec des entrées ou des sorties ne se trouvant que d'un seul côté, il est indispensable de spécifier ce côté par lequel une connexion doit rejoindre la cellule. Il faut dans ce cas pouvoir distinguer si une liaison a un accès du haut ou du bas de la cellule, donc deux bornes seront nécessaires pour discrétiser la hauteur de la cellule.

Les cellules occupent sur le réseau des positions dans des bandes de deux neurones de haut (figure II.16), donc la hauteur d'une cellule sera limitée à deux bornes.

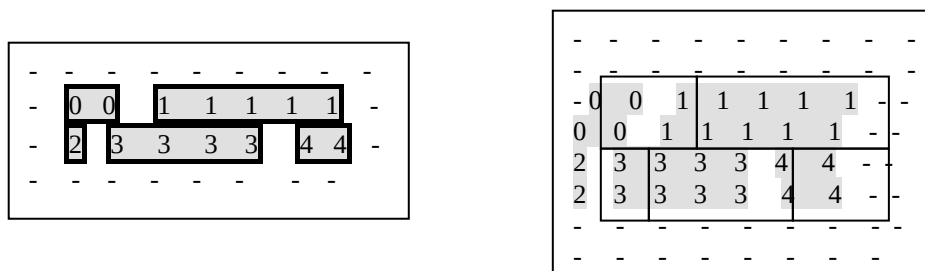


Figure II.16 Bande de neurones à un neurone (a) et deux neurones de haut (b)

II.14 ADAPTATION DE L'ALGORITHME

L'introduction des bornes nous amène à modifier l'algorithme, d'une part, on va travailler avec les cellules pour la séquence d'entrée par exemple, et d'autre part, on travaillera avec les bornes comme pour le placement.

II.14.1 ENONCE DU PROBLEME

On a un ensemble de cellules, chaque cellule est constituée d'un certain nombre de bornes, qui sont reliées entre elles, et ces liens sont exprimés dans la matrice de connexion entre bornes. On demande de placer sur le réseau un ensemble de bornes conformément à la topologie des cellules et des bornes, tout en minimisant la somme des longueurs de connexions. D'une manière plus formelle, le problème peut s'énoncer de la manière suivante:

Etant donné:

$K = \{0, \dots, C-1\}$, un ensemble de cellules c_k

$X = \{0, \dots, B-1\}$, un ensemble de bornes b_x

$I = \{0, \dots, N-1\}$, un ensemble de positions n_i

Une matrice symétrique de connexions entre bornes :

$CM_B = cm_b_{ab} \quad a, b \in X$

Donc on peut déduire

$K = \{0, \dots, C-1\}$, un ensemble de cellules c_k

$X_k = \{\text{borne de la cellule } k\}$

Donc :

$$X = \bigcup_{k \in K} X_k \quad (II.5)$$

On demande :

Un ensemble D de couples $(x, i) \quad \forall x \in X, i \in I$,

Tel que :

$$\sum_{k, l \in K} \sum_{x \in X_k} \sum_{y \in X_l} d_{xy} cm_b_{xy} \quad (II.6)$$

Soit minimum Avec $\forall x, y \in X, i, j \in I : d_{xy} = d_{ij} \quad \text{ssi} \quad (x, i) \in D \text{ et } (y, j) \in D$

II.14.2 PRETRAITEMENT

Le prétraitement est identique à celui utilisé dans le premier algorithme, mais avec une matrice de connexion entre les bornes et pas une matrice de connexion entre cellules.

II.14.3 SEQUENCE D'ENTREE

La séquence d'entrée est constituée de la même manière que précédemment. Toutes les cellules sont introduites dans la séquence d'entrée, et chacune d'elle est suivie des cellules qui y sont connectées, et ce par ordre décroissant de leur intensité de connexions.

La seule modification apportée ici, est la suivante : lorsque l'on introduit la totalité des cellules dans la séquence d'entrée, il est préférable de commencer par celles qui sont les plus interconnectées et de leur attribuer une position centrale et de terminer avec celles qui le sont le moins. Ceci leur permettra d'occuper cette position autour de laquelle les autres cellules viendront se situer.

Pour faciliter l'écriture et la compréhension, nous introduisons une matrice de connexion entre cellules, calculée à partir de la matrice de connexion entre bornes tel que :

$$cm_c[i, j] = \sum_{x \in X_i} \sum_{y \in X_j} cm_b_{xy} \quad (II.7)$$

La connexion entre la cellule i et j est égale à la somme des connexions des bornes de la cellule i avec les bornes la cellule j.

II.14.4 CALCUL DES REPONSES

Le but de l'algorithme est de minimiser une fonction d'énergie totale en minimisant la contribution de chaque cellule. La contribution de chaque cellule est la somme des contributions de chacune de ses bornes.

$$r_{ix} = \sum w_{iy} cm_b_{yx} \quad (II.8)$$

Puisque la réponse du réseau à une cellule concerne au moins deux neurones, un pour la partie supérieure l'autre pour la partie inférieure. Il n'existe pas un seul neurone gagnant mais un ensemble de neurones gagnants qui maximise la réponse d'une cellule.

Pour des raisons de commodité de langage, nous continuerons à considérer la réponse en prenant comme référence le neurone sur lequel est placée la borne supérieure gauche de la cellule. Tous les termes, comme celui de "neurone gagnant", utilisés jusqu'à présent, peuvent encore être utilisés.

A chaque borne est associée le neurone sur lequel celle-ci est placée. L'ensemble des couples bornes/neurones est l'ensemble D : Ensemble de couples (x, i) $x \in X$ et $i \in I$ / I ensemble de neurones.

Cet ensemble peut être mis en bijection avec l'ensemble E des couples cellules/neurones (ensemble de couples (k, i) $k \in K$ et $i \in I$) où chaque cellule est associée avec le neurone sur lequel est placée la borne du coin supérieur gauche de la cellule.

Ces deux ensembles reflètent deux vues différentes de la même réalité. L'ensemble D exprime le placement sur le réseau de bornes, et l'ensemble E exprime le placement sur le réseau de cellules. Il est possible de déduire l'un de l'autre.

Notons qu'on ne peut donner de sens à la réponse d'un neurone que si celui-ci attribue une position admissible à la cellule. Si les lignes du réseau sont numérotées à partir de zéro, seules les lignes paires contiennent de tels neurones. En effet, les neurones des autres lignes situent la cellule sur deux bandes, ce qui n'est pas autorisé. Il en va de même pour les neurones à l'extrême droite du plan neuronal qui placent partiellement les cellules en dehors du réseau. Par convention, la réponse de ces neurones est zéro.

Dans l'algorithme proposé dans [4], le calcul de la réponse du réseau à une cellule s'effectue de la manière suivante :

$$rik = \sum_{x \in X_k} \sum_{y \in X} w_{iy} mc_{xy} \quad (II.9)$$

Donc la réponse du neurone i pour la cellule k est égale à la réponse du neurone i pour l'ensemble des bornes de la cellule k.

Après avoir effectuer plusieurs tests avec cette méthode de calcul en utilisant l'adaptation appropriée, on n'est pas arrivé à un bon placement pour la raison suivante : la méthode ne prend en compte qu'un seul neurone alors que la cellule se place sur plusieurs neurones, et elle présente aussi l'inconvénient de ne pas utiliser en totalité les nouvelles valeurs trouvées après adaptations pour les neurones voisins du neurone gagnant. Donc on a proposé une deuxième méthode de calcul basée sur la première méthode, mais elle prend en considération la totalité des neurones qui nous placent la cellule :

$$rik = \sum_{j \in D_k} r_{jk} \quad (\text{II.10})$$

D_k ensemble des neurones qui nous placent la cellule k. avec

$$r_{jk} = \sum_{x \in X_k} \sum_{y \in X} w_{jy} mc_{-b_{xy}} \quad (\text{II.11})$$

Cette méthode a donné de meilleurs résultats, mais pas les résultats voulus. Certes cette méthode considère la place occupée par la cellule, mais elle ne prend pas en compte la position de la borne, car la réponse d'un neurone de l'ensemble de neurones D_k est calculée pour l'ensemble de bornes X_k .

Le fait que le neurone ne peut nous placer qu'une seule borne, cela nous a conduit à une troisième méthode qui concrétise cet état de chose

$$r_{ik} = \sum_{j \in D_k} \sum_{x \in X_k} r_{jx} \quad (\text{II.12})$$

Tel que j nous place la borne x de la cellule k avec

$$r_{jx} = \sum_{y \in X_k} w_{jy} mc_{-b_{xy}} \quad (\text{II.13})$$

II.14.5 RECHERCHE DU NEURONE GAGNANT

Lorsqu'on dispose d'une cellule sur le réseau, trois cas sont possibles:

- La place est vacante, dans ce cas la cellule se place là où la réponse est la plus forte.
- L'espace est occupé par la cellule elle même, il n'y a pas de changement de position.
- La place est occupée par une ou plusieurs cellules, il y a alors comme précédemment compétition entre les différentes prétendantes à cette place. Le conflit se règle en trois étapes:
 1. Calcul de la réponse de la cellule qu'on cherche à placer.
 2. Calcul de la somme des réponses de l'ensemble des cellules qui occupent la place convoitée.

3. La place sera attribuée à l'ensemble des cellules ayant la réponse la plus forte.

Puisque la place d'une cellule ne concerne pas un seul neurone, nous avons mis une fonction "espace" qui évalue le degré d'occupation de cette place et cela comme suit :

- Si le neurone supérieur gauche est occupé par la cellule en entrée la fonction espace retourne zéro, il en va de même si l'ensemble des neurones est libre de la place indiquée.
- Si la place est occupée par une ou plusieurs cellules, on prend le nombre de bornes supérieures de chaque cellule c'est à dire l'ensemble des bornes divisé par deux, et on le multiplie par 0.5, on lui ajoute ensuite 0.25 si une de ces cellules occupe une partie de cette place à partir de neurone supérieur gauche de la place.

Cette fonction nous a permis de minimiser le degré de déformation du circuit au cours du placement, parce qu'entre deux places ayant la même réponse et d'un degré d'occupation supérieur à 0 c'est à dire toute les deux sont occupées, de mettre la place ayant la plus petite valeur d'occupation en priorité, et par conséquent de mettre en premier, la cellule en entrée, en compétition avec le plus petit nombre de cellule de petite taille. Précédemment, une cellule chassée du réseau était immédiatement placée à l'itération suivante. En conservant le même principe, la ou les cellule(s) chassée(s) sont les premières à être présentées en entrée. Pour cela une pile LIFO est utilisée. La dernière cellule enlevée est la première à être proposée en entrée.

II.14.6 ADAPTATION DES POIDS SYNAPTIQUES

II.14.6.1 NOUVELLE DISTANCE

La structure par bandes (figure II.17) de ce nouvel algorithme et la présence ininterrompue de bandes des cellules, peut obliger les connexions à contourner certaines bandes en passant par les rails de connexions. Ceci allonge considérablement la longueur de ces connexions. L'évaluation de la distance entre deux neurones doit refléter cet état de chose.

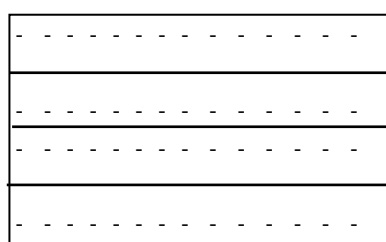


Figure II.17 Réseau neuronal découpé en bandes.

La distance entre deux neurones du plan neuronal dépend du circuit qu'occupe le réseau. Nous allons prendre comme exemple le circuit de la figure (figure II.18) où une borne **x** est placée, et nous allons évaluer la distance entre tous les neurones du plan et le neurone occupé par cette borne.

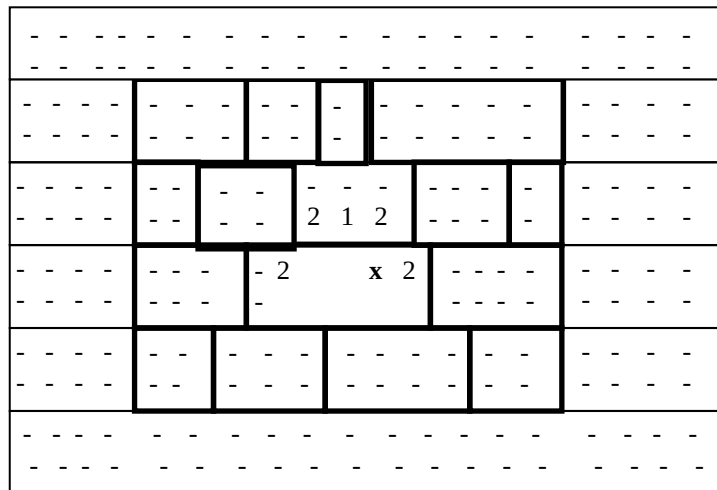


Figure II.18 Circuit placé sur le réseau avec les neurones voisins de X .

La figure II.19 indique les neurones les plus proches de X. Alors, des huit neurones proches de X, il y a :

- Un seul neurone à une distance à 1. A cet endroit, les connexions verticales ne contribuent pas à augmenter l'espace entre les bandes. Elle est meilleure, et ceci se traduit par une distance plus petite.
- Quatre neurones à une distance à 2, ces quatre positions nécessitent une ligne horizontale pour établir la connexion, la position est moins bonne et la distance est plus grande.
- Les trois neurones a, b, c, quoique proches graphiquement, en sont en fait éloignés. En effet, pour relier une borne qui les occuperait à la borne X, il faut contourner la bande de X.

Avec cette évaluation, l'algorithme va favoriser les connexions qui n'utilisent pas de lignes horizontales, et ne contournent pas les bandes, ainsi les deux manières de réduire la surface du circuit sont prises en compte.

Les neurones occupés par des bornes positionnées de part et d'autre de la zone de connexion qui donne l'accès à la borne X, sont situés à une distance croissante en s'écartant de X, comme l'indique la figure II.19.

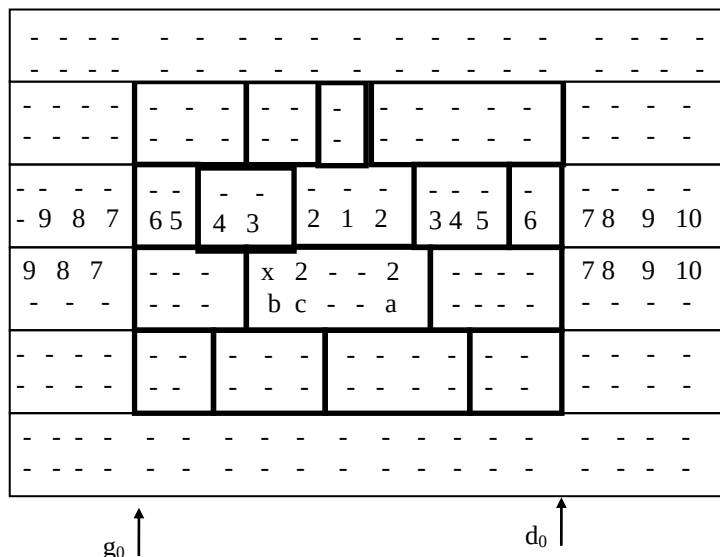


Figure II.19 Distance des neurones de part et d'autre de la même zone de connexion.

Arrivé en g_0 et d_0 , les limites du circuit, la connexion éventuelle peut atteindre les zones de connexion en utilisant le rail de connexion. La distance augmente donc en fonction du parcours vertical (figure II.20).

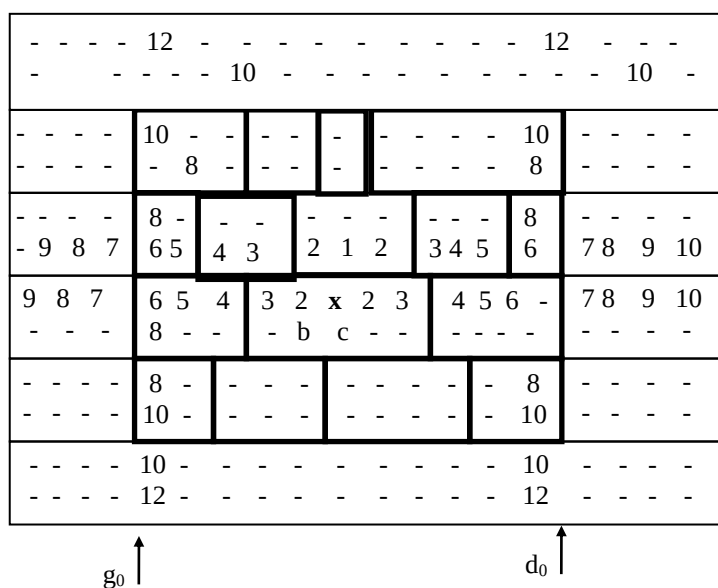


Figure II.20 La distance augmente en fonction du parcours vertical.

Pour les neurones occupés par les bornes positionnées de part et d'autre des autres zones de connexions et celles qui donnent accès à la borne X, la distance croît régulièrement en partant des rails de connexions vers l'intérieur du circuit. Quand plusieurs chemins sont possibles, seul le plus court est pris en considération (figure II.21).

-	-	-	12	-	-	-	-	-	-	-	12	-	-	-	-
-	-	-	10	-	-	-	-	-	-	-	10	-	-	-	-
-	-	-	-	10	11	12	13	14	15	14	13	12	11	10	-
-	-	-	-	8	9	10	11	12	13	12	11	10	9	8	-
10	9	8	7	8	-	-	-	-	-	-	-	8	α	-	-
-	-	-	-	6	5	4	3	2	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	x	2	3	4	5	6	7
-	-	-	-	8	9	10	11	12	13	12	11	-	-	-	-
-	-	-	-	8	9	10	11	12	13	12	11	10	9	8	-
-	-	-	-	10	11	12	13	14	15	14	13	12	11	10	-
-	-	-	-	10	11	12	13	14	15	14	13	12	11	10	-
-	-	-	-	10	-	-	-	-	-	-	-	10	-	-	-
-	-	-	-	12	-	-	-	-	-	-	-	12	-	-	-

$g_0 \uparrow$
 $d_0 \uparrow$

Figure II.21 La distance croît en partant des rails de connexion vers l'intérieur du circuit

La distance des autres neurones du neurone X n'est pas déterminée. En effet il est impossible de prévoir la distance du neurone de la borne de référence à un de ces neurones sans connaître la largeur de la cellule qui sera placée sur ces neurones. Pour la déterminer, on considère que la distance augmente par pas d'une unité, lorsque l'on s'écarte du circuit en restant sur la même ligne du réseau. La valeur α est initialisée de la dernière valeur exacte à l'intérieur des limites du circuit ceci nous donne : $\alpha = 9$. Donc la distance de l'ensemble des neurones du neurone X est déterminée comme l'indique la figure II.22.

13	12	11	10	11	12	13	14	15	14	13	12	11	10	11	12	13	14
11	10	9	8	9	10	11	12	13	12	11	10	9	8	9	10	11	12
11	10	9	8	9	10	11	12	13	12	11	10	9	8	9	10	11	12
9	8	7	6	5	4	3	2	x	2	3	4	5	6	7	8	9	10
11	10	9	8	9	10	11	12	13	12	11	10	9	8	9	10	11	12
11	10	9	8	9	10	11	12	13	12	11	10	9	8	9	10	11	12
13	12	11	10	11	12	13	14	15	14	13	12	11	10	11	12	13	14
15	14	13	12	13	14	15	16	17	16	15	14	13	12	13	14	15	16

$g_0 \uparrow$
 $d_0 \uparrow$

Figure II.22 La distance des autres neurones.

II.14.6.2 ADAPTATION

L'adaptation tiendra compte de la position de chaque borne de la cellule qui vient d'être placée, et donc d'adapter la matrice des poids synaptiques en prenant chaque fois le couple (x, y) tel que $x \in X_k$, et y place la borne x de cellule k .

ALGORITHME

DEBUT

Pour Chaque borne de la cellule k **Faire**

Pour $i \leftarrow 1$ à N **Faire**

debut

$t \leftarrow t + 1$;

$T \leftarrow 1 / 1 + t$;

Si i place la borne x **Alors** /* $y = i$ */

$\text{Adapt} \leftarrow \Gamma_{iy}(\text{nb_p} / \text{nmb_cel}) (\max(\text{CM_B}_x) - w_{ik}(t - 1)$;

Sinon

$\text{Adapt} \leftarrow \Gamma_{iy} (\max(\text{CM_B}_x) / d_{iy} - w_{ix}(t - 1)$;

$\text{noise} \leftarrow \text{rnd_val} * T$;

$w_{ix}(t) \leftarrow w_{ix}(t-1) + \text{cost}[(1 - \alpha) \text{adapt} + \alpha \text{noise}]$;

fin

FIN

II.15 ADAPTATION

Nous allons effectuer une série de tests. Dans un premier temps, la taille des problèmes à résoudre est limitée à un petit nombre de cellules. Les figures (23, 24 et 25) donnent trois circuits qui ont été utilisés comme exemples.

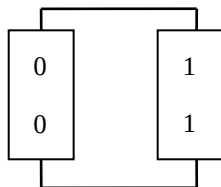


Figure II.23 Circuit de 2 cellules et 4 bornes

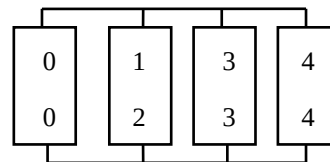


Figure II.24 Circuit de 4 cellules et 8 bornes

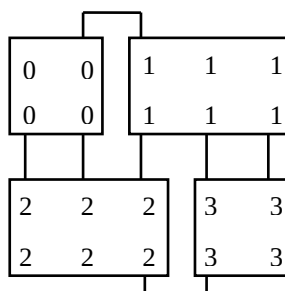


Figure II.25 Circuit de 4 cellules et 20 bornes.

Des trois circuits, seuls le premier et le troisième amènent le programme à converger vers la bonne solution (figure 26 et 28). En ce qui concerne le deuxième circuit, l'algorithme ne converge pas. Le programme boucle sans fin. Le problème provient de ce qu'après quelques itérations normales, deux cellules se placent en même endroit, l'une éjectant l'autre.

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - 0 1 - - - - -
- - - 0 1 - - - - -
- - - - -
- - - - -
- - - - -

```

Figure II.26 Solution du problème de la figure 23.

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - 0 1 3 - - - - -
- - - 0 1 3 - - - - -
- - - - -
- - - - -
- - - - -

```

Figure II.27 Solution du problème de la figure 24

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - 0 0 1 1 1 - - -
- - - 0 1 1 1 1 - - -
- - - 2 2 2 3 3 - - -
- - - 2 2 2 3 3 - - -
- - - - -

```

Figure II.28 Solution du problème de la figure 25

Lorsqu'une cellule prétend à une place déjà occupée, le conflit est réglé en déterminant à quelle cellule cet espace convient le mieux. La cellule éjectée va être replacée tout de suite après grâce à la pile. Elle va certainement aspirer à la position qu'elle vient de perdre. Un nouveau conflit éclate mais, normalement la cellule évincée doit être vaincue. Dans notre cas, elle retrouve sa place et l'ancienne cellule gagnante est ôtée du réseau.

Ce cycle se poursuit apparemment indéfiniment. La solution la plus simple pour éviter ce problème est de désactiver la pile (figure II.29). Ce qui nous donne un bon placement pour le circuit de figure II.24.

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - 0 1 2 3 - - - - -
- - - 0 1 2 3 - - - - -
- - - - -
- - - - -
- - - - -

```

Figure II.29 Solution du problème de la figure 25 après désactivation de la pile

Nous allons maintenant effectuer un test sur un circuit où le nombre des cellules est plus important et les connexions entre les bornes sont plus denses (figure II.30)

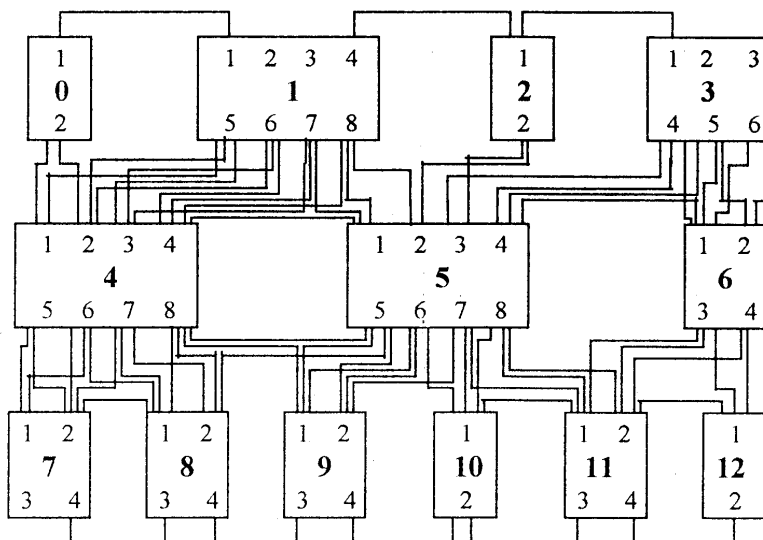


Figure II.30 Circuit de 13 cellules et 58 bornes

La solution de ce problème est montrée par la figure II.31

```

- - - - -
- - - 0 1 1 1 1 - 2 2 2 3 - -
- - - 0 1 1 1 1 - 2 2 2 3 - -
- - 4 4 4 4 5 5 5 5 6 6 - - -
- - 4 4 4 4 5 5 5 5 6 6 - - -
- - 7 7 8 8 - 9 9 10 12 11 11 - -
- - 7 7 8 8 - 9 9 10 12 11 11 - -
- - - - -
- - - - -

```

Figure II.31 Solution du problème de la figure II.30.

Il est clair que cette solution est mauvaise, en enlevant le prétraitement on arrive à une meilleure solution (figure II.32)

```

- - - - -
- - - - -
- - 0 1 1 1 1 2 2 2 3 - - - -
- - 0 1 1 1 1 2 2 2 3 - - - -
- - 4 4 4 4 5 5 5 5 6 6 - - -
- - 4 4 4 4 5 5 5 5 6 6 - - -
- - 7 7 8 8 9 9 10 11 11 - 12 - -
- - 7 7 8 8 9 9 10 11 11 - 12 - -
- - - - -
- - - - -

```

Figure II.32 Solution du problème de la figure II.30 sans le prétraitement.

II.16 CONCLUSION

La recherche d'un neurone gagnant et le règlement d'un conflit restent efficaces, même si le placement d'une cellule s'effectue sur plusieurs neurones. La division d'une cellule en un ensemble de bornes est une bonne idée. D'après les tests qu'on a effectué on a constaté que l'origine de la mauvaise qualité du placement de certains exemples est due au prétraitement. Celui-ci est identique à celui du premier algorithme, étant donné qu'il travaille sur la matrice de connexion, il ne prend en compte que des bornes et pas des cellules, il ne peut être efficace car il ne prend pas en considération toutes les possibilités de placement, et il ne doit établir des liens entre cellules que si celles-ci contribuent à structurer l'ensemble du circuit.

Au terme de l'étude du cas fondamental, une première conclusion partielle peut être tirée. L'approche neuronale est applicable pour l'optimisation du placement de cellules VLSI. Malgré les hypothèses très restrictives posées, l'algorithme de départ ne donnait que des résultats insatisfaisants.

Nous avons amélioré ce travail et des tests sur de véritables circuits peuvent être envisagés. Ces améliorations portaient plus sur l'obtention de solutions acceptables plutôt que d'optimiser la vitesse d'exécution. En ce qui concerne les cellules de tailles différentes, la division d'une cellule en plusieurs bornes semble être une bonne idée. Cette méthode doit permettre d'élargir les problèmes traités aux cas de circuits comportant des macro-cellules.

Nous avons rencontré des problèmes de non terminaison dans le dernier algorithme, à cause sûrement du prétraitement, car nous ne pouvons pas prévoir son efficacité pour des problèmes de grande taille.

Comme perspective, et pour résoudre le problème de la non terminaison, le prétraitement doit prendre en considération toutes les possibilités de placement et n'établit des liens entre cellules que si ceux-ci contribuent à structurer l'ensemble du circuit. Il faut aussi prévoir un test plus élargi en considérant les circuits de référence [18] (benchmarkd) pour mieux évaluer les performances de l'algorithme.

RÉFÉRENCES DU CHAPITRE 2

- [1] Kohonen T; "Self-Organization and Associative Memory", 1989. 3rd Edition. Springer-Verlag. NY USA
- [2] Hemani A; Postula A; "Cell placement by self-organization. Neural Networks".Vol.3,. pp. 377-383, 1990.
- [3] Pinaki Mazumder; "VLSI Cell Placement Techniques", ACM Comput. Surv. 23(2) pp 143-220, 1991
- [4] Mehmet Can Yildiz, Patrick H. Madden; "Global objectives for standard cell placement", ACM Great Lakes Symposium on VLSI 2001 68-72
- [5] A. Kennings, "Cell Placement Using Constructive and Iterative Methods", PhD thesis, University of Waterloo, Ont. Canada, 1997.
- [6] Guofang Nan, Minqiang Li, Dan Lin, Jisong Kou, "Adaptative Simulated Annealing for standard Cell Placement", ICNC (3) pp 943-947. 2005
- [7] S. Goto, "An Efficient Algorithm for the Two-Dimensional Placement Problem", in Electrical Circuit Layout. IEEE Transactions on Circuits and Systems, CAS-28, Jan. 1981.
- [8] W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of MacroCells," Proc. Int. Conf. on Computed-Aided Design, 1990, pp. 336- 339.

- [9] U. Lauther, "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," Proc. 16th Design Automation Conference, 1979, pp. 1-10.
- [10] Jason Cong, Michail Romesis, Min Xie "Optimality, Scalability and Stability Study of partitioning and Placement Algorithms", *ISPD '03*, April 6-9, 2003, Monterey, California, USA.
- [11] S. Areibi, M. Moussa and H. Abdullah, "A Comparison of Genetic/Memetic Algorithms and Other Heuristic Search Techniques", in International Conference on Artificial Intelligence, pp. 660–666, Las Vegas, Nevada, June 2001.
- [12] K. Shahookar and P. Mazumder, "A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization," IEEE Trans. on Computer-Aided Design, Vol. 9, No. 5, May 1990, pp. 500-513.
- [13] P. Mazumder and E.M. Rudnick, "Genetic Algorithms for VLSI Design", Layout & Test Automation, Prentice Hall, Toronto, Canada, 1999.
- [14] Ray-I Chang and Pei-Yung Hsiao, "Arbitrarily Sized Cell Placement by Self-organizing Neural Networks", 1993 IEEE.
- [15] H. Azizi, K. Benmmahamed, O.E. Aktouf, "VLSI cells placement, using Kohonen networks", *AJIT*, 5 (2), pp 183-198, 2006.
- [16] Laarhoven V, and Aarts E; "Simulated Annealing: Theory and Application". D. Reidel publishing, Dordrecht, The Netherlands, 1987.
- [17] H. Azizi, K. Benmmahamed, L. Zouaoui, "Aide au placement de cellules par recuit simulé", *ICCMD'06*, May 22-24, Annaba 2006.
- [18] Patrick H. Madden, "Reporting of Standard Cell Placement Results", IEEE Transactions on CAD of IC and systems, Vol. 21, N°2, Fev. 2002

Algorithme de Routage

III.1 INTRODUCTION

Les progrès technologiques constants effectués dans le domaine des circuits intégrés permettent d'implanter des circuits de plus en plus complexes pour un coût de fabrication de plus en plus faible.

Cependant, l'accroissement de la complexité des circuits électroniques a introduit un temps de conception de plus en plus lourd; ceci est vrai en particulier dans les étapes d'implantation des composants et de test des circuits. Pour réduire ces coûts, l'industrie des circuits électroniques a fait appel à de nouveaux moyens tel que la conception assistée par ordinateur CAO. Cette dernière intervient aux différentes étapes de la conception d'un circuit électronique et en particulier lors de la phase de routage.

Pour le problème de routage, il s'agit d'implanter la totalité des connexions tout en respectant les règles de dessin et en optimisant certains critères tels que la longueur de connexion, le nombre de points de liaison entre deux niveaux...

On distingue deux approches fondamentales du routage; un routage qui est essentiellement utilisé sur des structures régulières présentant des canaux d'interconnexion et des passages prédéterminés entre les cellules. Cette méthode utilise deux étapes [1][2]: routage global et routage détaillé (circuit intégré) et un routage pour structures irrégulières (circuit imprimé).

Dans ce chapitre, après une brève présentation des algorithmes développés par les scientifiques, nous allons présenter l'algorithme A* [3] que nous avons utilisé et ensuite amélioré avec l'adaptation de celui-ci aux problèmes de connexions à plusieurs terminaisons et l'utilisation de plus d'un niveau de connexion comme établi avec la version initiale de l'algorithme.

III.2 FORMULATION DU PROBLEME

Le problème du routage consiste à trouver une manière de relier plusieurs points grâce à des liaisons physiques tout en respectant certaines contraintes.

Typiquement, dans un espace discrétisé composé d'un ensemble de points, étant donné une liste $L = \{L_i\}$ de liaisons $L_i = \{S_i D_i\}$ à créer avec $S_i \in P$ et $D_i \in P$, un tel algorithme doit retourner une liste de connexions $C = \{C_i\}$ où $C_i = \{S_i, P_{i,1}, P_{i,2} \dots P_{i,j-1}, P_{i,j} D_i\}$. De plus, il est en général intéressant de générer des chemins les plus courts possibles, afin de minimiser les délais sur les fils.

III.3 LES APPROCHES DU ROUTAGE

Il existe deux approches pour le problème de routage, l'approche séquentiel et l'approche concurrent [4] [5].

1. *Approche séquentielle*: Cette approche possède les caractéristiques suivantes :

- Connecte un seul noeud à la fois.
- L'ordre dépend de certains facteurs tels que l'estimation de la longueur totale de connexions et le nombre de terminaux.
- Si certaines connexions ne peuvent être établies à cause du blocage des autres connexions déjà réalisées, alors il faut appliquer la technique du "Rip-up and_Reroute" .

2. *Approche concurrent* : Cette technique possède les caractéristiques suivantes :

- Considère tous les noeuds en même temps, c'est-à-dire sans aucun ordre pré-établi.
- Peut être formulé par la programmation parallèle.

La figure III.1 donne les différents types de routage.

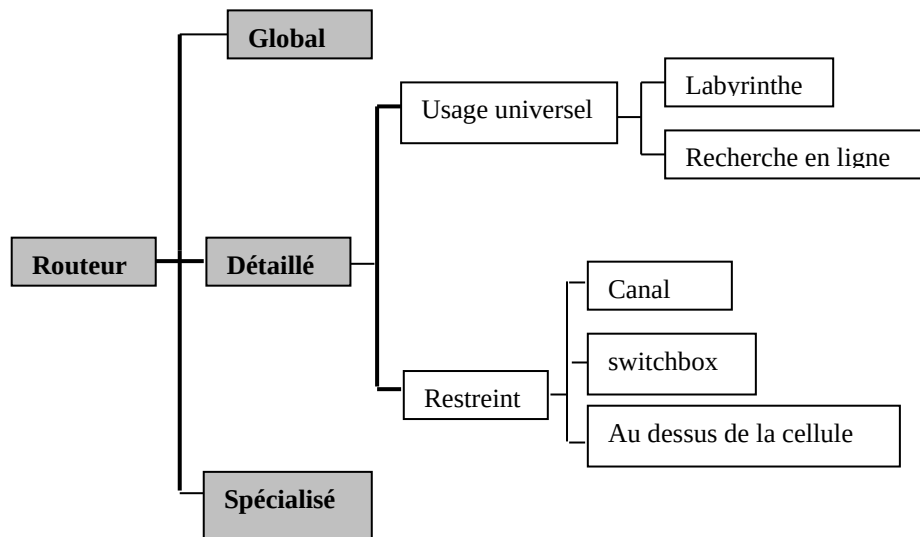


Figure III.1 Classification du routage

III.3.1 ROUTAGE GLOBAL

Pour résoudre le problème du routage global, il faut donc choisir un chemin à travers les zones de routage dont on dispose pour chacune des interconnexions à réaliser. Une solution doit posséder essentiellement deux qualités:

- a) respecter les contraintes imposées par la densité maximale des divers canaux de routage ;
- b) permettre de réduire le plus possible la longueur des interconnexions.

Le routage global est divisé en trois phases:

1. Définition des zones
2. Assignement des zones
3. Assignement des noeuds aux zones de routage

III.3.1.1 DEFINITION DES ZONES

Dans cette phase, il faut diviser l'espace en zones de routage rectangulaires. (Figure III.2)

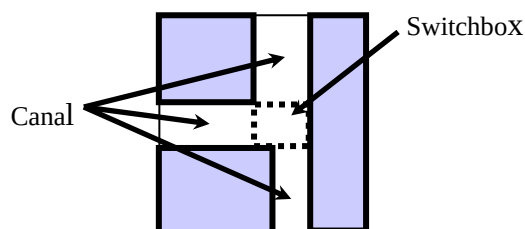


Figure III.2 Définition des zones

Les zones de routage peuvent être de plusieurs types :

- Canal: les noeuds sont dans des faces opposées.
- 2-D Switchbox: présence de noeuds sur les 4 faces.
- 3-D Switchbox: présence des noeuds sur les 6 faces.

Ces zones dépendent du style de circuit utilisé. Dans la conception en portes logiques (gate array) et en cellules standard (standard cell), l'espace de routage est principalement les canaux de routage horizontaux. Le canal est une région limitée avec des points de connexion sur la borne supérieure et inférieure. La connexion inter canal se fait au moyen d'allée (avenue) verticale sur les faces ou au centre de la puce. La principale différence entre les deux approches est que la hauteur des canaux pour les portes logiques est fixe tandis qu'en conception de cellules standard la hauteur des canaux n'est pas spécifiée.

Dans la conception de circuit sur mesure (full custom), les modules sont de configuration et de dimension irrégulières. Les zones de routage sont définies après le placement.

Pour les trois approches, il s'agit de trouver le plus court chemin pour connecter les réseaux. Des méthodes standard existent, par exemple l'algorithme de LEE [6] qui permet de connecter un réseau à deux points. Pour un réseau à plusieurs points la méthode de l'arbre de Steiner [7] est la plus utilisée.

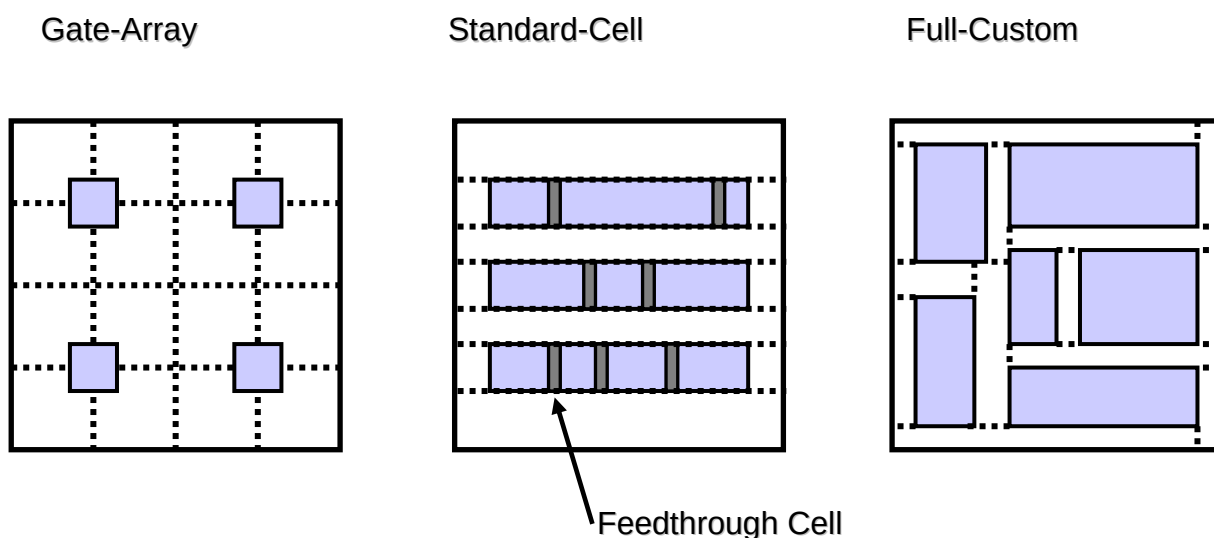


Figure III.3 Zones de routage dans les différents styles de Conception

III.3.1.2 ASSIGNEMENT DES ZONES

Dans cette phase, il s'agit d'assigner à chaque nœud des zones de routage (Figure III.4)

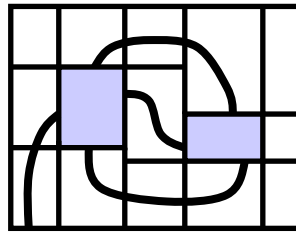


Figure III.4 Assignment des zones

III.3.1.3 ASSIGNEMENT DES POINTS DE CONNEXION

Assigner chaque point de connexion dans les bandes des zones de routage. (Préparer la phase de routage à canal) (Figure III.5).

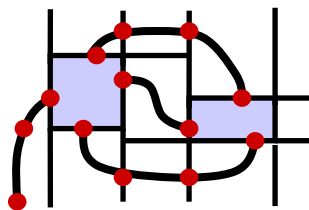


Figure III.5 Assignment des points de connexion

En résumé, Puisque toutes les versions du problème de routage global sont NP-complete [1], une variété d'algorithmes heuristiques a été développée. Ces algorithmes peuvent être classés en deux catégories : *Routage global séquentiel* et *routage global concurrent*. L'approche la plus commune au routage global est le routage séquentiel. Dans une telle approche, les nœuds sont d'abord classés selon leur importance, ensuite ils seront connectés séquentiellement. La qualité du routage dépend en grande partie de l'ordre des nœuds. La figure III.6 donne la classification du routage global

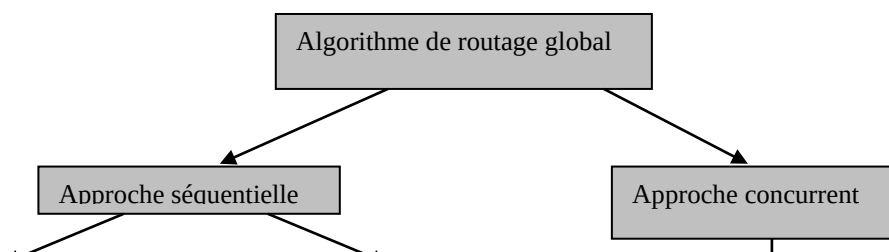


Figure III.6 Classification du Routage Global

III.3.2 ROUTAGE DETAILLE

Dans cette phase, il s'agit d'implémenter physiquement les connexions. Chaque nœud étant assigné à une zone. Il est exécuté en deux temps :

- dans un premier temps les nœuds seront router des bornes de cellules vers les canaux de routage
- la deuxième phase correspond au routage à canal où les bornes de connexion sont dans un canal à deux faces (canal), à quatre faces (switchbox 2D) ou à six faces (switchbox 3D)

III.4 ETAT DE L'ART

Cette partie est consacrée aux algorithmes expansion-de-grille (labyrinthe) et recherche en ligne. Ce sont les méthodes les plus classiques et les plus utilisées dans les problèmes de routage. Nous allons donner les différents algorithmes de routage.

Kruskal est le premier, en 1956, à proposer une méthode de création de l'arbre couvrant minimal d'un graphe non-orienté [7]. Partant d'un graphe $G = (V, E)$, où V est un ensemble de noeuds, et un ensemble d'arêtes, associées à un poids et connectant ces noeuds, il permet de créer l'arbre de poids minimum.

L'approche, triviale, consiste en la création d'un sous-graphe du graphe initial par ajout successif des arêtes. A chaque pas, une nouvelle arête qui ne crée pas de cycle, et dont le poids est minimum,

est ajoutée à la liste finale. Lorsque toutes les arêtes du graphe initial ont été testées, le graphe final constitue l'arbre de poids minimum.

III.4.1 ALGORITHME DE LEE

L'algorithme de LEE est le premier à avoir utilisé le câblage automatique dans les problèmes de routage. Il permet de trouver le chemin de longueur minimal. Beaucoup d'auteurs ont contribué à l'amélioration de l'algorithme original de LEE.

Bien que cet algorithme garantisse une solution, si elle existe, deux problèmes le rendent inadéquat:

- 1) il utilise un espace mémoire énorme,
- 2) le temps de calcul est proportionnel au carré de la distance mesurée en unités élémentaires de la grille.

Une autre classe d'algorithmes appelée recherche en ligne, dans laquelle la recherche du chemin de connexion est représentée par une séquence de segments, permet de minimiser les problèmes énoncés ci-dessus.

Cependant, l'algorithme de Lee, qui est cité par tous ses successeurs, présente une méthode de spécification de problèmes de traitement de patterns, qu'il applique à la création de chemins dans un graphe régulier.

Commençons par définir un ensemble de cellules (Lee est le premier à introduire le terme *cellule*) : $C = \{c^1, c^2, \dots\}$ pour chaque cellule $c^j \in C$ nous définissons son 1-voisinage $N(c^j) = \{c_1^j, c_2^j, \dots, c_n^j\}$ qui doit obéir aux deux règles suivantes:

- Tout 1-voisinage a exactement n cellules, où $n \geq 1$.
- $c^j \in N(c^i) \Leftrightarrow c^i \in N(c^j)$, qui signifie que le graphe est non-orienté.

Pour deux cellules c^j et c^i distinctes, nous définissons un chemin entre ces deux cellules par $p(c^i, c^j) = \{c^0 = c^i, c^1, \dots, c^m = c^j\}$, de manière à ce que $c^{i+1} \in N(c^i)$ pour tout $i=0,1,\dots,m-1$.

Pour le problème du plus court chemin, nous pouvons définir la fonction f comme ceci :

$$\begin{aligned}
 &1) \quad f(p(c^*, c^*)) = 0 \\
 &2) \quad \text{Si } s(c^i) \text{ est inoccupée, alors} \\
 &f(p(c^*, c^i)) = \begin{cases} \min\{ f(p(c^*, c^j)) \mid c^j \in N(c^i) \} + 1 \\ \text{pour tout } f(p(c^*, c^j)) \text{ ayant été définie,} \\ \text{et indéfinie sinon} \end{cases} \quad (\text{III.1})
 \end{aligned}$$

Il est intéressant de noter que LEE, dans son article, ne propose pas directement de solution pour le plus court chemin, mais une solution pour minimiser la distance totale et le contact avec des cellules occupées. Pour ce faire, il utilise deux fonctions, dont une est celle de l'équation III.1. Il est évident que le plus court chemin était créé avec cette fonction-ci, et c'est cet algorithme qui est toujours cité. Or, par le mécanisme de son algorithme général, disposer d'une simple fonction $f(p(c^*, c^i)) = 0$ suffit à trouver le plus court chemin entre deux points. Ceci évite de devoir stocker la distance à l'origine de chaque point, ce qui est toujours reproché à Lee, et ne nécessite que de mémoriser le prédécesseur de chaque cellule.

A partir de maintenant, nous nous référons au cas de la recherche du plus court chemin grâce à la fonction III.1 comme étant l'algorithme de LEE. Appliqué à une grille d'éléments reliés à leurs quatre voisins, il fonctionne à la manière dont une vague se propage à partir de la source, créant une structure en forme de diamant. Dans ce que nous appellerons dorénavant la phase d'expansion, un front d'onde atteint les cellules les unes après les autres et lorsqu'une cellule est touchée par le front d'onde, elle stocke sa distance à la source en incrémentant de 1 la distance de la cellule par laquelle l'expansion est arrivée. La phase d'expansion se termine lorsque la destination est atteinte, signe que le chemin est trouvé. Il ne reste plus alors qu'à remonter jusqu'à la source en passant d'une cellule de distance d à une de distance $d-1$ jusqu'à arriver à la source, qui a une distance de 0 (phase de rétropropagation). La figure III.7 montre, pour un exemple, les pas de l'algorithme lors de l'expansion, ainsi que le chemin généré.

L'aspect fondamental de l'algorithme de Lee réside en son parallélisme intrinsèque. En effet, dans sa présentation, Lee propose une approche séquentielle, où les cellules sont ajoutées au

front d'onde les uns après les autres, mais il est clair qu'avec un système parallèle la phase d'expansion pourrait être plus efficace. Alors que la solution séquentielle a une complexité de $O(d^2)$ pour une distance entre la source et la destination de d , la solution parallèle offre une complexité de $O(d)$.

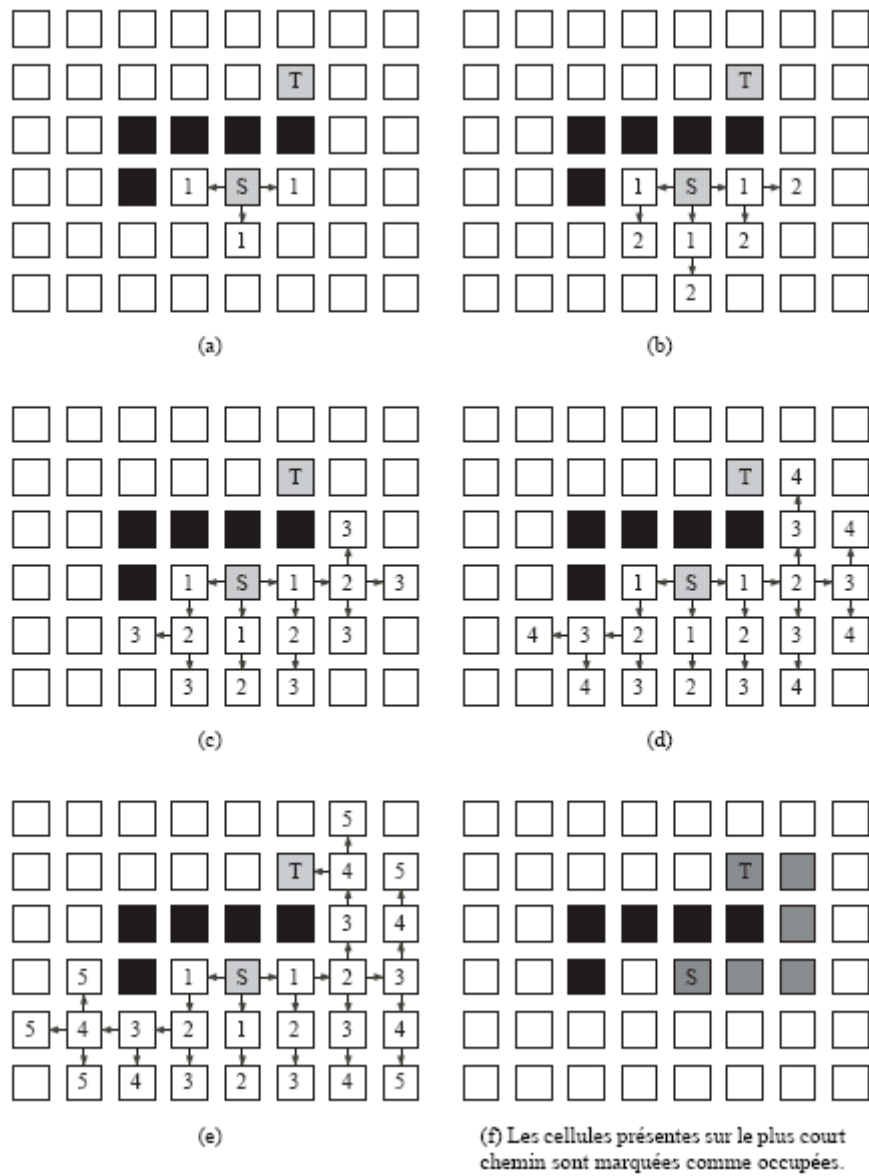


Figure III.7 Déroulement de l'algorithme de LEE

(S est une source, et T est une destination).

III.4.2 VARIATIONS SUR LEE

De nombreuses variations sur l’algorithme de LEE ont été proposées. Nous en présentons ici les principales, mais on pourra en trouver des secondaires dans [8, 9,10].

III.4.2.1 AKERS

En 1967, Akers publie un article [10] où il propose une variante de l’algorithme de LEE, appliquée à une grille rectangulaire de cellules, dans laquelle sont placées des barrières. Au lieu de mémoriser, dans chaque cellule, la distance à l’origine, il lui suffit d’y placer une valeur binaire, réduisant ainsi la quantité de mémoire nécessaire à la bonne marche de l’algorithme.

Il part de l'observation suivante : Dans l'algorithme de Lee, une cellule qui a été atteinte par la phase d'expansion et numérotée X a comme voisines des cellules qui sont soit vides, soit des murs, soit qui contiennent la valeur $X - 1$ ou $X + 1$. Il suffit donc, pour pouvoir tracer le chemin durant la phase de rétropropagation, que le prédécesseur ait une valeur différente de celle de son successeur, et la séquence 1, 1, 2, 2, 1, 1, 2, 2,... remplit parfaitement cette condition.

L'algorithme de Lee est donc adapté de manière à placer la séquence susmentionnée au lieu de la séquence 1, 2, 3, 4, 5, ... dans les cellules visitées. Il ne reste plus qu'à la phase de rétropropagation de parcourir la séquence dans l'ordre correct, en fonction de la manière (11, 12, 21 ou 22) dont l'expansion a atteint le point B (Figure III.4).

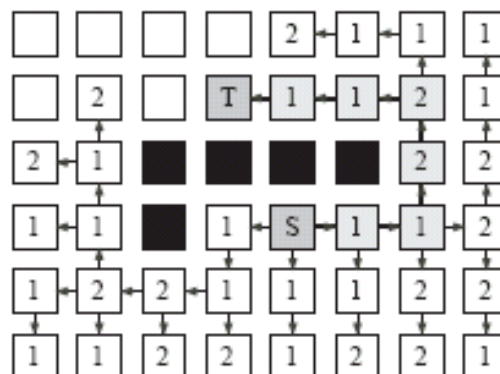


Figure III.8 Le plus court chemin trouvé grâce à l'algorithme de Akers.

III.4.2.2 RUBIN

En 1974, Rubin [11] propose des améliorations à l'algorithme de Lee dans l'optique de réduire le temps de calcul. Les deux plus importantes sont les suivantes :

- *Recherche bidirectionnelle* Une solution de Pohl suggère de lancer la phase l'expansion depuis la source ET la destination. De cette manière, le nombre de cellules explorées peut être réduit, la somme des cellules visitées par les deux vagues étant plus petite que celle d'une seule vague.
- *Profondeur d'abord* Une autre méthode, développée par Rubin, utilise une recherche en profondeur d'abord. L'expansion se poursuit prioritairement dans la direction de la destination, tant qu'aucun obstacle n'est présent. Le temps d'exécution peut être grandement réduit par cette approche, mais une connaissance globale de la position de la destination est nécessaire, ce qui n'est pas la panacée dans le cas d'une implémentation matérielle.

III.4.2.3 HOEL

En 1976, Hoel [12] propose deux nouvelles variations à l'algorithme de LEE, pour des coûts de chemins quelconques. La première est purement algorithmique, vouée à une implémentation logicielle, et propose d'utiliser plusieurs listes pour le stockage des cellules présentes sur la frontière, afin d'améliorer la rapidité de la recherche de la prochaine cellule à étendre. La deuxième est très intéressante, et vise à réduire la mémoire nécessaire à l'exécution de l'algorithme. Alors que l'approche de Akers qui plaçait des 1 et des 2 dans les cellules atteintes par l'expansion n'est valable que dans le cas où il n'y a pas de coûts différents entre les cellules, celle de Hoel est applicable à des coûts différenciés. Seuls 2 bits par cellule sont nécessaires, et permettent de définir si la cellule est inoccupée, occupée, ou sur le front, de même que l'origine de l'expansion : occupé=tout droit=0, frontière=virage à gauche=1, inoccupé=2, virage à droite=3. Cette réduction, qui économise un maximum la mémoire, a toutefois un coût. Durant l'exécution de la création d'un chemin, la configuration initiale (cellules inoccupées ou occupées) est perdue, et une mémoire secondaire où le tableau initial (avec les cellules occupées) est stocké est nécessaire. Après chaque création d'un nouveau chemin, il doit être mis à jour avec les nouvelles cellules occupées. Finalement, il est intéressant de noter que cette approche peut être appliquée à un voisinage de 6, sans que des bits supplémentaires ne soient nécessaires.

III.4.2.4 MIKAMI ET TABUCHI

En 1968, Mikami et Tabuchi [13] introduisent un nouvel algorithme, appelé linesearch, pour le routage d'un circuit imprimé dont un des côtés peut contenir des lignes horizontales, et l'autre des verticales, les deux couches étant reliées par des trous. Leur but est d'offrir une alternative moins coûteuse en terme de temps d'exécution et de mémoire à celui de Lee et de Akers. Au lieu de travailler sur une grille de cellules, l'algorithme stocke des lignes, déterminées par seulement trois coordonnées. La recherche s'effectue alternativement depuis la source et la destination, par l'extension de lignes étendues dans les quatre directions. Les lignes originaires de la source sont comparées à celles originaires de la destination afin de détecter la fin de l'algorithme, lorsque deux lignes se croisent. En outre, l'algorithme exploite les lignes déjà connectées à la source ou à la destination, rendant le mécanisme de liaison, initialement point-à-point, point-à-chemin ou chemin-à-chemin. De cette manière, il évite de créer de nouvelles lignes qui risqueraient de surcharger la carte inutilement.

Cet algorithme garantit de trouver une solution si elle existe, mais ne fournit pas forcément le chemin le plus court entre les deux points, mais le chemin le plus simple, c'est-à-dire avec le moins de coins possibles.

III.4.2.5 BACKTRACKING

Le routage à la manière de Lee peut conduire à une impasse si trop de chemins doivent être créés. Cette congestion peut être évitée par l'utilisation de backtracking. Agrawal et Breuer, en 1977, introduisent un algorithme basé sur deux principes supplémentaires: le backtracking et la déviation. Les connexions y sont réalisées dans un ordre quelconque, mais après chaque ajout, un test détermine si les connexions restantes sont forcément bloquées par celui-ci. Si c'est le cas, l'ajout est supprimé et un autre chemin est cherché. Dans le cas où aucun des plus courts chemins n'est acceptable, une déviation de taille δ est autorisée, et agrandit le chemin de δ cellules.

Cet algorithme, dont tous les détails peuvent être trouvés dans [9], garantit de trouver une solution au problème général du routage de N chemins, si une telle solution existe. Toutefois, le prix à payer

pour une telle réussite est le temps de calcul, qui peut s'avérer prohibitif, de par les nombreuses phases potentielles de backtracking.

III.4.2.6 SOUKUP

En 1978, Soukup [14] propose une amélioration de l'algorithme de Lee, qui exploite une recherche en profondeur d'abord, quasiment de la même manière que Rubin. L'expansion se poursuit dans le sens de la destination, tant qu'aucun obstacle n'est rencontré.

Lorsque cette expansion de type line-search n'est plus possible, l'algorithme de Lee prend le pas, jusqu'à ce que l'obstacle ait été contourné. Le temps d'exécution peut être grandement réduit par cette approche, mais une connaissance globale de la position de la destination est nécessaire, ce qui n'est pas la panacée dans le cas d'une implémentation matérielle.

Les figures III.8 et III.9 illustrent la différence entre l'approche de Soukup et celle de Lee dans le nombre de cellules visitées lors de la recherche de la destination. Dans la figure III.8, les points noirs ont été atteints par une expansion de type line-search, alors que les blancs l'ont été par une expansion de type Lee.

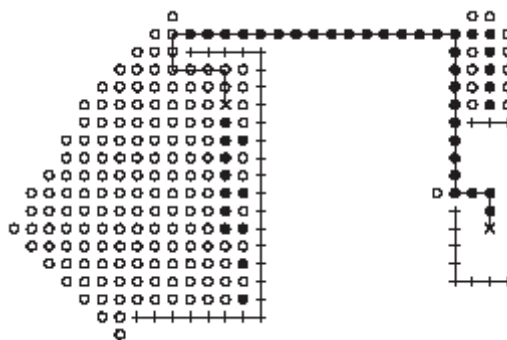


Figure III.8 Nombre de cellules visitées par l'algorithme de Soukup.

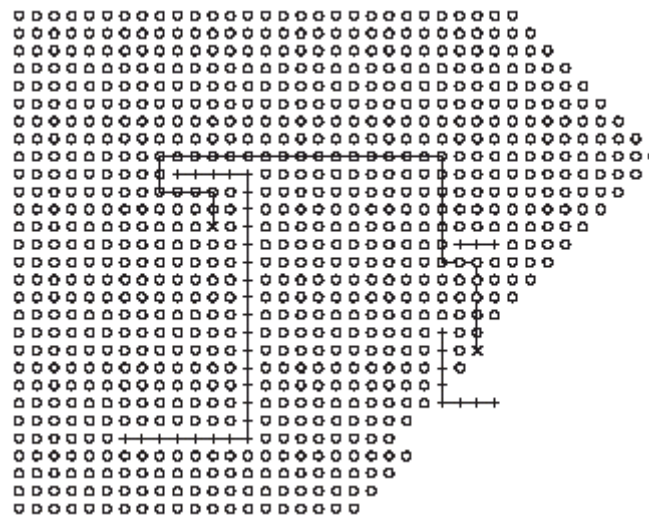


Figure III.9 Nombre de cellules visitées par l'algorithme de Lee,
À comparer avec l'approche de Soukup, figure III.8

D'autres algorithmes de routage, dont notamment [15, 16, 17, 18], dédiés aux FPGA, ont également vu le jour après les années 85, date de création des premiers FPGA. Le problème global y est légèrement différent que pour les ASIC, de par la structure fixe des circuits reconfigurables, et le nombre limité de ressources de routage disponibles. L'optimisation y a pour but de router l'ensemble des connexions demandées, ainsi que de minimiser les délais de propagation. D'autres variations de l'algorithme de LEE ont été développées [19, 20, 21, 22, 23]

III.5 ROUTAGE CONCURRENT

Cette approche du routage considère tous les nœuds simultanément (aspect parallèle) et peut être formulée par la programmation entière. Dans ce cas, le routage procède à une hiérarchisation de l'espace de routage. La figure III.10 donne un exemple de décomposition dans le cas d'un routage hiérarchique.

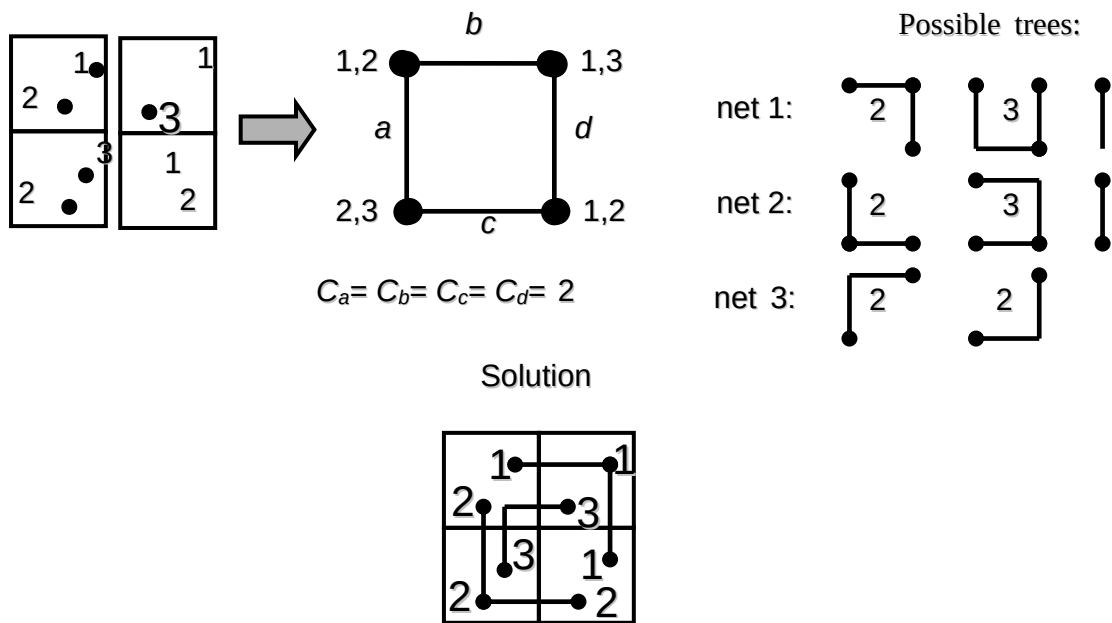


Figure III.10 Routage hiérarchique

III.6 APPLICATION

III.6.1 ALGORITHME DE ROUTAGE

L'algorithme utilisé dans l'étape de routage est l'algorithme A*[24]. C'est un algorithme qui utilise la recherche heuristique (une heuristique est une technique qui aide à la découverte de solutions aux problèmes bien qu'il n'y ait aucune garantie qu'elle ne mènera jamais dans la mauvaise direction) et le choix de la fonction heuristique appropriée est important dans le guidage efficace du processus de recherche vers une solution.

Pour appliquer l'algorithme aux circuits intégrés, il a fallu procéder à quelques adaptations [25]:

- Dans sa version élémentaire, l'algorithme ne prévoit que des interconnexions à deux points. Toutefois, on peut raffiner l'algorithme pour traiter les interconnexions entre plusieurs points en étendant autant d'onde qu'il y a de points à interconnecter; lorsque deux ondes se rencontrent une des deux peut cesser de se propager jusqu'à ce que toutes les ondes soient en contact.
- Lors de la phase de propagation d'onde, on ne trouve pas de cellules libres, la nouvelle adaptation permet de créer un autre niveau.

Alors que la plupart des algorithmes précédents effectuent une recherche en largeur d'abord, il est possible de chercher en profondeur d'abord, c'est-à-dire, dans un processus séquentiel, d'étendre le dernier noeud visité plutôt que le premier. Sur le plan de l'implémentation logicielle, ceci correspond à utiliser une liste LIFO (Last In First Out) plutôt que FIFO (First In First Out). La recherche "meilleur en premier" utilise une connaissance de la distance potentielle au but pour choisir le noeud à étendre. Il s'agit toujours du noeud qui est potentiellement le plus proche de la destination. Dans le cas d'une grille à deux dimensions, comme pour le problème qui nous intéresse, la distance à la destination est simplement la distance de Manhattan (La distance entre deux points du plan, $A = (ax, ay)$ et $B = (bx, by)$ vaut $|ax - bx| + |ay - by|$) entre le noeud courant et la destination. Cet algorithme garantit de trouver une solution si elle existe, mais elle n'est pas forcément la plus courte.

L'algorithme de recherche A^* , qui est grandement utilisé en intelligence artificielle, est de type "meilleur en premier", et tire parti d'un heuristique qui combine le coût $g(n)$ du chemin entre la source et le noeud courant avec le coût potentiel $h(n)$ entre ce noeud et la destination:

$f(n) = g(n) + h(n)$ Le noeud ayant un $f(n)$ le plus faible est alors celui qui est étendu. Dans notre cas, $g(n)$ est calculé durant l'expansion, et correspond à la distance minimal entre le noeud et la source, tandis que $h(n)$ est la distance de Manhattan entre le noeud n et la destination. Le chemin trouvé par cet algorithme est toujours optimal si $h(n)$ ne surestime jamais le coût nécessaire à atteindre la destination.

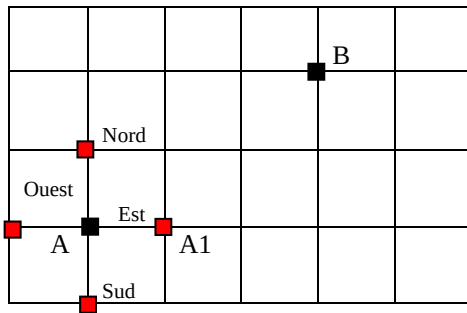
L'avantage de cet algorithme est que son nombre d'itérations est grandement réduit par rapport à une recherche en largeur d'abord. Toutefois, il ne se prête pas à une implémentation parallèle, car il nécessite une vision globale nécessaire à l'évaluation de $h(n)$, et que son essence même est séquentielle.

III.6.2 FONCTION HEURISTIQUE ET GENERATION DES NOEUDS

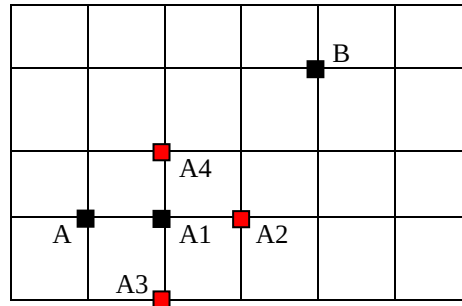
L'algorithme A^* est un moyen d'implémenter la recherche "meilleur en premier" des points générés dans la phase de propagation de l'onde de l'algorithme de Lee. A chaque pas du processus de recherche du meilleur en premier, il faut choisir le plus prometteur des points générés.

Ceci se fait en appliquant une fonction heuristique appropriée à chacun d'eux. Cette

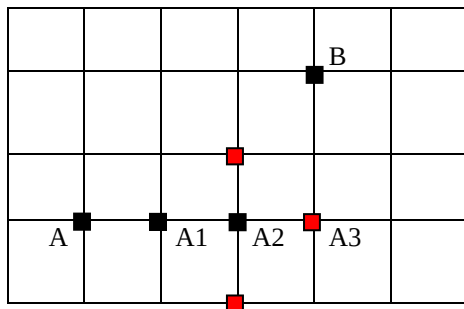
fonction est la distance de MANHATTAN qui sépare le noeud généré de la cible ($F' = \text{distance}(\text{noeud}, \text{cible})$). L'élargissement du point choisi se fait en générant 4 noeuds voisins: les noeuds est, ouest, nord et sud. Ceci pour respecter le modèle de MANHATTAN qui n'utilise que les segments horizontaux ou verticaux. Si l'un des points générés est une solution on peut arrêter. Sinon tous les nouveaux points sont ajoutés aux points générés jusqu'ici. De nouveau le point le plus prometteur est choisi et le processus continu. La figure III.11 donne l'élargissement de l'onde de propagation de l'algorithme :



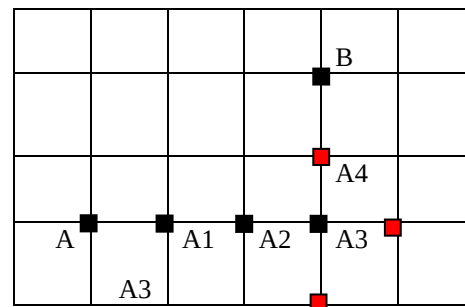
(a) le nœud retenu est A1



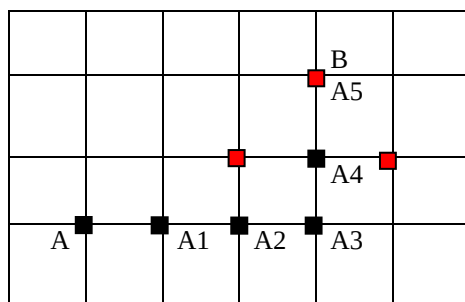
(b) le nœud retenu est A2



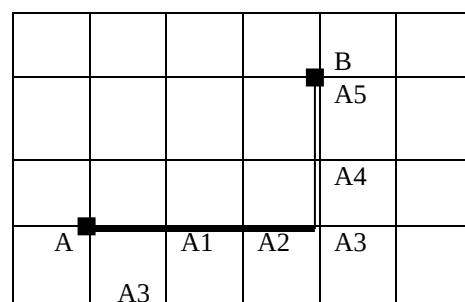
(c) le nœud retenu est A3



(d) le nœud retenu est A4



(d) le nœud retenu est A5 (cible)



(e) On trace la connexion

Figure III.11 Onde de propagation de l'algorithme A*

Nous aurons à utiliser deux listes de points :

- **OUVERTS** ; les points qui ont été générés et auxquels la fonction heuristique a été appliquée, mais qui n'ont pas encore été examinés (autrement leurs successeurs n'ont pas été générés). Cette liste est vraiment une file de priorité dans laquelle les éléments avec la priorité la plus élevée sont ceux affectés de la valeur la plus prometteuse de la fonction heuristique.
- **FERMES** ; les points qui ont été déjà examinés.

Nous aurons également besoin d'une fonction heuristique qui estime les mérites de chacun des points que nous avons générés. Appelons cette fonction f^* pour indiquer que c'est une approximation d'une fonction f qui donne la véritable évaluation. Alors, la mise en œuvre de l'algorithme est très simple. Elle se fait par étape, en élargissant un nœud à chaque pas, jusqu'à ce que soit généré un point qui correspond à un état but. A chaque pas, l'algorithme prend le plus prometteur des points générés jusque là mais non élargis. Il génère les successeurs du point choisi, leur applique la fonction heuristique et les ajoute à la liste des points OUVERTS après avoir examiné si chacun d'eux n'a pas été généré auparavant. Ce processus est décrit ci-dessous :

Etape 1 :

Commencer avec OUVERTS contenant seulement le point initial. Mettre la fonction heuristique à zéro et mettre FERMES sur la liste vide

Etape 2 :

Jusqu'à ce qu'un point but (cible) soit trouvé, répéter la procédure suivante ; s'il n'y a aucun point sur OUVERTS signaler l'échec, sinon prendre le point sur OUVERTS qui a la valeur f^* la plus petite. L'enlever de OUVERTS et le placer sur FERMES. Voir si ce point est un point but, si oui sortir et signaler une solution, sinon générer les successeurs de ce point. Pour chaque successeur généré tester s'il est sur OUVERTS. Si oui le repositionner suivant sa valeur f^* , sinon s'il est sur FERMES le mettre sur OUVERTS.

Etape 3 :

Reconstruire le chemin trouvé à partir des points sélectionnés.

L'algorithme A* est plus rapide que l'algorithme de LEE avec l'incorporation de la fonction heuristique qui permet d'orienter les recherches en direction de la cible et éviter ainsi de générer des points de grille inutile. L'espace mémoire utilisé se trouve ainsi réduit par rapport à l'algorithme de LEE.

Algorithme III.1 Algorithme A* fondamental

DEBUT

OUVERTS = {S} (Le fichier OUVERTS contient le premier élément : la source)

FERMES = {} (Le fichier FERMES reçoit l'ensemble vide)

$F'(s) = 0$ (Fonction heuristique initialisée)

Pred(s) = S (Vecteur de reconstruction du trajet)

Found = Faux

Tant que OUVERTS $\neq \{\}$ et Not Found Faire

Début

X=Le premier élément de OUVERTS (le nœud avec la valeur F' la plus prometteuse)

OUVERTS = OUVERTS - X (enlever X de OUVERTS)

FERMES = FERMES + X (le mettre dans FERMES)

Si X = T (cible) Alors Found = Vrai

Sinon (Continuer les recherches à travers l'ensemble des nœuds voisins de X)

Début

Soit R l'ensemble des nœuds voisins de X

Pour tout y de R faire

Si $y \notin$ OUVERTS ni FERMES alors

$F'(y) = \text{Distance}(y, \text{cible})$

Pred(y) = X

OUVERTS = OUVERTS + {y}

Sinon (y est dans OUVERTS ou FERMES)

Si $y \in$ OUVERTS alors Placer y dans OUVERTS selon sa valeur F'

Sinon (y est dans FERMES)

FERMES = FERMES - {y}

OUVERTS = OUVERTS + {y}

Si Found alors tracer le chemin en utilisant le vecteur Pred

Sinon les S et T ne peuvent être connectés.

FIN

III.6.3 GRILLE

L'idée retenue est que la taille de la case élémentaire de la grille correspond à l'unité de longueur λ (plus petite distance pour une technologie de fabrication).

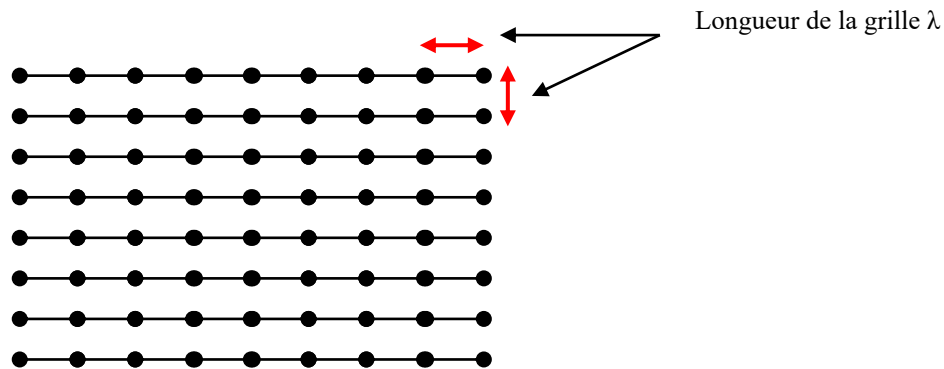


Figure III.12 Grille

III.6.4 CONNEXION A PLUSIEURS TERMINAISON

Dans le cas de connexions à plusieurs terminaisons, on choisit arbitrairement deux points qu'on relie ensuite par un premier segment. L'ensemble de ce segment représente alors une cible pour la prochaine source à considérer. Ce procédé est répété jusqu'à ce qu'il n'y ait plus de cases à connecter.

Ce problème peut résolu par extension à l'algorithme de Lee:

- Connecter un terminal à la fois ou
- Faire une recherche pour les différentes cible simultanément ou
- Propager l'onde à partir des différentes sources simultanément

Avant les solutions au plus court chemin, ce fut le problème de la création de l'arbre couvrant de poids minimum qui fut traité par les scientifiques. Il est défini comme l'arbre couvrant minimal (minimum spanning tree) d'un graphe, dont un exemple est montré à la figure III.13, est l'arbre contenant tous les noeuds (ou terminaux) du graphe, et dont la somme des poids des arêtes est minimale.

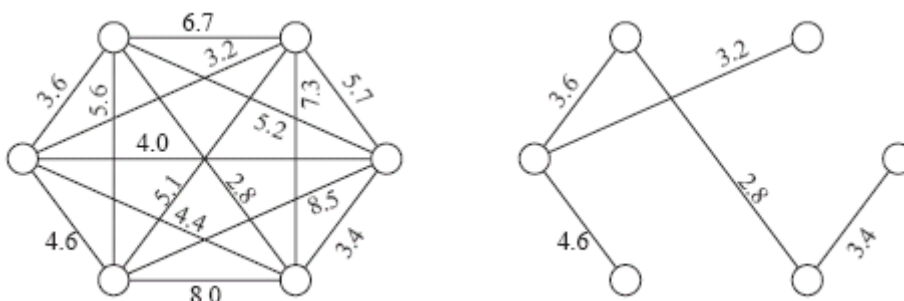


Figure III.13 A droite, l'arbre couvrant minimal du graphe de gauche.

L'arbre couvrant de poids minimum permet de relier plusieurs points par un réseau de poids minimum. Les réseaux électriques, par exemple, en tirent parti, afin de minimiser la quantité de câbles qui servent à interconnecter l'ensemble des points du réseau.

Notre algorithme, présente une solution pour les connexions multipoints, où une source doit être connectée à plusieurs destinations. Son but est de construire un arbre de Steiner quasi-minimum entre les différents points (L'arbre de Steiner de poids minimum d'un graphe connecte tous les noeuds de ce graphe par un arbre de poids minimum, qui peut inclure des nœuds supplémentaires appelés points de Steiner), afin de minimiser le nombre de ressources nécessaires. Pour y parvenir, une phase d'expansion part de la source, jusqu'à atteindre une destination (Figure III.14 (a)). Ensuite, une deuxième vague part de ce point pour retrouver la source (Figure III.14 (b)), et permet de définir l'ensemble des plus courts chemins entre les deux points (Figure III.14 (c)). Pour relier le point suivant, une expansion est lancée depuis cet ensemble de points (Figure III.14 (d)), et de la même manière une nouvelle zone est créée, les deux zones étant reliées par un point P (Figure III.14 (e)). Il ne reste alors plus qu'à relier les trois points en passant par ce point P (Figure III.14 (f)). Le procédé peut être réitéré dans le cas où plus de trois points sont à connecter, suivant le même principe.

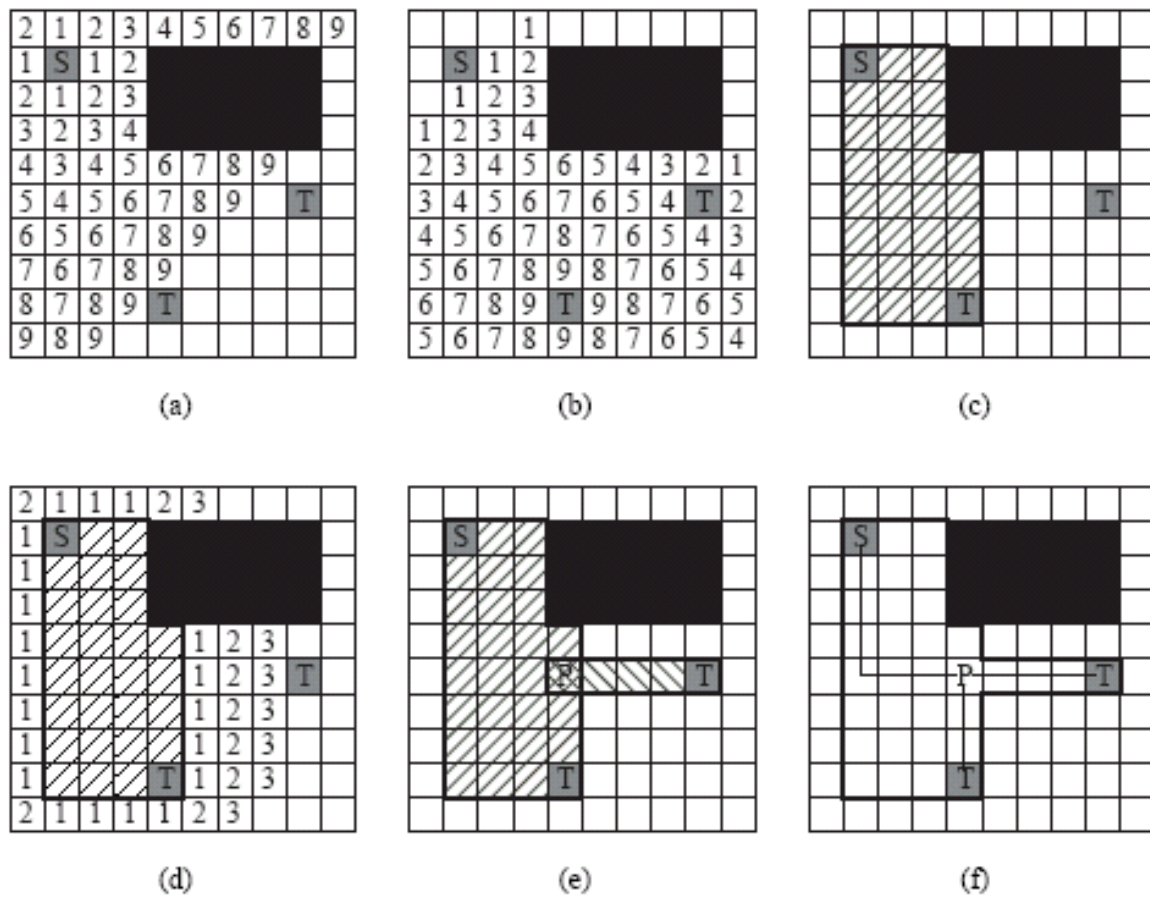


Figure III.14 Principe de base de l'algorithme pour la construction d'un arbre de Steiner quasi-minimum.

En résumé, la connexion à plusieurs terminaux est exécutée suivant les étapes suivantes :

Etape 1: Etendre l'onde de propagation depuis la source S vers la destination D.

Etape 2: Mettre toutes le cellules dans le chemin de S

Etape 3: Propager l'onde depuis toutes les cellules S

Etape 4: Continuer jusqu'à ce que toutes les cellules soient atteintes

Etape 5: Appliquer l'heuristique pour réduire le coût de d'arbre

III.6.5 UTILISATION DE PLUSIEURS NIVEAUX

Si lors de la phase de l'expansion de l'onde, on aboutit à une case occupée par une connexion déjà tracée, le programme crée un niveau de métal avec un point de connexion entre les deux niveaux (via). L'utilisation des contacts permet de poursuivre l'extension sur un autre niveau.

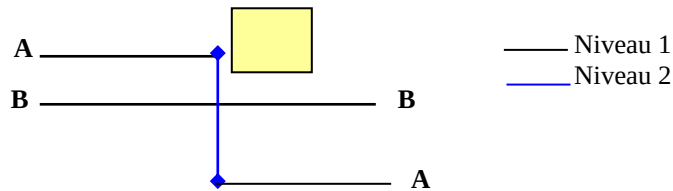


Figure III.15 Connexion à deux niveaux

Les figures 16 et 17 montrent le comportement du routeur pour des circuits à 10 et 20 connexions respectivement. Les entrées du routeur sont les coordonnées des nœuds à connecter

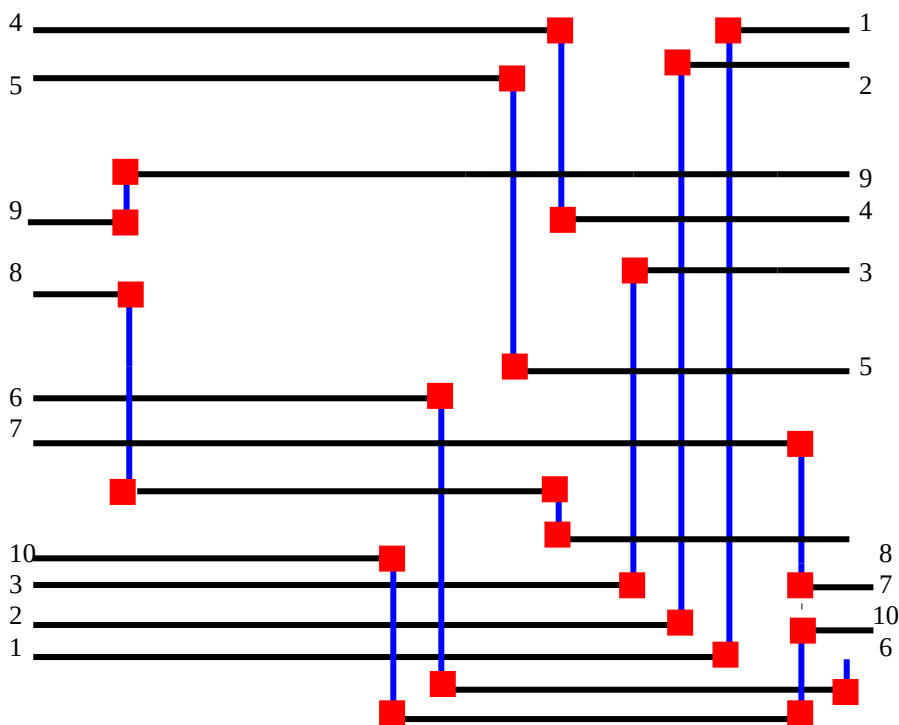


Figure III.16 Comportement du Routeur pour un circuit à 10 connexions

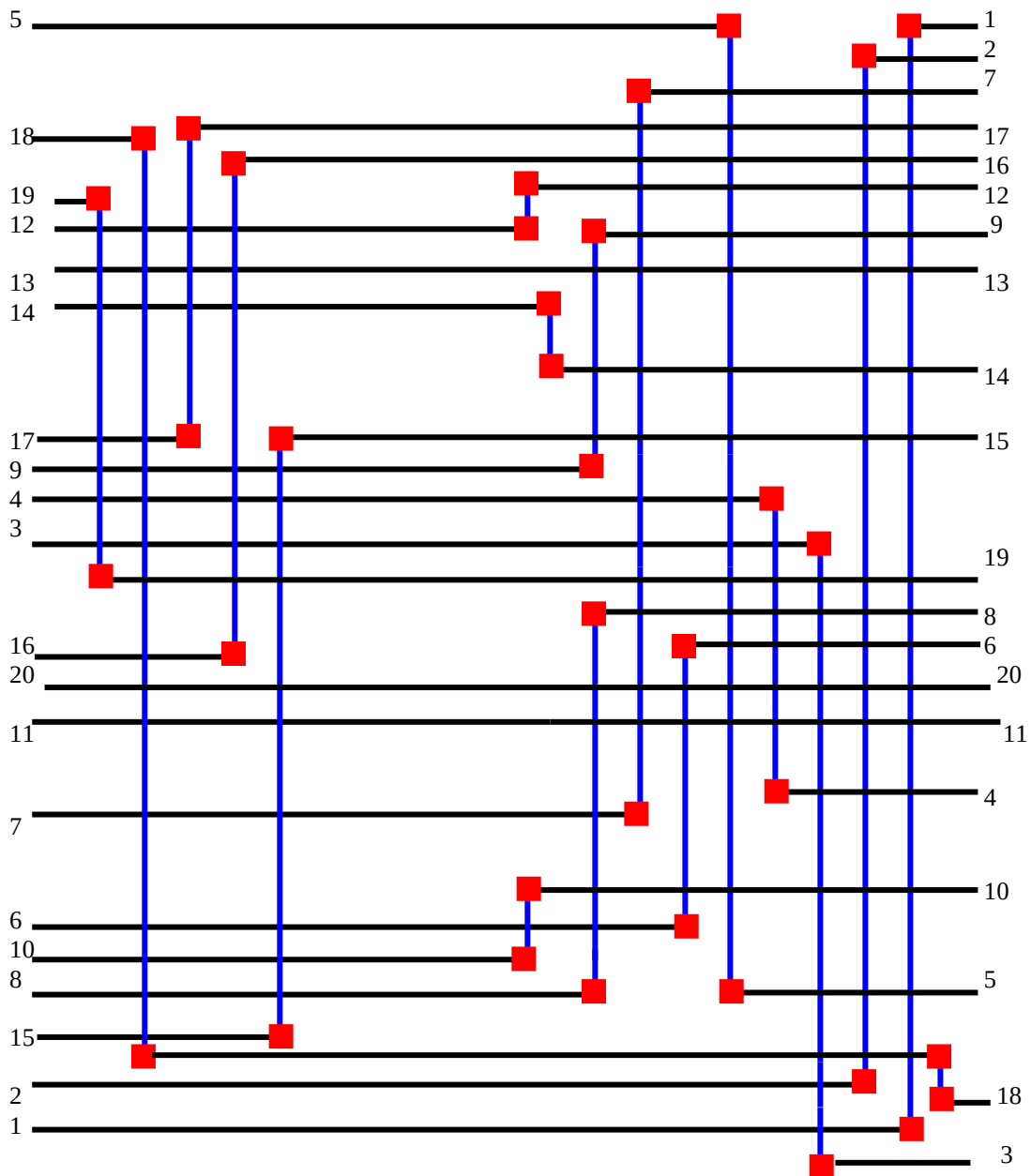


Figure III.17 Comportement du Routeur pour un circuit à 20 connexions

Le tableau III.1 donne les différents effectués sur le routeur développé.

Nombre de nœuds	Nombre de VIA
10	26
20	36
45	70
68	85
100	125

Tableau III.1 Résultats des tests

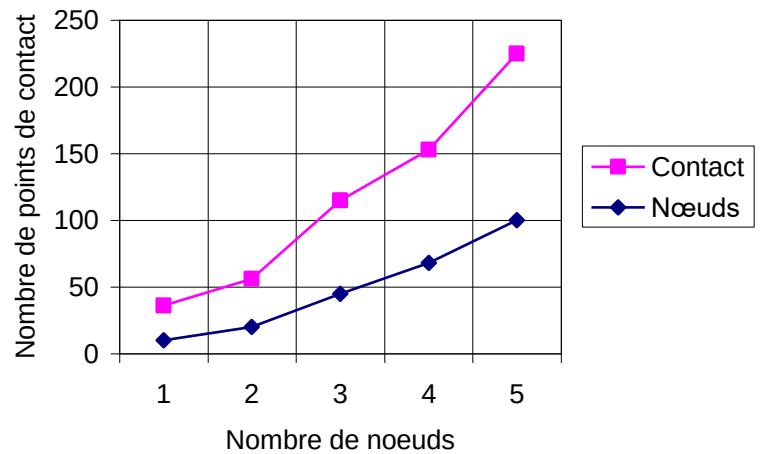


Figure III.18 Evolution des contacts par rapport aux nombres de nœuds

	Routage expansion de grille				Routage recherche en ligne	
	Lee	Soukup	Hadlock	A*	Mikami	Hightower
Temps	$O(MN)$	$O(MN)$	$O(MN)$	$O(MN)$		
Espace	$O(MN)$	$O(MN)$	$O(MN)$			
Trouve t-il le chemin s'il existe	Oui	Oui	Oui	Oui	Oui	Non
Le chemin est-il court	Oui	Non	Oui	Oui	Non	Non
Type d'expansion	Grille	Grille	Grille	Grille	Ligne	Ligne

Tableau III.2 Comparaison de l'algorithme A* avec d'autres méthodes

Dans l'approche séquentielle, nous avons besoin de donner un certain ordre à nos nœuds de connexion. Un "mauvais" ordre peut augmenter la longueur totale de connexion et peut même empêcher l'accomplissement du routage certains circuits (figure III.19).

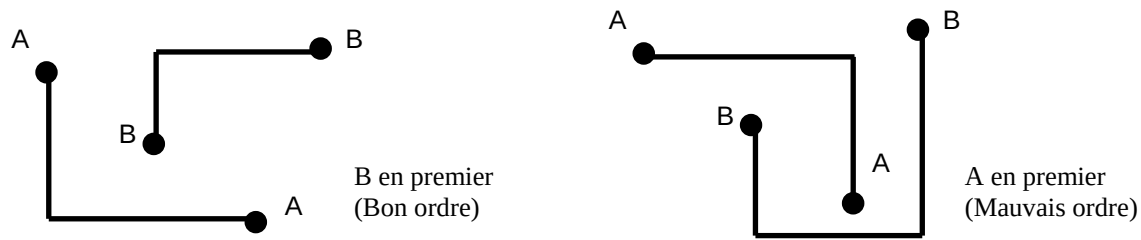


Figure III.19 Importance de l'ordre des nœuds

III.7 CONCLUSION

Au terme de ce travail, nous avons dressé un bilan des travaux effectués. Le but proposé était de développer un routeur. A cet effet, nous avons fait une étude exhaustive des algorithmes de routage. Ceci nous a permis de choisir un algorithme qui nous a permis de développer un programme de routage dont les résultats sont jugés acceptables. Pour le problème d'optimisation, nous avons remarqué que notre programme garantit de trouver la solution mais au dépend d'un autre paramètre d'optimisation : le nombre de points de contact.

En effet, nous avons remarqué que le routeur génère un nombre important de via. Nous avons pensé améliorer ce paramètre par l'optimisation multi-objective, c'est-à-dire que le programme aura à prendre en considération deux paramètres d'optimisation, la longueur de connexion et le nombre de via. Alors que, pour mieux optimiser la longueur des connexions il faut étendre à plus de deux niveaux de métal et penser à l'architecture X (autre que le modèle de MANHATTAN)[26]

RÉFÉRENCES DU CHAPITRE 3

- [1] N. Sherwani, Algorithms for VLSI Physical Design Automation, Kluwer Academic, Boston, MA, 1999.
- [2] M.S Khalid "Routing architecture and layout synthesis for multi-FPGA systems", PHD thesis 1999, Toronto.
- [3] Gary W. Clow, "A Global Routing Algorithm for General Cells", 21st Design Automation Conference, IEEE 1984.
- [4] Bence Golda et al, "Implementation of VLSI routing algorithms", IEEE 6th International Conference, on IES Opatija, 2002.
- [5] W. Choi et G. Sobelman "Hardware Rip-up Router with Concurrent Wavefront Propagation", Electronics Letter, pages 373–374, mars 1989.
- [6] C. Lee. "An Algorithm for Path Connections and its Applications". IRE Transactions on Electronic Computers, Sept. 1961.
- [7] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". Proceedings of the American Mathematical Society, 7(1) :48–50, février 1956.
- [8] H. G. Adshead, "Towards VLSI Complexity : The DA Algorithm Scaling Problem : Can Special DA Hardware Help ", Proceedings of the 19th Conference on Design Automation, pages 339–344. IEEE Press, 1982.
- [9] P. Agrawal et M. A. Breuer, "Some Theoretical Aspects of Algorithmic Routing". Proc. 14th Design Automation Conference, pages 23–31, Piscataway, NJ, USA, 1977. IEEE Press.
- [10] S. Akers, "A Modification of Lee's Path Connection Algorithm". IEEE Transactions on Electronic Computers, EC-16(1) :97–98, février 1967.
- [11] F. Rubin, "The Lee Path Connection Algorithm". IEEE Transactions on Computers, c-23(9) :907–914, septembre 1974.
- [12] J. H. Hoel, "Some Variations of Lee's Algorithm". IEEE Transactions on Computers, C-25(1) :19–24, janvier 1976.
- [13] K. Mikami et K. Tabuchi, "A Computer Program for Optimal Routing of Printed Circuit Conductors". IFIP Congress, volume 2, pages 1475–1478, 1968.
- [14] J. Soukup, "Fast Maze Router". Proc. 15th design automation conference, pages 100–102, Piscataway, NJ, USA, 1978. IEEE Press.
- [15] S. Brown, J. Rose et Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays". Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 11(5) :620–628, 1992.

- [16] D. D. Hill, "A CAD System for the Design of Field Programmable Gate Arrays". DAC '91 : Proceedings of the 28th conference on ACM/IEEE design automation, pages 187–192. ACM Press, 1991.
- [17] L. McMurchie et C. Ebling, "PathFinder : A Negotiation-Based Performance-Driven Router for FPGAs". Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 111–117. ACM, février 1995.
- [18] R. Nair, "A Simple yet Effective Technique for Global Wiring". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 6(2) :165–172, mars 1987.
- [19] D.W. Hightower, "A Solution to Line-Routing Problems on the Continuous Plane". Proc. 6th annual conference on Design Automation, pages 1–24, New York, NY, USA, 1969. ACM Press.
- [20] R. K. Korn, "An Efficient Variable-Cost Maze Router". Proceedings of the 19th conference on Design automation, pages 425–431, Piscataway, NJ, USA, 1982. IEEE Press.
- [21] F. M, A. Tabara et R. K. Brayton, "A Force-Directed Maze Router". Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD 2001), pages 404–407, 2001.
- [22] I.-L. Yen, R.M. Dubash et F. B. Bastani, "Strategies for Mapping Lee's Maze Routing Algorithm onto Parallel Architectures". Proc. Seventh International Parallel Processing Symposium, pages 672–679. IEEE, 1993.
- [23] W. Heyns, W. Sansen et H. Beke, "A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution". Proc. Seventeenth design automation conference on Design automation, pages 243–249, New York, NY, USA, 1980. ACM Press.
- [24] P. Hart, N. Nilson et B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on System Sciences and Cybernetics, SSC-4(2) :100–107, 1968.
- [25] H. AZIZI, "Développement d'un routeur automatique de cellules", Thèse de Magister, Université de SETIF, 1993
- [26] CAO électronique, " Le routage en diagonale optimise le rapport coût/ performance des puce", Revue Electronique, Mai 2006.

Application de la Logique Floue dans la Conception VLSI

VI.1 INTRODUCTION

L'application des structures de logique floue dans la (CAO) de la microélectronique améliore sensiblement la qualité des solutions de conception en fournissant aux concepteurs la flexibilité. Les aspects d'un processus de conception sont effectués par application de logique floue : utilisation de la connaissance de domaine, interprétation des incertitudes dans des données de conception, et adaptation d'algorithmes de conception.

La théorie des ensembles flous a été appliquée dans beaucoup de domaines scientifiques [1]-[6]. Des travaux montrant la possibilité d'application de la logique floue dans la CAO microélectronique ont commencé à apparaître dans la fin des années 80 et le début des années 90. Dans [7]-[10], les auteurs proposent la structure de logique floue pour la prise de décision dans la synthèse architectural et le placement dans la conception de circuits VLSI.

La majorité des tâches de conception peuvent être formulé en tant que problèmes d'optimisation combinatoires avec des objectifs et des contraintes multiples. À la différence de l'objectif simple, aucun concept de solution optimale n'est universellement admis pour l'optimisation multi-objective.

Dans beaucoup de problèmes de CAO, la recherche de solution multi-objective est exécutée hiérarchiquement. Les structures de logique floue sont efficaces en soutenant la hiérarchie multi-objective pour la prise de décision en produisant sur chaque niveau hiérarchique des valeurs de niveau pour que les critères soient employés comme valeurs d'adhésion. Une autre raison de considérer l'approche de logique floue est le traitement des incertitudes dans les données de conception.

Un argument de plus plaide en faveur de l'application de la logique floue dans la CAO, est dérivé de la nature des algorithmes utilisés pour les problèmes de conception. Il s'avère que plusieurs de ces algorithmes sont NP-complet même sous leurs formes les plus simples. La majorité des algorithmes pour la synthèse de conception sont heuristiques. L'heuristique est basée sur la connaissance acquise par l'humain par l'expérience et la compréhension des problèmes. Le langage naturel, qui est une base de logique floue, est un moyen plus commode pour exprimer une telle connaissance.

Les informations sur l'objet de conception existent sous deux formes. La première forme est l'information numérique dérivée des modèles mathématiques des objets conçus. Exemples d'une telle information sont les tensions et les courants à différents éléments du circuit, les valeurs booléennes des variables logiques, etc... La seconde forme de représentation est une forme linguistique, quand la description est formulée dans la langue normale ou artificielle.

Habituellement, les descriptions linguistiques sont données en termes flous pas seulement parce que c'est la forme la plus commune de représentation de la connaissance humaine, mais également parce que notre connaissance dans beaucoup d'aspects de la conception est floue. Le concept flou existe à plusieurs niveaux de la conception.

Ce chapitre est divisé en quatre parties. La première partie donne une brève description des fondations de la théorie des ensembles flous. Ceci inclut les définitions des opérations d'ensemble flou (ou des relations), raisonnement approximatif par des variables linguistiques et leurs valeurs, et l'optimisation multi objective par approche de logique floue.

La deuxième partie présente l'application de la logique floue aux différentes phases de la conception des circuits intégrés.

La troisième partie donne une approche de logique floue au problème de placement. Les résultats obtenus par le système de logique floue sont comparés aux résultats obtenus avec l'algorithme initial (chapitre 2) et des conclusions sont déduites de cette comparaison.

Enfin la quatrième partie traite le problème de routage avec une fonction multi-objective qui inclue, en plus du critère de la longueur des connexions, un autre critère qui représente le nombre de points de contact à minimiser.

VI.2 THEORIE DE BASE DES ENSEMBLES FLOUS

Un ensemble flou **A**, comme défini par Zadeh [2], est une classe des objets avec le grade d'appartenance. Un élément peut appartenir partiellement à un ensemble flou. C'est contraire à la théorie des ensembles ordinaires, où un élément appartient ou pas à un ensemble. Formellement, un ensemble flou est défini comme suit [2]:

Définition I : Un sous-ensemble flou **A** d'un univers du discours **X** est défini par

$$A = \{(x, \mu_A(x)) \text{ pour tout } x \in X\} \quad (\text{VI.1})$$

Où **X** est l'espace des points et $\mu_A(x)$ est la fonction d'appartenance de $x \in X$ étant un élément **de A**. En général, une fonction d'appartenance $\mu_A(\cdot)$ est tracée à partir de **X** à l'intervalle [0,1]. Si $\mu_A(x)=1$ ou 0 pour tout $x \in X$, alors l'ensemble **A** est un ensemble ordinaire. Une appartenance possible de l'ensemble flou **A** est illustrée par la figure IV.1. Sur cette figure le point x_1 appartient à l'ensemble flou **A** avec un degré d'appartenance $\mu_A(x_1)$.

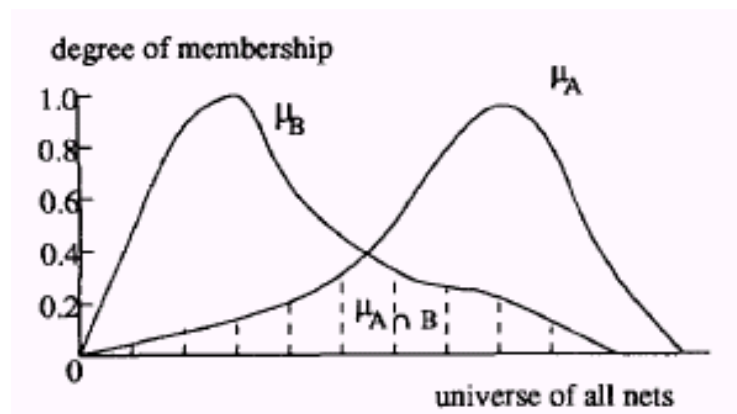


Figure VI.1 Fonction d'appartenance pour un ensemble flou $A \cap B$

Les opérations telles que l'union, l'intersection, et la complémentation, etc., sont naturellement présentées dans la théorie des ensembles flous. En raison de la nature des fonctions

d'appartenance, les résultats des opérations pour les ensembles flous sont également des ensembles flous.

$$\begin{aligned}\mu_{A \cap B}(x) &= \min(\mu_A(x), \mu_B(x)) \\ \mu_{A \cup B}(x) &= \max(\mu_A(x), \mu_B(x)) \\ \mu_{\neg A}(x) &= 1.0 - \mu_A(x)\end{aligned}\quad (VI.2)$$

$\cap$, $\cup$ et $\neg$ sont appelés ET, OU et NON respectivement. Certains opérateurs algébriques peuvent également être employés pour définir ces opérations d'ensemble flou. Par exemple, le produit algébrique de **A** et **B** est dénoté par **AB**, avec la fonction d'appartenance il est défini par $\mu_{AB} = \mu_A(x) \cdot \mu_B(x)$; la somme algébrique est dénotée par **A + B** avec la fonction d'appartenance est définie par

$$\mu_{A+B} = \mu_A(x) + \mu_B(x) \quad (VI.3)$$

La région hachurée dans la figure VI.1, montre la fonction d'appartenance de l'ensemble flou $A \cap B$, qui résulte de l'intersection des ensembles flous **A** et **B**. Ici $\mu_A(.)$ et $\mu_B(.)$ sont les fonctions d'appartenance pour les ensembles flous A et B, respectivement.

La grandeur $\mu_A(x)$ définit le degré d'appartenance de l'élément x à l'ensemble A.

$$\begin{aligned}\mu_A : U &\rightarrow [0, 1] \\ x &\rightarrow \mu_A(x) \\ A &= \{ (x, \mu_A(x)) \mid x \in U \} \\ \text{supp}(A) &= \{ x \in U \mid \mu_A(x) \geq 0 \} \\ \text{noy}(A) &= \{ x \in U \mid \mu_A(x) = 1 \}\end{aligned}\quad (VI.4)$$

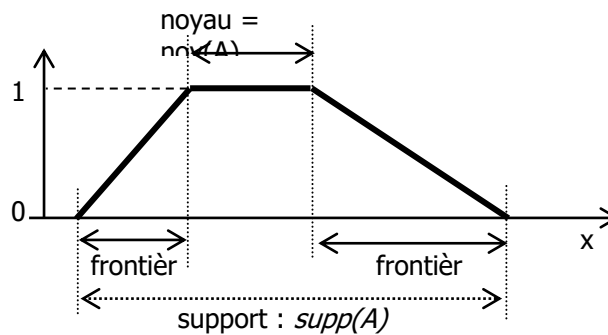


Figure VI.2 Ensemble flou

L'ensemble flou vide est noté f , il est défini par :

$$\mu_\phi(x) = 0, \forall x \in U \quad (VI.5)$$

Le plus grand ensemble flou sur U est noté 1_U , il est défini par :

$$\mu_{1_U}(x) = 1, \forall x \in U \quad (\text{VI.6})$$

Les fonctions d'appartenance peuvent avoir diverses formes selon leur définition, elles peuvent avoir une fonction triangulaire, trapézoïdale, Gaussienne, ou Sigmoides...

VI.2.1 OPERATIONS SUR LES ENSEMBLES FLOUS

Comme dans le cas des ensembles classiques, les opérations logiques d'union (ou), d'intersection (et) et de complémententation (non) peuvent être appliquées aux ensembles flous. Leurs définitions ne sont pas uniques.

Les définitions les plus souvent rencontrées sont : le max et le min (Mandani), le produit et la somme moins le produit (Sugeno).

Mandani: $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ et $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

Sugeno: $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$ et $\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$

Dans les deux cas : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

pour $x \in U$

Exemple dans le cas Mandani

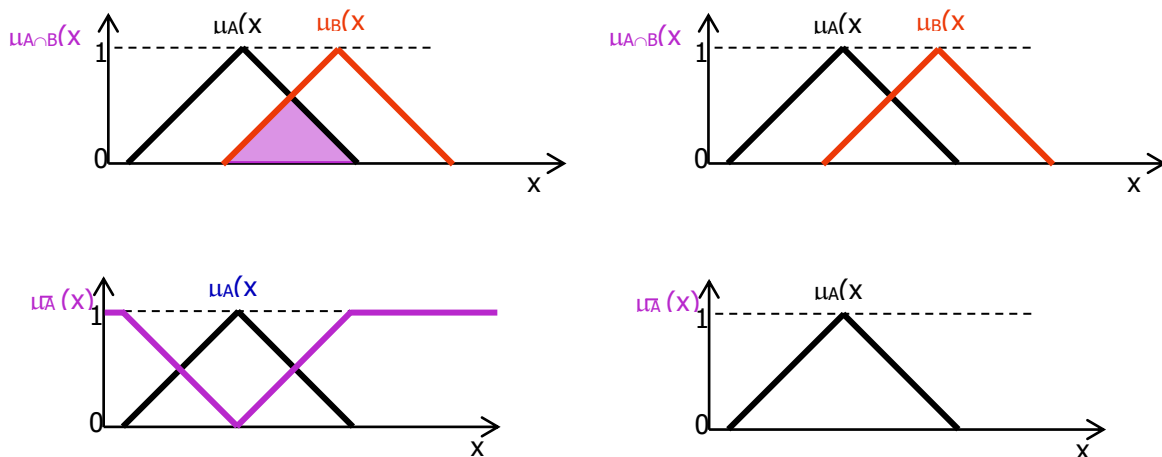


Figure VI.3 Opérations floues dans le cas Mandani

VI.2.2 PROPRIETES DES ENSEMBLES FLOUS

Comme dans le cas des ensembles «classiques», les ensembles flous possèdent certaines propriétés (équation VI.7).

Commutativité : $A \cup B = B \cup A$, $A \cap B = B \cap A$

Associativité : $A \cup (B \cap C) = (A \cup B) \cap C$, $A \cap (B \cup C) = (A \cap B) \cup C$

Distributivité : $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (VI.7)

Idempotence : $A \cup A = A$, $A \cap A = A$

Identité : $A \cup \phi = A$, $A \cup 1_U = 1_U$, $A \cap \phi = \phi$, $A \cap 1_U = A$

Les deux propriétés suivantes ne sont pas «classiques». L'intersection d'un ensemble flou et de son complément n'est pas vide :

Loi de contradiction: $A \cap \bar{A} \neq \phi$ (VI.8)

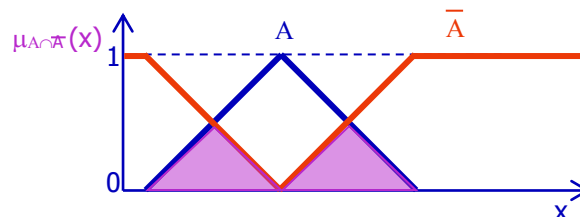


Figure VI.4 Loi de contradiction

L'union d'un ensemble flou et de son complément ne donne pas l'univers du discours

Loi du "excluded middle": $A \cup \bar{A} \neq 1_U$ (VI.9)

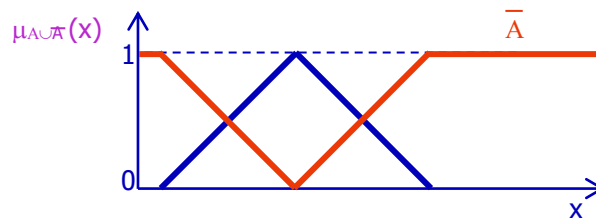


Figure VI.5 Union d'un ensemble flou et de son complément

VI.2.3 LES RELATIONS FLOUES

Se sont des structures qui représentent l'absence ou la présence d'une corrélation ou d'une interaction entre les éléments de divers ensembles.

Les degrés de relation entre des éléments de deux ou plusieurs ensembles flous peuvent prendre un nombre infini de valeurs entre les deux extrêmes «reliés totalement» et «non reliés».

a) Relation binaire sur un ensemble classique :

Soient X_1 et X_2 deux ensembles, $R \subset X_1 \times X_2$ est appelée relation binaire. Si $X_1 = X_2 = \mathfrak{R}$, alors la fonction caractéristique de R est définie par :

$$\chi_R(u, v) = \begin{cases} 1 & \text{si } (u, v) \in R \\ 0 & \text{autrement} \end{cases} \quad (\text{VI.10})$$

Exemple :

$$\begin{aligned} (u, v) \in R &\Leftrightarrow u \in [a, b] \text{ et } v \in [0, c] \\ \chi_R(u, v) &= \begin{cases} 1 & \text{si } (u, v) \in [a, b] \times [0, c] \\ 0 & \text{autrement} \end{cases} \end{aligned} \quad (\text{VI.11})$$

b) Relation floue

Une relation floue sur deux univers U et V est un ensemble flou :

$$\begin{aligned} R: U \times V &\rightarrow [0, 1] \\ (u, v) &\rightarrow \mu_R(u, v) \end{aligned} \quad (\text{VI.12})$$

On ne parle plus de fonction caractéristique mais de fonction d'appartenance. La relation floue est binaire si $U=V$

Exemple :

Soit l'univers $U = \{1, 2, 3\}$, la relation R « est approximativement égal à » peut-être définie par :

$$\begin{aligned} R: \{1, 2, 3\} \times \{1, 2, 3\} &\rightarrow [0, 1] \\ (u, v) &\rightarrow \mu_R(u, v) = \begin{cases} 1 & \text{si } u = v \\ 0,8 & \text{si } |u - v| = 1 \\ 0,3 & \text{si } |u - v| = 2 \end{cases} \end{aligned} \quad (\text{VI.13})$$

Notation matricielle

$$\begin{pmatrix} & 1 & 2 & 3 \\ 1 & 1 & 0.3 & 0.8 \\ 2 & 0.8 & 1 & 0.8 \\ 3 & 0.3 & 0.8 & 1 \end{pmatrix}$$

On définit l'union et l'intersection de deux relations floues par :

$$\text{Soient } R: U \times V \rightarrow [0,1] \quad (u,v) \rightarrow \mu_R(u,v) \quad \text{et} \quad S: U \times V \rightarrow [0,1] \quad (u,v) \rightarrow \mu_S(u,v) \quad (\text{VI.14})$$

Alors au sens Mandani:

$$\mu_{R \cup S}(u,v) = \max(\mu_R(u,v), \mu_S(u,v)) \quad (\text{VI.15})$$

$$\text{et } \mu_{R \cap S}(u,v) = \min(\mu_R(u,v), \mu_S(u,v))$$

Alors au sens Sugeno:

$$\mu_{R \cup S}(u,v) = \mu_R(u,v) + \mu_S(u,v) - \mu_R(u,v) \cdot \mu_S(u,v) \quad (\text{VI.16})$$

$$\text{et } \mu_{R \cap S}(u,v) = \mu_R(u,v) \cdot \mu_S(u,v)$$

VI.2.4 LE RAISONNEMENT APPROXIMATIF PAR DES VARIABLES LINGUISTIQUES

La logique est définie comme une science du raisonnement correct. Dans le domaine de la résolution des problèmes, l'information imprécise nous empêche toujours de faire le raisonnement précis. Le raisonnement de l'être humain est rarement exact. Par conséquent, la logique binaire traditionnelle basée sur la théorie booléenne des ensembles ordinaires n'est pas appropriée pour modéliser le raisonnement humain.

La logique floue est définie comme "art du raisonnement flou". A chaque fois que le mot "art" est mentionné, il fait participer des jugements plus ou moins subjectifs ainsi que des études empiriques au sujet des domaines fondamentaux. Le raisonnement fait sur un domaine imprécis s'appelle le raisonnement flou. Le concept de la logique floue peut mieux être compris si un "degré d'adhésion" dans un ensemble flou est interprété comme "degré de vérité".

Par exemple, (1) peut être interprété comme étant l'élément $x \in X$ dans l'ensemble flou A et dans l'ensemble flou B avec un degré de vérité $\mu_{A \cap B}(x)$. C'est-à-dire, $A \cap B = \{(x, \mu_{A \cap B}(x)) \text{ pour tout } x \in X\}$. Le raisonnement de l'être humain utilise souvent des termes linguistiques. La variable linguistique est une variable dont les valeurs sont des mots ou des phrases dans une langue normale ou artificielle [2]. La définition formelle [4] est présentée ci-dessous.

Définition II : Une variable linguistique est caractérisée par un quintuple $(R, T(\Omega), X, G, M)$ où Ω est le nom de la variable linguistique; $T(\Omega)$ est l'ensemble des termes de Ω , c.-à-d., la collection de ses valeurs linguistiques; G est une règle syntactique qui produit des limites dans $T(\Omega)$; et M est une règle sémantique qui associe à chaque valeur linguistique sa signification $M(\omega)$, où $M(\omega)$ dénote un sous-ensemble flou X et $\omega \in T(\Omega)$.

VI.3 APPLICATION DE LOGIQUE FLOUE AUX OUTILS DE LA CAO

Des facteurs humains impliqués dans le choix et l'évaluation de conception font que l'application de logique floue est plus appropriée. Ceci n'exclut pas une application de logique floue pour certains types d'analyse quand le manque de "netteté" dans des données d'entrée devrait être pris en compte.

Le critère final pour l'efficacité de la logique floue dans la conception est un critère pratique, c.-à-d., si la qualité des solutions obtenues par des outils de logique floue est meilleure que celles produites par les outils qui n'utilisent pas la logique floue.

La conception des systèmes numériques complexes est un processus à plusieurs étages, ce qui implique l'application d'une variété d'algorithmes. Dans ce paragraphe, nous présentons la description d'une classification utile pour décrire la logique floue appliquée dans la CAO.

Par leur niveau dans la hiérarchie de conception :

- 1) conception physique;
- 2) conception logique;
- 3) conception architecturale.

Par des aspects de conception adressés par logique floue :

- 1) prise de décision multi-objective;
- 2) adaptation des algorithmes de conception;
- 3) incertitude dans des données de conception et leur interprétation.

Par des types d'algorithmes utilisés pour trouver les solutions:

- 1) constructif;
- 2) itératif;
- 3) combinaison de constructif et d'itératif.

Nous emploierons cette classification dans les descriptions suivantes.

VI.3.1 APPLICATION DE LOGIQUE FLOUE DANS LES ALGORITHMES CONSTRUCTIFS

Beaucoup de problèmes de conception physique sont résolus par des algorithmes constructifs, appelé également "l'augmentation successive". Ces algorithmes produisent une solution finale en ajoutant successivement un ou plusieurs nouveaux éléments à la fois à la solution partielle. Des critères multiples sont employés pour prendre des décisions dans chaque étape. Par exemple, avec un placeur constructif flou de cellules standard, chaque nouvelle cellule présentée au placement partiel est caractérisée par trois variables linguistiques : *puissance*, *surface du placement partiel*, et *longueur d'interconnexion* [17]. Chaque variable linguistique est caractérisée par une valeur linguistique; par exemple, *bonne satisfaisante*, ou *mauvaise*.

IV.3.2 APPLICATION DE LOGIQUE FLOUE DANS LES ALGORITHMES ITERATIFS

À la différence des algorithmes constructifs pour la conception physique, qui produise une solution seulement à la fin du processus de conception, les algorithmes itératifs fonctionnent avec des solutions de conception définies à chaque itération.

La plupart des algorithmes utilisés dans la conception physique sont itératif de nature (algorithmes génétiques, recuit simulés). Une valeur de la fonction objective est employée pour comparer les résultats des itérations consécutives et pour choisir une solution basée sur la valeur (minimale) maximale de la fonction objective.

L'équilibrage des différents objectifs par la fonction de poids est difficile parce qu'il exige la comparaison des critères contradictoires. La logique floue est un outil commode pour résoudre ce problème parce qu'il permet de tracer des valeurs des différents critères dans des valeurs linguistiques, qui caractérisent le niveau de satisfaction du concepteur avec la valeur numérique des objectifs et à fonctionner au-dessus des valeurs à partir de l'intervalle [0,1] défini par fonctions d'adhésion pour chacun objectif.

Par exemple, la fonction objective pour le recuit simulé, le modèle du problème de placement [18], inclut la longueur d'interconnexion, une pénalité de recouvrement, et une pénalité pour longueur de dépassement de limites. La fonction objective est définie par la formule :

$$F(x) = \sum_i^3 \omega_i f_i(x) \quad (\text{VI.17})$$

Où ω_i représente le coefficient de poids pour la fonction objective f_i et x est le point dans l'espace de solution du problème de placement.

En fait, la fonction objective n'est pas une grandeur scalaire, mais une fonction vecteur :

$$\bar{F}(x) = [f_1(x), f_2(x), f_3(x)] \quad (\text{VI.18})$$

Où $f_1(x)$ est la longueur de connexion, $f_2(x)$ est la surface de recouvrement, et $f_3(x)$ est le délai.

Pour obtenir une définition de logique floue de la fonction objective pour un algorithme itératif, trois variables linguistiques, *la longueur de connexion*, *recouvrement*, et *délai*, sont définies pour ces fonctions. Un ensemble de valeurs linguistiques est défini pour chaque composant de la fonction objective. Dans le cas du placement, seulement une valeur linguistique est définie pour chaque variable ; c'est-à-dire, *petite longueur*, *petit recouvrement*, et *bon délai*. Ces valeurs linguistiques caractérisent un degré de la satisfaction du concepteur avec des valeurs des objectifs $f_i(x)$, ($i=1, 2, 3$). Ces degrés de satisfaction sont décrit par des fonctions d'appartenance $\mu_i(.)$ sur les ensembles flous des valeurs linguistiques.

L'évaluation pour le vecteur de la solution correspond à l'évaluation la de la règle suivante :

R.0 si une solution x a *la petite longueur* et *le moins de recouvrement* et *la bonne propagation*, alors x est *une bonne solution*.

Les opérateurs compensatoires de logique floue [5] sont employés dans les opérations d'intersection et d'union dans des applications de conception.

Selon ces définitions, adhésion dans l'intersection des ensembles flous est définie par :

$$\begin{aligned}
 F(x) &= \mu(\text{bon délai} \cap \text{petit recouvrement} \cap \text{petite connexion})(x) \\
 &= \mu(\text{bonne solution})(x) \\
 &= \frac{\mu_1(x) \cdot \mu_2(x) \cdot \mu_3(x)}{\max\{\mu_1(x), \mu_2(x), \mu_3(x), \alpha\}} \quad (\text{VI.19})
 \end{aligned}$$

$$\alpha \in [0,1]$$

Ici, α est une valeur définie par l'utilisateur; puis une valeur de la fonction vectrice $\bar{\mu}(x)$ est évaluée par une valeur de fonction scalaire $\mu(\text{bonne solution})(x)$ et la solution optimale est trouvée:

$$F_2(x) = \max \mu(\text{bonne solution})(x) \quad (\text{VI.20})$$

IV.3.3 ALGORITHMES ET OUTILS DE LOGIQUE FLOUE POUR LA SYNTHESE LOGIQUE

Il est clair que l'application de la logique floue dans la CAO ne devrait pas être limitée à la conception physique. De nombreux problèmes dans la synthèse de logique impliquent la multiplicité de buts et de contraintes et la connaissance de domaine sous la forme linguistique. Un domaine d'application prometteur de logique floue est (FPGA) [13] [14].

VI.3.4 ALGORITHMES ET OUTILS DE LOGIQUE FLOUE POUR LA SYNTHESE ARCHITECTURALE

Le niveau de synthèse architectural présente de meilleures opportunités pour l'application de la logique floue parce que la connaissance en ce niveau de conception est très souvent exprimée en

formes linguistiques. Une telle connaissance devrait être combinée avec les valeurs cartésiennes pour que les critères choisis obtiennent de bonnes solutions de conception [10].

VI.4 APPLICATION

VI.4.1 APPLICATION DE LA LOGIQUE FLOUE POUR LE PROBLEME DE PLACEMENT

Le placement est l'une des étapes les plus importantes dans la conception physique VLSI. Beaucoup d'algorithmes de placement ont été présentés ces dernières années. Tous ces algorithmes ne considèrent pas explicitement deux aspects importants du problème de placement : la présence d'objectifs multiples et contradictoires et l'utilisation de la connaissance experte dans la prise de décision (figure VI.6). Ces aspects du problème de placement peuvent être traités par la technique basée sur la théorie des ensembles flous. Des tentatives pour l'application de la théorie des ensembles flous aux problèmes de la conception physique ont été utilisées. Dans [11], la matrice de connexion a été modifiée et alors la théorie des ensembles flous a été employée pour réaliser le placement.

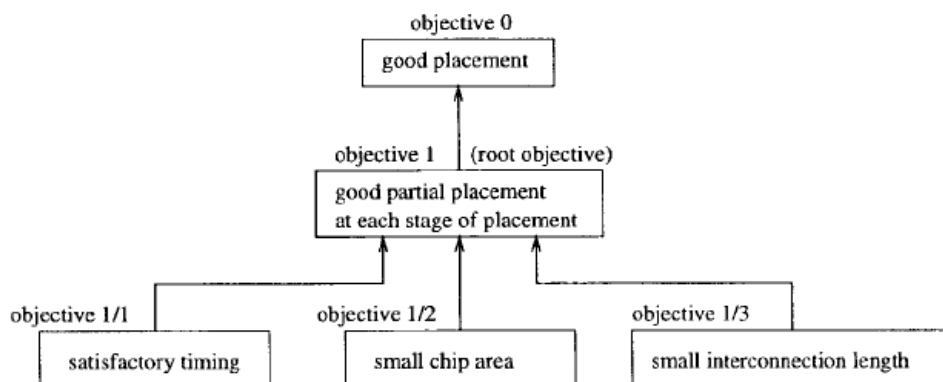


Figure VI.6 Objectif multiple pour le placement de cellules standard.

Dans cette partie, nous allons appliquer le concept de logique floue au problème de placement, en modifiant la matrice de connexion [15].

IV.4.2 ADAPTATION DE LA MATRICE DE CONNEXION

Dans la pratique, des objectifs plus simples et plus précis sont employés comme par exemple la longueur totale du routage, le nombre maximum des lignes coupées et la densité maximale. Trouver une solution optimale à un objectif donné est considérée comme un problème NP-

complete (le temps de calcul se développe exponentiellement avec la taille de problème). Par conséquent des algorithmes heuristiques sont employés.

Il existe une variété de règles pour relier ensemble les modules qui sont souvent dictés par la technologie spécifique et conçoivent des modèles utilisés pour la fabrication du circuit.

Les cellules et leurs interconnexions dans un problème de placement sont données par **une matrice de connexion**. La figure VI.7 donne un exemple de matrice de connexion d'un problème de placement avec 9 cellules {0, 1, 2, 3, 4, 5, 6, 7, 8}, chaque élément de la matrice représente la connectivité entre deux éléments dans l'ensemble, une valeur nulle indique qu'il n'y a pas de connexions entre les cellules correspondantes.

	0	1	2	3	4	5	6	7	8
0	0	5	0	5	2	0	0	0	0
1	5	0	5	0	5	0	0	0	0
2	0	5	0	0	2	5	0	0	0
3	5	0	0	0	5	0	5	0	0
4	2	5	2	5	0	5	2	5	2
5	0	0	5	0	5	0	0	0	5
6	0	0	0	5	2	0	0	5	0
7	0	0	0	0	5	0	5	0	5
8	0	0	0	0	2	5	0	5	0

Figure VI.7 Matrice de connexions

VI.4.3 LA RELATION FLOUE ET LA MATRICE DE CONNEXION

Une relation dans le contexte de la théorie des ensembles conventionnels (ordinaires) représente l'absence ou la présence de l'interaction ou de l'association entre les éléments de deux ensembles ou plusieurs ensembles. Cependant une relation floue prolonge ce concept de relation cartésienne pour laisser divers degrés entre les éléments des ensembles, et elle indique ces degrés d'association par ses catégories d'adhésion comme des fonctions d'appartenance sont représentées dans un ensemble flou.

Quand la relation floue est définie sur deux ensembles X et Y, c.-à-d. $R(X, Y)$ est appelée une relation binaire et peut être représentée par une matrice de connexion dont les éléments sont des catégories d'adhésion de la relation. Il suit qu'une relation binaire floue $R(X, X)$ défini sur un ensemble simple X montre les degrés d'association entre les éléments du même ensemble X.

Afin de pouvoir appliquer la théorie des ensembles flous à un problème de placement, nous avons convenablement modifié la matrice associée de raccordement pour représenter une relation floue sur l'ensemble de modules B à placer. Les éléments modifiés de cette matrice sont alors les catégories d'adhésion de la relation floue.

Nous avons développé l'algorithme en utilisant la matrice de connexion d'un problème de placement comme relation floue. L'algorithme doit maintenir le même choix des règles de positionnement que dans des algorithmes conventionnels de placement, mais exploiter la propriété de la matrice de connexion modifiée. Cet algorithme flou a généralement comme conséquence une meilleure configuration finale de placement qu'un algorithme conventionnel.

VI.4.4 MODIFICATION DE LA MATRICE DE CONNEXION [11]

Ce nouvel algorithme dépend de la matrice de connexion C. Celle-ci ne doit pas représenter forcément les valeurs "vraies" de connectivité entre les cellules. Si la matrice de connexion $[c_{ij}]$ possède un élément $c_{ij} > 0$, $c_{jk} > 0$ et $c_{ik} = 0$, ceci implique que les cellules i et j sont reliées entre elles, de même que les cellules j et k, par contre aucune connexion n'existe entre les cellules i et k. En réalité, il existe une relation implicite entre les cellules i et k que la première matrice de connexion ne met pas en évidence.

En effet, en employant la théorie des ensembles flous il est possible de déduire des valeurs "vraies" de la matrice de connexion initiale de nouvelles valeurs et ainsi construire la nouvelle matrice de connexion modifiée. Cette matrice établit le lien implicite entre des cellules qui ne sont pas directement connectées.

Nous considérons le problème de placement, dans lequel la distance entre deux cellules i et j est défini par le terme d_{ij} , et la distance entre les cellules j et k est définie par d_{jk} . Si la distance entre les cellules et définie par la distance de Manhattan, alors il est évident que la distance entre la cellule i et la cellule k est égale à $d_{ij} + d_{jk}$ ou $|d_{ij} - d_{jk}|$.

$$\text{Avec } d_{ik} \leq d_{ij} + d_{jk}$$

Nous assumons que la connectivité entre deux cellules est inversement proportionnel à la distance entre elles, alors :

$$d_{ij} = \alpha. 1/c_{ij} \quad \text{d'où } 1/c_{ik} \leq 1/c_{ij} + 1/c_{jk}$$

$$\text{Ou } c_{ik} \geq (c_{ij} * c_{jk}) / (c_{ij} + c_{jk}) \quad (\text{VI.21})$$

La relation (VI.21) montre que même si aucune connexion n'existe entre les cellules i et k, la connectivité entre elles est supérieure au rapport $(c_{ij} * c_{jk}) / (c_{ij} + c_{jk})$.

Soit $P(i) = \sum_j C_{ij}$ et $P_{\max} = \text{MAX}(P(i))$, alors la forme de

$$c_{ik} / P_{\max} \geq (c_{ij} * c_{jk}) / ((c_{ij} + c_{jk}) * P_{\max}) \quad (\text{VI.22})$$

$$\text{On déduit } c_{ik} / P_{\max} \geq \text{MAX}(c_{ij} / P_{\max} * c_{jk} / P_{\max}) \quad (\text{IV.23})$$

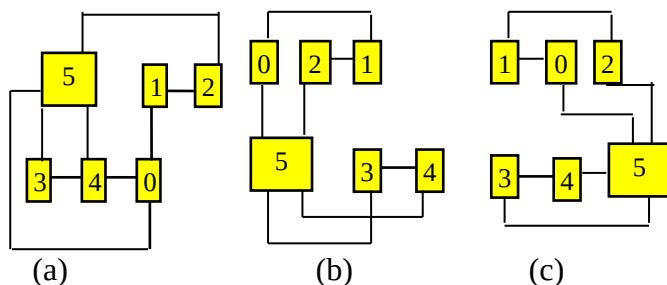
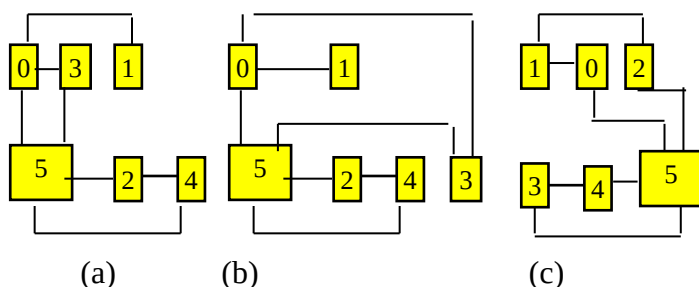
Alors, on définit $C' = C / P_{\max}$ qui est une matrice équivalente à C dans la mesure où elle représente la connectivité entre les cellules. Tant que $c_{ij} \leq 1$, la matrice C' peut être considérée comme une relation floue.

Les interconnexions entre cellules sont données par la matrice de connexions suivante (tableau 1) :

Séquence d'entrée	Matrice de connexions	
	Matrice de Connexion initiale	Matrice de Connexion modifiée

4 x 4	28	4 x 4	20	4 x 4
6 x 6	27	5 x 4	16	5 x 4
8 x 8	24	4 x 4	16	4 x 4

Tableau VI.2 Résultats du circuit de la figure 9

Figure VI.12 Placement conventionnel du circuit de la figure 8 avec différents réseaux
(a) 4 x 4 (b) 6 x 6 (c) 8 x 8Figure VI.13 Placement flou de la figure 8 avec différents réseaux
(a) 4 x 4 (b) 6 x 6 (c) 8 x 8

Réseau	Placement flou		Placement conventionnel	
	Itération n	Densité	Itération	Densité
10 x 10	65	6 x 10	41	10 x 8
12 x 12	65	6 x 10	41	10 x 8
14 x 14	65	6 x 10	41	10 x 8
16 x 16	65	6 x 10	41	10 x 8

Tableau VI.3 Résultats du circuit de la figure 10

Réseau	Placement Flou		Placement conventionnel	
	Itération n	Densité	Itération	Densité
12 x 12	42	6 x 12	38	8 x 11
14 x 14	38	6 x 12	38	8 x 11
16 x 16	38	6 x 12	38	8 x 11

Tableau VI.4 Résultats du circuit de la figure 11

VI.4.6 CONCLUSION

La logique floue, a été avec succès, appliquée à résoudre des problèmes du placement de cellules VLSI. On a montré qu'elle peut être facilement employée pour la matrice de connexion modifiée qui fait mieux apparaître les liaisons indirectes entre les cellules. Les résultats obtenus démontrent l'utilité de cette théorie quant à l'amélioration du placement.

En perspectives et pour exploiter la faculté de la théorie des ensembles flous à résoudre les problèmes avec plusieurs critères d'optimisation, nous proposons d'introduire un autre critère, en plus de la longueur des connexions, la taille du circuit ; en effet, dans l'algorithme que nous avons développé (chapitre 2), nous avons défini la fonction objective par la relation :

$$\sum_{k \in K} \sum_{l \in K} d_{kl} \cdot cm_{lk} \quad (IV.24)$$

Dans le but de combiner deux paramètres d'optimisation et une contrainte, la règle suivante est suggérée : une solution avec une longueur de connexion acceptable et une surface du circuit petite alors c'est une bonne solution.

Alors la règle ci-dessus peut être interprétée de la façon suivante :

$$\mu(x) = \min (\mu_l(x), \mu_t(x)) \quad (IV.25)$$

où $\mu(x)$ est un élément de la solution x de l'ensemble des solutions acceptables, $\mu_l(x)$ et $\mu_t(x)$ appartiennent respectivement aux ensembles des solutions longueur de connexion acceptable et la taille du circuit petite.

Une autre perspective est à considérer, pour mieux voir les performances de l'algorithme, est de faire des tests poussés avec des circuits de références (benchmarks) qui permettent une meilleure évaluation de l'algorithme.

VI.5 APPLICATION DE LA LOGIQUE FLOUE AU PROBLEME DE ROUTAGE

Dans l'algorithme que nous avons développé (chapitre 3), nous avons défini la fonction objective par la distance de Manhattan séparant deux ou plusieurs nœuds. Dans le but de combiner deux paramètres d'optimisation et une contrainte, la règle suivante est suggérée :

Règle: une solution avec une longueur de connexion acceptable et un nombre de points de connexion (via) réduit.

Alors la règle ci-dessus peut être interprétée de la façon suivante :

$$\mu(x) = \min(\mu_l(x), \mu_v(x)) \quad (\text{IV.5})$$

Où $\mu(x)$ est un élément de la solution x de l'ensemble des solutions acceptables, $\mu_l(x)$ et $\mu_v(x)$ appartiennent respectivement aux ensembles des solutions longueur de connexion acceptable et le nombre de points de contact réduits.

Nous avons remarqué que notre algorithme favorise l'expansion sur le premier niveau, et de ce fait il a tendance à créer à chaque fois un point de contact pour rejoindre le premier niveau. L'algorithme ainsi développé utilise un nombre important de points de contact ce qui le rend inadéquat et coûteux en terme économique. L'addition d'une deuxième contrainte (nombre de contacts), en plus de la longueur de connexion, dans la fonction objective permet de réduire la création de ces points de contact.

La figure IV.14 Montre un exemple de réalisation où, nous remarquons cette tendance à créer les "via" à chaque fois que l'occasion se présente (pointillée figure IV.14).

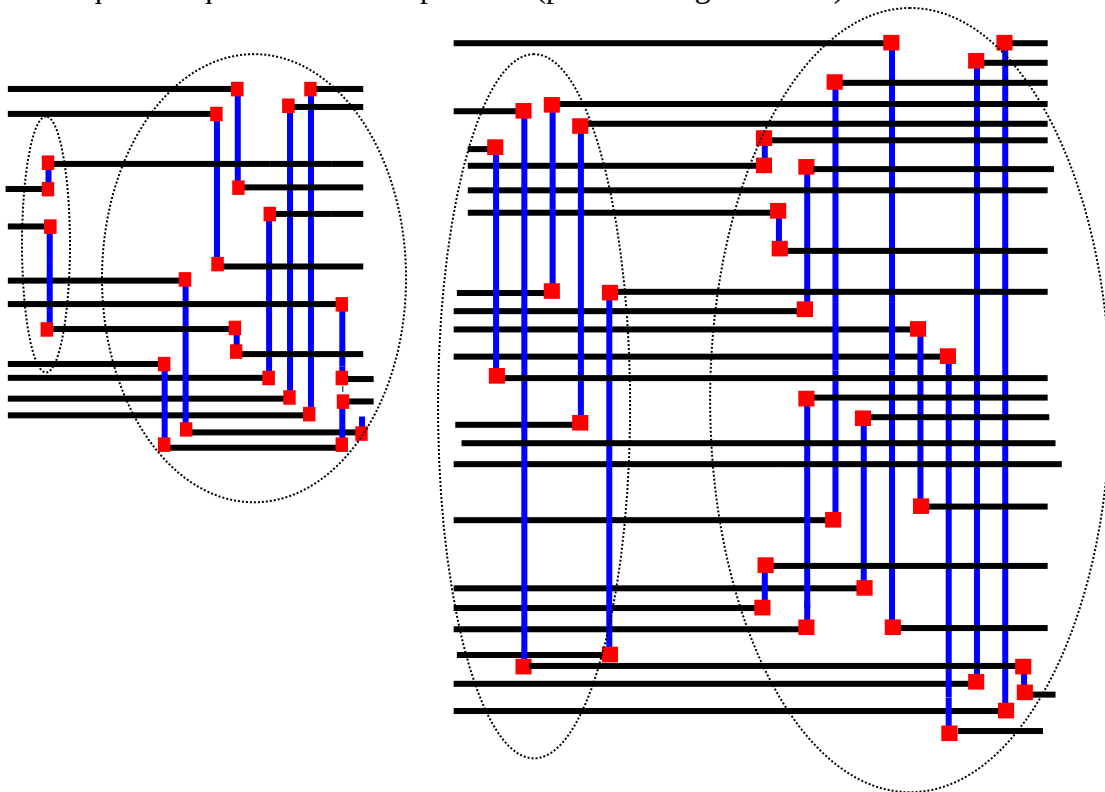


Figure III.14 Comportement du Routeur avec la fonction coût=longueur de connexion

L'incorporation d'un deuxième critère d'optimisation dans le fonction objective permet, à priori de réduire le nombre de contact. Seulement, et après avoir testé l'algorithme, nous avons remarqué qu'il n'y avait pas une nette amélioration de la qualité de routage. En effet l'algorithme favorise l'expansion de l'onde en direction du nœud cible et tout changement de direction se traduit par la création d'un niveau métal.

VI.6 CONCLUSION

Ce chapitre présente la caractérisation, la justification, et la classification des outils basés sur la logique floue pour la CAO des circuits intégrés. Il est démontré que l'application de la logique floue est salubre dans tous les niveaux hiérarchiques du processus de conception. Il laisse fusionner les techniques numériques traditionnelles avec l'information de domaine disponible sous la forme linguistique pour améliorer la qualité des solutions de conception.

Des structures de logique floue peuvent être employées dans des procédures constructives et des procédures itératives. L'application de celle-ci au problème de placement a montré une certaine amélioration par rapport au placement conventionnel. Ceci est rendu possible par le changement de la matrice de connexion entre cellules. Par contre, pour le problème de routage l'incorporation d'un autre critère d'optimisation dans la fonction objective, à savoir le nombre de points de contact, n'a pas amélioré la qualité de routage.

REFERENCES DU CHAPITRE 4

- [1] M. Pirlot, "General local search methods," *Eur. J. Operat. Res.*, vol. 92, pp. 493–511, 1996.
- [2] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—I," *Inform. Sci.*, vol. 8, pp. 199–249, 1975.
- [3] H. J. Zimmermann, "Fuzzy Set Theory and Its Applications", 2nd ed. Norwell, MA: Kluwer, 1991.
- [4] Norwell, MA: Kluwer, Fuzzy Sets, Decision Making, and Expert Systems, 1987.
- [5] Fuzzy and neural control, Robert Babuska Control Engineering Laboratory Faculty of Information Technology and Systems Delft University of Technology, Delft, the Netherlands
- [6] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-3, No.1, 1973, Proceedings of -Fuzz 1992 and 1993 conferences.
- [7] T. Gao, P. M. Vaidya, and C. L. Liu, "A performance driven macro-cell placement algorithm," in *Proc. ACM/IEEE 29th Design Automat. Conf.*, Anaheim, CA, June 1992, pp. 147–152.
- [8] W. Donath, R. Norman, B. Agrawal, S. Bello, S. Han, J Kurtzberg, P. Lowy, and R. McMillan, "Timing-driven placement using complete path delay," in *Proc. ACM/IEEE 27th Design Automat. Conf.*, Orlando, FL, June 1990, pp. 84–89.
- [9] E. Shragowitz, J.-Y. Lee, and E. Kang, "Application of fuzzy logic to computer-aided design of electronic systems," in *Proc. SPIE Int. Symp. Aerosp./Defense Sensing Contr.*, Orlando, FL, Apr. 1996, pp. 196–209
- [10] R. Felix and A. Hollmann, "A Knowledge-based system using reasoning with uncertainty in the VLSI chip synthesis domain," *Eng. Applicat. Artificial Intell.* vol. 2, pp. 144–161, June 1989.
- [11] M. Razaz and J. Gan, "A novel placement method for VLSI design," *IEEE Proc. VLSI & Computers*, 1987, pp. 642-645.
- [12] E. Kang, R.-B. Lin, and E. Shragowitz, "Fuzzy logic approach to VLSI placement," *IEEE Trans. Very Large Scale Integrat. Syst.*, vol. 2, pp. 489–501, Dec. 1994.
- [13] , "Technology mapping for FPGA's with complex block architectures by fuzzy logic technique," in *Proc. Asia South-Pacific Design Automat. Conf. (ASP-DAC)*, Makuhari, Japan, Aug. 1995, pp. 295–300.
- [14] , "Application of fuzzy logic for technology mapping of field programmable gate arrays," in *Proc. SPIE Int. Symp. Aerosp./Defense Sensing Contr.*, Orlando, FL, Apr. 1996, pp. 237–248.
- [15] H. AZIZI, K. Benmmahamed, L. Zouaoui, "Application of Fuzzy Logic to Layout of Integrated Circuit", *AJIT* 5(9), 1001-1004, 2006.

- [16]** H. AZIZI, K. Benmmahamed, L. Zouaoui, "Application of Fuzzy Logic to VLSI cells placement", CMNT06, Tizi-Ouzou Algeria, 19-23 Nov. 2006
- [17]** Sadiq M. Sait, Habib Youssef et al., "Iterative heuristics for multiobjective VLSI standard-cell placement", IEEE 2001
- [18]** Junaid A. Khan, Sadiq M. Sait, "Fuzzy biasless simulated evolution for multiobjective VLSI placement", IEEE 2002.

Le Parallélisme dans la Conception VLSI

V.1 INTRODUCTION

Avec l'évolution rapide des circuits intégrés VLSI, la conception automatique est devenue très grande et demande une énorme quantité de traitement. Il était donc indispensable de penser à augmenter cette capacité de calcul: le parallélisme offre cette opportunité [1] [2].

Bien souvent, il faut avoir recours à une conception et à un développement parallèle de l'application pour exploiter le parallélisme de façon intéressante. Les langages de programmation parallèles sont une composante essentielle dans la chaîne de conception d'une application. Nous présentons les principaux. Enfin, l'intérêt des architectures parallèles réside dans les performances qu'elles permettent d'atteindre.

L'objectif de ce chapitre est d'introduire la notion de parallélisme, de discuter la nécessité de l'exploiter pour atteindre de hautes performances et de présenter les différentes formes d'architectures d'ordinateurs parallèles. Ensuite donner une proposition de solutions d'algorithmes parallèles pour résoudre le problème de placement/routage. Le placement est un problème important et difficile dans la conception physique automatisée. Un placement de mauvaise qualité peut empêcher un circuit de fonctionner à une vitesse acceptable. En outre, si l'algorithme de placement a la complexité élevée, nous ne pourrions pas obtenir un placement dans une quantité de temps acceptable. Ceci, alternativement, peut nous forcer à sacrifier la qualité du placement pour réaliser un temps de placement plus rapide.

Le chapitre est divisé en trois parties, la première partie présente le concept général du parallélisme, la deuxième partie concerne la proposition de solution pour paralléliser la phase de placement de cellules. En effet, le parallélisme intrinsèque des réseaux de neurones a été exploité pour réaliser le placement. Enfin, dans la troisième partie une solution au problème de routage a été proposé en utilisant le principe de partition du circuit en plusieurs parties indépendantes.

V.2 CONCEPT GENERAL

V.2.1 PROGRAMMATION PARALLELE

Avant d'aborder les principales architectures parallèles, nous présentons les spécificités de la programmation parallèle. Nous introduisons les notions de granularité des applications et des machines parallèles, les modes de communication entre les différents processeurs de la machine, la synchronisation de ces processeurs...

Lors de l'exécution d'une application séquentielle, la totalité des calculs affectés à cette application sont effectués par un unique processeur. Charge est au programmeur d'adapter son implantation à cette caractéristique.

Lors d'une exécution sur machine parallèle, généralement effectués pour abaisser le temps d'exécution de l'application, plusieurs processeurs sont utilisés. Les calculs sont effectués par ces différents processeurs et le programmeur doit répartir les calculs sur les différents processeurs. Cet apport de puissance de calcul entraîne donc des modifications dans la conception de la programmation des applications.

V.2.2 PARALLELISME ET GRANULARITE

La granularité d'une application fait référence à la division de la tâche à accomplir en calculs parallèles autonomes. La granularité donne une information sur la quantité de calculs effectués par les processeurs entre deux opérations de communication ou de synchronisation.

Un parallélisme à grain fin désigne une application à bas niveau de parallélisme, constituée d'une somme de tâches parallèles relativement courtes. Les applications parallèles à grain fin désignent donc des applications effectuant fréquemment des opérations de synchronisation ou de communication entre les différents processeurs (Figure V.1 (a)).

Un parallélisme à gros grain désigne une application à haut niveau de parallélisme, avec de grandes périodes parallèles autonomes, donc avec une fréquence de communication et synchronisation assez faible (Figure V.1 (b)) comparée aux périodes de calcul des processeurs.

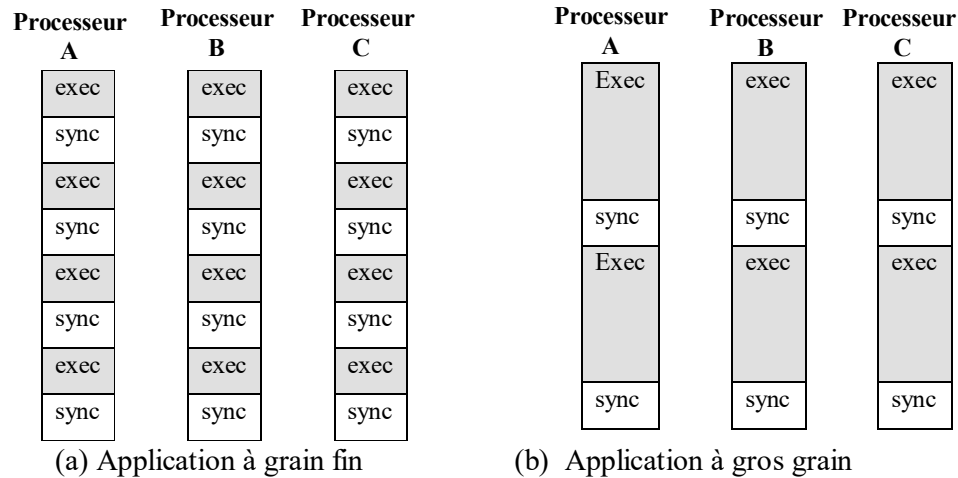


Figure V.1 Granularité

Une programmation à grain fin d'un réseau de neurones consisterait, par exemple, à placer un neurone par processeur, un parallélisme à gros grain consisterait à recopier le réseau sur chacun des processeurs disponibles et à distribuer les exemples sur ces processeurs. Dans le premier cas, chaque communication entre les neurones entraîne une opération de communication entre deux processeurs. Dans le second cas les communications entre les processeurs n'auront lieu que pour comparer/ajuster les résultats, donc beaucoup plus rarement.

V.2.3 LOI DE AMDHAL

Les raisons essentielles de l'utilisation d'une machine parallèle, donc de la parallélisation d'un programme, sont d'ordre temporel. Une implantation sur machine parallèle doit accélérer l'exécution d'un programme. Dans tous les cas, l'idéal recherché lors de la parallélisation d'une application est de diviser le temps de son application par le nombre de processeurs utilisés (Equation V.1).

$$\text{Temps d'Exécution Parallèle} = \frac{\text{Temps d'Exécution Séquentiel}}{\text{Nombre de processeurs Utilisés}} \quad (\text{V.1})$$

Cette équation sous-entend que la totalité des opérations effectuées par l'application s'exécutent en parallèle. Amdhal remarque qu'une application possède toujours une partie non parallélisable, sa fraction séquentielle, ce qui implique que l'accélération d'une application parallèle, exécutée sur P processeurs, est plafonnée à une accélération maximale, accélération dans l'équation V.2

$$\text{Accélération Maximale} = \frac{\text{Temps Séquentiel}}{\frac{\text{Part Parallèle}}{P} + \text{Part séquentielle}} \quad (\text{V.2})$$

La loi de Amdhal indique donc qu'il existe une limite aux performances de la parallélisation d'un programme. Soit f_s la proportion non parallélisable d'une application. Si $S(P)$ représente l'accélération de l'application sur P processeurs, les performances de la parallélisation de l'application seront limitées, quel que soit le nombre de processeurs utilisés, comme décrit dans l'équation V.3

$$S(P) = \frac{1}{f_s + \frac{(1-f_s)}{P}} \leq \frac{1}{f_s} \quad (\text{V.3})$$

V.2.4 LA COMMUNICATION ENTRE PROCESSEURS

Lors de l'exécution d'une application parallèle, quel que soit le type de machine parallèle utilisée, chaque processeur manipule des données. L'une des spécificités de la programmation parallèle est la transmission des données d'un processeur à un autre. Il s'agit de se demander comment une donnée utilisée par un certain processeur peut être manipulée (en lecture et/ou en écriture) par un ou plusieurs autres processeurs de la machine parallèle. Il existe en fait deux paradigmes principaux permettant de communiquer des données d'un processeur à un (aux) autre(s) sur les machines parallèles : l'envoi de messages ou le partage de mémoire.

La communication par envoi de message considère que chaque processeur est propriétaire des données qu'ils manipulent. Dans ce protocole les communications sont explicites. Si l'exécution d'un processeur nécessite l'utilisation d'une variable gérée par un processeur différent, la donnée doit être envoyée par le processeur hôte de la variable au processeur qui la demande, ce dernier devant, quant à lui, la recevoir explicitement.

Le second paradigme de communication entre processeurs d'une machine parallèle est la communication par partage de mémoire. Une application à communication par partage de mémoire considère la machine parallèle comme une somme de processeurs ayant accès à une mémoire principale unique. Avec ce paradigme de communication, comme pour la programmation séquentielle, chaque variable dispose d'un adressage unique et est accessible en lecture comme en écriture par tous les processeurs de la machine. Par cette caractéristique, et en comparaison de la communication par envoi de message, cette méthode de communication est souvent considérée

comme la plus simple à utiliser, car la plus proche de la programmation séquentielle. Néanmoins, la parallélisation à l'aide de ce type de communication entre les processeurs comporte des difficultés, toujours liées à la gestion des données.

V.2.5 LA SYNCHRONISATION ENTRE LES PROCESSUS

La programmation parallèle entraîne le besoin de synchronisation entre les processeurs. Il est en effet utile pour un processeur de s'assurer de la disponibilité de certains résultats, calculés par d'autres processeurs, pour finir son exécution. Pour ce faire, il est nécessaire de synchroniser les processeurs, c'est-à-dire d'attendre que certains processeurs, explicitement désignés par le programmeur, soient tous arrivés à un point de rendez-vous pour reprendre le cours de l'exécution.

Il existe diverses formes de synchronisation dans un programme. Un processeur peut juste signaler qu'il est passé par un point défini de son algorithme, et continuer son exécution. Un processeur peut être obligé d'attendre l'arrivée d'autres processeurs à un certain point pour continuer son exécution, on parle dans ce cas de barrière de synchronisation.

V.2.6 MOTIVATION POUR LE PARALLELISME

L'exploitation du parallélisme dans l'architecture des ordinateurs est née de la conjonction de trois éléments : les besoins des applications, les limites des architectures séquentielles et l'existence dans les applications de la propriété de parallélisme. C'est le parallélisme présent dans une application qui permet d'exécuter simultanément, par des ressources matérielles différentes, plusieurs parties de cette application.

L'approche intuitive introduit la notion de dépendance entre deux ou plusieurs programmes (ou opérations). Pour que deux programmes ou deux opérations puissent être exécutés en parallèle, il faut :

- 1) qu'ils (elles) soient indépendants (tes) ;
- 2) que l'on puisse détecter cette indépendance ;
- 3) qu'il existe suffisamment de ressources pour les exécuter simultanément. En pratique, il existe trois limites au parallélisme : les dépendances de données, les dépendances de contrôle et les dépendances de ressources [4] [5] [6].

V.3 LES TROIS PHASES DE L'EXPLOITATION DU PARALLELISME

V.3.1 EXPLOITATION DU PARALLELISME DE CONTROLE

Voici une définition du parallélisme de contrôle [3] : c'est un type de parallélisme pour lequel des opérations différentes sont réalisées simultanément. Ce parallélisme peut provenir de l'existence dans le programme de fonctions indépendantes. Il peut aussi provenir d'opérations indépendantes dans une suite d'opérations. Ce parallélisme ne dépend donc pas des données mais de la structure du programme à exécuter. C'est l'absence de dépendances entre différentes parties du programme (quelle que soit leur taille : fonctions, boucles, opérations) qui est la source du parallélisme de contrôle.

L'exploitation de ce parallélisme suppose que les processeurs fonctionnant simultanément soient capables de dérouler leur propre programme (puisque les traitements peuvent être tous différents). Il existe une autre forme d'exploitation du parallélisme de contrôle. Elle est appropriée à un cas particulier de parallélisme de contrôle pour lequel il existe une dépendance entre les parties (fonctions, boucles, opérations) du programme qui peuvent être exécutées en parallèle.

Cette forme d'exploitation du parallélisme de contrôle s'appelle le **pipeline**. Elle repose, comme nous venons de le voir, sur l'organisation des données à traiter sous la forme d'un flux d'informations et sur l'application de plusieurs traitements consécutifs sur chaque élément de ce flux.

On distingue deux cas de dépendances:

- Dépendance de contrôle de séquence : correspond au séquençage dans un algorithme classique,
- Dépendance de contrôle de communication : lorsqu'une action envoie des informations à une autre action.

V.3.2 EXPLOITATION DU PARALLELISME DE DONNEES

Voici une définition du parallélisme de données [3] : c'est un type de parallélisme pour lequel la même opération est réalisée simultanément par de nombreux processeurs sur des données différentes. Cette définition recouvre deux notions : la présence d'un parallélisme issu des données et la manière d'exploiter ce parallélisme. La définition indique aussi une manière d'exploiter ce parallélisme.

Il s'agit d'utiliser de nombreux processeurs simultanément et de leur faire exécuter la même opération. Généralement, le nombre de processeurs est beaucoup plus petit que le parallélisme potentiel et chaque processeur devra donc traiter plusieurs données.

V.3.3 EXPLOITATION DU PARALLELISME DE FLUX

L'exploitation du parallélisme de Flux vient de la constatation naturelle que certaines applications fonctionnent selon le mode du travail à la chaîne : on dispose d'un Flux de données, généralement similaires, sur lesquelles on doit effectuer une suite d'opérations en cascade. Les ressources de calcul sont associées aux actions et chaînées de manière à ce que les résultats des actions effectuées au temps T soient passés au temps $T+1$. Ce mode de fonctionnement est appelé Pipe-Line.

V.4 CLASSIFICATIONS DES ARCHITECTURES PARALLELES

La classification la plus utilisée est connue historiquement sous le nom de classification de Flynn [3]. Elle classe les architectures suivant les relations entre les unités de traitement et les unités de contrôle. Les unités de traitement calculent chacune un flux de données ou « Data stream ». Les unités de contrôle déroulent chacune un programme (un flux d'instructions ou « Instruction stream »). La figure V.2 présente cette classification.

Classification des architectures parallèles selon Flynn		
Flux	Un flux d'instructions (un programme)	Plusieurs flux d'instructions
Un flux de données	SISD	MISD (pipe line)
Plusieurs flux de données	SIMD	MIMD

Figure V.2 Classification des architectures parallèles

Le modèle SISD (Single Instruction stream Single Data stream) correspond au modèle classique de Von Neuman (architecture séquentielle) pour lequel une seule unité de traitement reçoit un seul flux d'instructions pour traiter un seul flux de données.

Dans une architecture SIMD (Single Instruction stream Multiple Data stream), une seule unité de contrôle gère le séquençement du programme pour plusieurs unités de traitement. En pratique, les unités de traitement fonctionnent en synchronisme et reçoivent les mêmes instructions en même temps. Comme chaque unité de traitement calcule sur un flux de données différent (des données différentes), la même opération est appliquée à plusieurs données simultanément.

Dans les architectures MISD (Multiple Instruction stream Single Data stream), un seul flux de données reçoit plusieurs traitements simultanément. Le mode pipeline d'exploitation du parallélisme de contrôle correspond assez bien à cette classe.

L'organisation MIMD (Multiple Instruction stream Multiple Data stream) est la plus utilisée. Plusieurs unités de contrôle gèrent chacune une ou plusieurs unités de traitement. Ce modèle permet donc d'exécuter un programme différent sur tous les processeurs.

Les applications peuvent donc être conçues comme un ensemble de tâches différentes coopérantes (applications parallèles et/ou distribuées).

Le modèle MIMD correspond aussi à l'organisation de la grande majorité des architectures parallèles. La tendance à l'utilisation massive de composants standards pour la réalisation des machines parallèles se concrétise par l'adoption des microprocesseurs pour réaliser les parties calcul et contrôle de ces machines.

Dans les microprocesseurs, on trouve une unité de contrôle et une ou plusieurs unités de traitement. En répliquant les microprocesseurs, on réplique donc, dans la même proportion, les unités de contrôle et les unités de traitement. Lors de l'exécution d'un programme parallèle, plusieurs flux d'instructions différents peuvent donc être déroulés pour traiter plusieurs flux de données.

Pour le traitement numérique, écrire des programmes différents est très difficile et souvent inutile. Aussi, on préfère exécuter le même programme sur tous les processeurs. Bien évidemment, le programme traite des données différentes sur chaque processeur. Ce mode d'exécution est nommé SPMD (Single Program over Multiple Data stream).

V.5 EXPLOITATION DU PARALLELISME DANS LA CONCEPTION VLSI

Avec l'accroissement de la complexité des circuits VLSI, il y a une grande nécessité d'avoir des outils plus développés dans les différentes phases de conception des circuits intégrés.

Malgré, l'amélioration des performances des processeurs (vitesse d'horloge, mémoire cache..), il existe encore des tâches, dans le processus de CAO, qui continuent à prendre beaucoup de temps d'exécution. L'approche qui permet d'améliorer le temps d'exécution reste exclusivement le parallélisme.

Quelques tâches dans le processus de conception des circuits intégrés ont utilisé les algorithmes parallèles : plan de masse [7], extraction [8,9], simulation [10,11], simulation logique [12], et le test [13].

Pour le problème de placement, comme le temps d'exécution a considérablement augmenté (placement d'un très grand nombre de cellules) alors une variété d'algorithmes parallèles a été développé. Ces algorithmes sont classés en fonction de la machine cible. En utilisant, l'envoi de message, Banerje et al [14,15] développe un algorithme parallèle de recuit simulé pour le placement cellules. Ravikumar et Sastry [16] utilise la technique diviser pour conquérir, ainsi le circuit n'est pas traité en globalité mais divisé en partitions traitées séparément. Le partage de mémoire a été aussi utilisé pour réaliser des algorithmes parallèles. Casotto et al [17] utilise cette approche dans son algorithme.

Pour le problème de routage, la majorité des algorithmes développés sont séquentiels. Cependant, il existe une variété d'algorithmes utilisant l'approche parallèle. Ainsi, dans [18,19] l'algorithme de routage de labyrinthe a été implémenté sur une machine à multiprocesseurs hypercube. Une implantation sur machine parallèle utilisant le partage de mémoire avec un autre algorithme proposé dans [20-22]. Par contre, d'autres algorithmes proposent un algorithme parallèle combiné pour le placement et routage [23-27].

Dans cette partie, nous allons proposer deux solutions pour paralléliser la phase placement/routage. La première utilise le parallélisme intrinsèque des réseaux de neurones pour la phase de placement et la deuxième méthode utilise la décomposition hiérarchique pour la phase de routage.

V.6 APPLICATION

La recherche de puissance de calcul, inhérente à la recherche en réseau de neurone artificiel, entraîne naturellement à s'intéresser aux outils informatiques proposant les meilleures performances

dans ce domaine: les calculateurs parallèles. En plus des capacités proposées par ces machines, le passage au parallélisme, ardu pour la plupart des problèmes, pourrait sembler aisés au niveau connexionniste. Les réseaux de neurones sont en effet connus pour leur parallélisme naturel, dit intrinsèque. Exécuter des réseaux de neurones d'unités de calcul neuronales sur un réseau de processeurs semble être une solution efficace et simple.

Nous allons, dans cette partie, présenter un survol des parallélisation des réseaux connexionnistes. Nous allons présenter le(s) parallélisme(s) présent(s) dans les architectures de réseaux de neurones artificiels. Nous verrons ensuite la méthode utilisée pour implanter notre programme "parallèle" en utilisant le parallélisme naturel des réseaux de neurones qui est à base de notre algorithme de placement de cellules VLSI (Chapitre 2).

V.6.1 LE PARALLELISME INTRINSEQUE DES RESEAUX DE NEURONES

Les réseaux de neurones sont souvent décrit comme des modèles de calcul et de mémoire distribués. Thomas Nordström [28][29] propose une étude plus détaillée du parallélisme disponible au sein des modèles connexionnistes.

Algorithme V.1 décomposition d'un algorithme classique de réseaux de neurones

```

Pour tout Session d'apprentissage faire
  Pour tout Ensemble de la session faire
    Pour tout Ensemble de couche faire
      Pour tout Neurone de la couche faire
        Pour tout Poids du neurone faire
          Pour tout Octet de la valeur du poids faire

```

Partant d'une décomposition d'une généralisation de l'exécution d'un réseau de neurone connexionniste, telle que présentée par l'algorithme, il extrait six principaux degrés de parallélisme dans l'algorithmique connexionniste, chaque degré affinant le parallélisme :

1. le parallélisme de session d'apprentissage : Il est possible d'exécuter et de faire apprendre, simultanément, différents réseaux. Ce degré de parallélisme signifie, concrètement, qu'il est possible de faire apprendre une même tâche à deux architectures neuronales différentes.

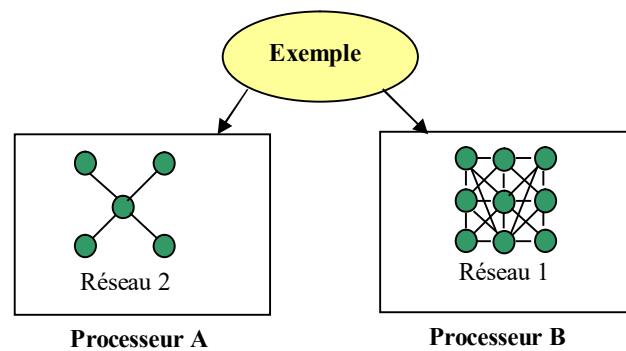


Figure V.3 Utilisation du parallélisme de session d'apprentissage.

(Des réseaux différents peuvent être exécutés simultanément, sur des espaces d'exemples différents, en étant placés sur des processeurs différents)

- le parallélisme des exemples d'apprentissage : Ce degré de parallélisme signifie qu'il est possible de faire traiter plusieurs exemples différents simultanément à un même réseau. En terme d'implantation, il s'agit de disperser les éléments du corpus d'exemples et de faire s'exécuter une copie identique d'un même réseau sur chacun des processeurs. Chaque copie du réseau calcule ses sorties et met à jour, différemment, ses poids. Chaque processeur travaille sur un unique espace de données.

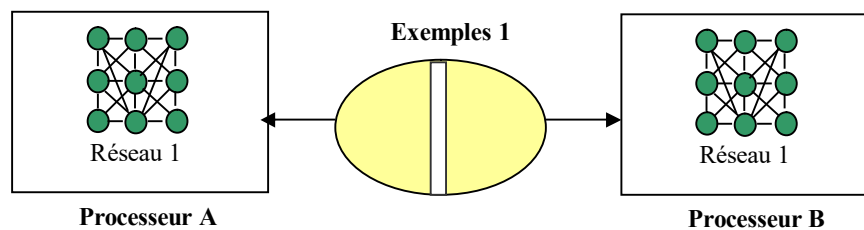


Figure V.4 Le parallélisme des exemples d'apprentissage

(Le réseau est recopié intégralement sur chaque processeur et l'espace des exemples est réparti sur ces processeurs)

- le parallélisme des couches : Ce parallélisme, applicable aux réseaux à couches, indique que les différentes couches d'un réseau peuvent s'exécuter simultanément. Le traitement d'un réseau à couche peut être vu comme un flux d'information transmis de couche en couche, tous les neurones d'une même couche recevant leur information simultanément. En terme d'implantation, les différentes couches sont réparties sur différents processeurs et les résultats des couches transmis de processeurs en processeur.

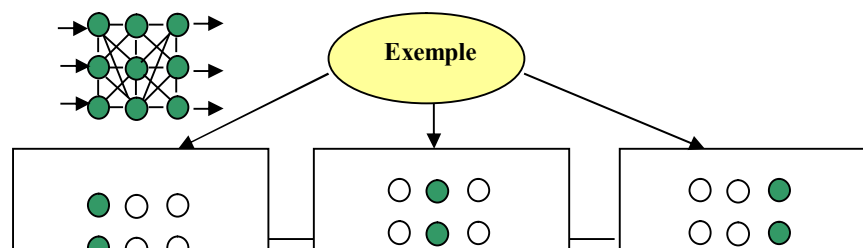


Figure V.5 Le parallélisme des couches du réseau.

(Chaque couche est placée sur un processeur distinct. Les différentes couches du réseau travaillent donc simultanément)

4. le parallélisme des neurones : Il s'agit de la possibilité d'exécuter simultanément les tâches imparties aux différents neurones d'un même réseau. En terme d'implantation, les neurones du réseau sont répartis sur les différents processeurs et travaillent de façon concurrente. L'espace des entrées est partagé par différents processeurs.
5. le parallélisme des synapses : Ce parallélisme postule la possibilité d'exécution concurrente des différentes synapses d'un même neurone, et par la même, des différentes synapses d'un même réseau. En terme d'implantation, les différentes synapses d'un même neurone peuvent être réparties, sur différents processeurs lorsque le neurone doit effectuer la somme pondérée de ses entrées.
6. le parallélisme des octets

Avec le deuxième type de parallélisme, nous avons procédé à un travail de simulation où les calculs sont exécutés par un seul processeur.

L'idée d'apporter la solution du parallélisme matériel aux difficultés de performances des réseaux de neurones vient essentiellement du parallélisme intrinsèque de ces modèles. Les réseaux connexionnistes sont en effet considérés comme des modèles parallèles et distribués.

Comme mentionné dans le paragraphe ci-dessus, il existe six degrés de parallélisme dans les modèles des réseaux de neurones. Pour paralléliser une application il est nécessaire d'en extraire les tâches pouvant être exécutées de manières concurrentes. Utiliser le parallélisme inhérent aux modèles connexionnistes implique donc d'utiliser au moins l'un de ces six degrés de Nordström.

De ces six parallélismes, l'exécution des applications sensiblement différentes sur des processeurs, ou même des ordinateurs, différents (parallélisme de session d'apprentissage) semble le mieux adapté pour notre application puisqu'il nous permet d'exécuter sur une machine séquentielle plusieurs variantes d'un même réseau. Nous avons effectué des tests sur une machine séquentielle en changeant le réseau pour le même exemple (parallélisme simulé).

L'implantation de ce programme est possible sur une machine parallèle SPMD (Single Program over Multiple Data stream). Aussi, on préfère exécuter le même programme sur tous les processeurs. Bien évidemment, le programme traite des données différentes sur chaque processeur.

Exemple :

Après avoir présenté les domaines des réseaux de neurones artificiels et du parallélisme informatiques, nous présentons dans cette partie le choix qui nous a guidé dans l'implantation du programme de placement de cellules. Nous pouvons ensuite, justifier notre choix en fonction des conclusions obtenues sur ces différents aspects.

Nous prenons un exemple de 13 cellules à 58 bornes. Nous exécutons notre programme avec le même exemple sur trois réseaux différents. L'application peut être définie comme suit :

Exemple : c'est un circuit à 13 cellules et 58 bornes.

Réseau 1: réseau connexionniste de dimension 8 x 8

Réseau 2: réseau connexionniste de dimension 12 x 12

Réseau 3: réseau de dimension 16 x 16

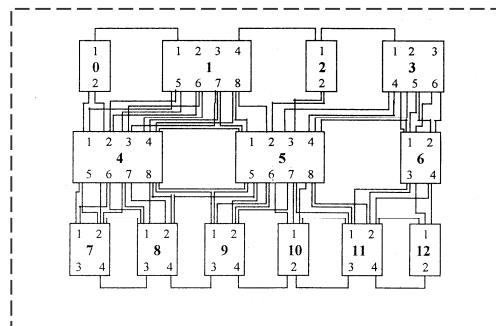


Figure V.6 Exemple de parallélisation

	Processeur A	Processeur B	Processeur C
	Réseau 1 (8 x 8)	Réseau 2 (12 x 12)	Réseau 3 (16 x 16)
Exemple	63 Itérations	58 Itérations	58 Itérations
13 Cellules 58 bornes	- 0 1 1 1 1 - 2 - 0 1 1 1 1 - 2 - 4 4 4 4 3 3 3 - 4 4 4 4 3 3 3 12 7 7 8 8 9 9 10 12 7 7 8 8 9 9 10 6 6 11 11 5 5 5 5 6 6 11 11 5 5 5 5	- - - - - 6 6 - - - - - - - 6 6 - - - 0 1 1 1 1 2 3 3 3 - 0 1 1 1 1 2 3 3 3 - 4 4 4 4 5 5 5 5 12 - 4 4 4 4 5 5 5 5 12 - 7 7 8 8 9 9 10 11 11	- 0 1 1 1 1 2 3 3 3 - - - 0 1 1 1 1 2 3 3 3 - - - 4 4 4 4 5 5 5 5 6 6 - - 4 4 4 4 5 5 5 5 6 6 - - 7 7 8 8 9 9 10 11 11 12 - - 7 7 8 8 9 9 10 11 11 12 - - - - - - - - - - - -

Figure V. 7 Résultats de l'exemple de la figure 6

Il est clair que le placement de qualité optimale est obtenu avec le réseau 16 x 16.

	Processeur A		Processeur B		Processeur C	
	Réseau 1		Réseau 2		Réseau 3	
Exemples (cellules)	Itérations	Temps(s)	Itérations	Temps(s)	Itérations	Temps(s)

25	1602 (5 x 5)	5	54 (7 x 7)	1	54 (10 x 10)	1
36	188 (6 x 6)	1	188 (10 x 10)	2	188 (12 x 12)	2
Hyper cube (8)	129 (3 x 3)	1	121 (4 x 4)	2	121 (6 x 6)	2
Asymétrique (15)	486 (4 x 4)	1	140 (6 x 6)	2	140 (8 x 8)	2
100	211 (10 x 10)	13	211 (15 x 15)	18	211 (20 x 20)	21
144	302 (12 x 12)	59	302 (18 x 18)	69	302 (24 x 24)	82
225	Diverge (15x 15)	Infini	469 (20 x 20)	467	469 (30 x 30)	518
256	632 (16 x 16)	771	532 (25 x 25)	819	532 (32 x 32)	1411
289	25000 (17 x 17)		600 (27 x 27)	1412	600 (27 x 27)	1495

Tableau V.8 parallélisme simulé

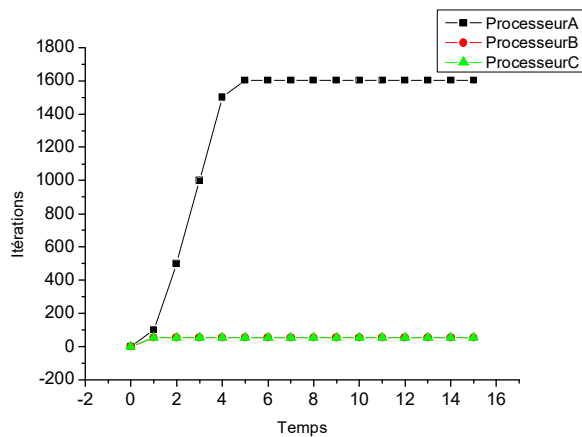


Figure V.9 Evolution des itérations avec un exemple de 25 cellules

V.6.2 PARALLELISME POUR LE ROUTAGE

Une manière de paralléliser le programme de routage est de considérer tous les nœuds simultanément et peut être formulée par la programmation entière. Dans notre cas, nous avons utilisé la méthode de décomposition de l'espace de routage en n zones, chaque zone sera connectée indépendamment de l'autre et par conséquent chaque zone sera affectée à un processeur.

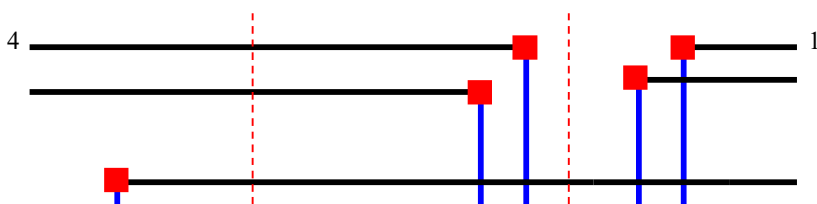




Figure V.10 Décomposition verticale de l'espace de routage

Un exemple de décomposition (verticale): un circuit avec 10 nœuds à connecter (Figure V.10). Il s'agit d'affecter au processeur A la zone 1 (borne gauche à la limite L1), au processeur B la zone 2 (entre les limites L1 et L2) et au processeur C la zone 3 (la limite L2 et la borne droite).

Cette méthode ne va pas améliorer la qualité du placement, par contre le temps d'exécution sera nettement diminué. D'après la loi de AMDHAL (Equation V.1) le temps d'exécution parallèle est égal au temps d'exécution séquentiel, divisé par le nombre de processeurs utilisés, pour une application complètement parallélisable, ce qui est le cas de cette application.

Nombre de nœuds	Nombre de VIA			Total des VIA
	Processeur A	Processeur B	Processeur C	
10	4	10	11	25
20	8	13	15	36
45	15	25	30	70
68	15	30	40	85
100	25	45	55	125

V.11 Tableau Résultats des tests avec décomposition verticale de l'espace de routage

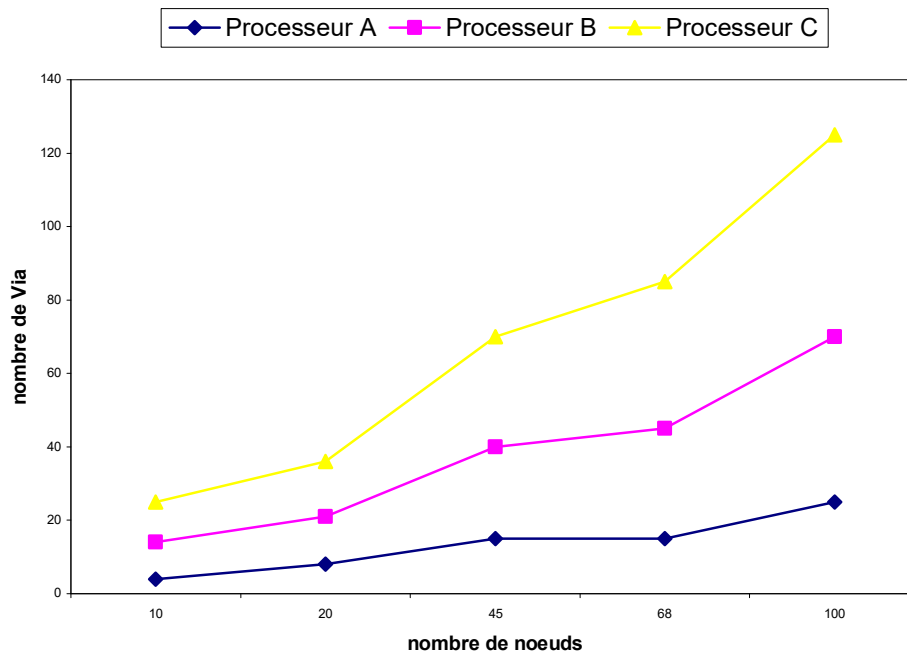


Figure V.12 Evolution des contacts par rapport aux nombres de nœuds

V.7 CONCLUSION

Le parallélisme est appliqué aux problèmes d'optimisation dans un cadre purement temporel. En effet, toute application parallélisable peut être décomposée en un certain nombre de sous-problèmes et exécutés sur des processeurs différents. Dans notre cas, nous avons exploité le parallélisme intrinsèque pour réaliser le placement de cellules. Les résultats étaient satisfaisants. Néanmoins, en perspective nous pouvons aussi utiliser le recuit simulé pour paralléliser notre tâche. En effet, le placement de cellules standard basé sur le recuit simulé a été longtemps reconnu comme un processus de calcul intensif, et par conséquent plusieurs efforts de recherches ont été entrepris pour paralléliser cet algorithme.

L'approche utilisée pour paralléliser le routage est une technique aussi utilisée pour la phase de placement (bisection, quadrisection, ...). La remarque qui se dégage de cette approche est que l'amélioration qui touche exclusivement l'aspect temporel et n'améliore en aucun cas la qualité du routage. D'ailleurs c'est ce qu'on reproche aux solutions parallèles.

RÉFÉRENCES DU CHAPITRE 5

- [1] D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawak, and C. Packer. "BEOWULF: A parallel workstation for scientific computation". In International Conference on Parallel Processing, 1995.
- [2] P. Banerjee. "Parallel Algorithms for VLSI Computer-Aided Design". Prentice Hall, Englewood Cliffs, New Jersey, 1994.
- [3] B. Perhami, "Introduction to Parallel Processing, Algorithm and Architect", University of California, Kluwer Academic Publishers, N.Y 2002.
- [4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. "Optimal partitioners and end-case placers for standard cell layout". In Proc. Int. Symp. on Physical Design, pages 90–96, 1999.
- [5] A. Casotto and A. Sangiovanni-Vincentelli. "Placement of standard cells using simulated annealing on the connection machine". In Proceedings of the International Conference on Computer-Aided Design, pages 350–353, November 1987.
- [6] R. J. Brower "Parallel Algorithms for Placement and Routing in VLSI design", PHD Thesis, Urbana Illinois, 1991.
- [7] R. Jayaraman and R. A. Rutenbar, "Floorplanning by annealing on a hypercube multiprocessor," Proc. Int. Conf. Computer-Aided Design, pp. 346-349, Nov. 1987.
- [8] K. P. Belkhale and P. Banerjee, "PACE2: An improved parallel VLSI extractor with parametric extraction," Proc. Int. Conf. Computer-Aided Design, pp. 526-530, Nov. 1989.
- [9] B. Tonkin, "Circuit extraction on a message-based multiprocessor," Proc. 27th Design Automat. Conf., pp. 260-265, June 1990.
- [10] G. G. Hung, Y. C. Wen, K. Gallivan, and R. Saleh, "Parallel circuit simulation using hierarchical relaxation," Proc. 27th Design Automat. Conf., pp. 394-399, June 1990.
- [11] G. C. Yang, "PARASPICE: A parallel circuit simulator for shared-memory multiprocessors," Proc. 27th Design Automat. Conf., pp. 400-405, June 1990.
- [12] K. Subramanian and M. R. Zargham, "Distributed and parallel demand driven logic simulation," Proc. 27th Design Automat. Conf., pp. 485-490, June 1990.
- [13] S. Patil and P. Banerjee, "A parallel branch and bound approach to test generation," IEEE Trans. Computer-Aided Design, vol. 9, no. 3, pp. 313-322, Mar. 1990.
- [14] P. Banerjee, M. H. Jones, and J. S. Sargent, "Parallel simulated annealing algorithms for standard cell placement on hypercube multiprocessors," IEEE Trans. Parallel and Dist. Syst., vol. 1, no. 1, pp. 91-106, Jan. 1990.
- [15] J. Sargent and P. Banerjee, "A parallel row-based algorithm for standard cell placement with integrated error control," Proc. 26th Design Automat. Conf., pp. 590-593, June 1989.
- [16] C. P. Ravikumar and S. Sastry, "Parallel placement on hypercube architecture," Proc. Int. Conf. Parallel Process., Vol. III, pp. 97-101, 1989.

- [17] A. Casotto, F. Romeo, and A. Sangiovanni-Vincentelli, "A parallel simulated annealing algorithm for the placement of macro-cells," Proc. Int. Conf. Computer-Aided Design, pp. 30-33, Nov. 1986.
- [18] S. A. Kravitz and R. A. Rutenbar, "Placement by simulated annealing on a multiprocessor," IEEE Trans. Computer-Aided Design, vol. CAD-6, no 4, pp. 534-549, June 1987.
- [19] O. A. Olukotun and T. N. Mudge, "A preliminary investigation into parallel routing on a hypercube computer," Proc. 24th Design Automat. Conf., pp. 814-820, June 1987.
- [20] Y. Won and S. Sahni, "Maze routing on a hypercube multiprocessor computer," Proc. Int. Conf. Parallel Process., pp. 630-637, Aug. 1987.
- [21] Jonathan Rose, "LocusRoute: A parallel global router for standard cells," Proc. 25th Design Automat. Conf., pp. 189-195, June 1988.
- [22] R. J. Brouwer and P. Banerjee, "PHIGURE: A Parallel Hierarchical Global Router," Proc. 27th Design Automat. Conf., pp. 650-653, June 1990.
- [23] A. A. Szepieniec, "Integrated placement/routing in sliced layouts," Proc. Design Automat. Conf., pp. 300-307, 1986.
- [24] E. Shragowitz, J. Lee, and S. Sahni, "Placer-router for sea-of-gates design style," Proc. Int. Conf. Computer Design, pp. 330-335, 1987.
- [25] W. M. Dai and E. S. Kuh, "Simultaneous floor planning and global routing for hierarchical Building-Block Layout," IEEE Trans. Computer-Aided Design, vol. CAD-6, no. 5, pp. 828-837, Sept. 1987.
- [26] M. Igusa, M. Beardslee, and A. Sangiovanni-Vincentelli, "ORCA: A sea-of-gates place and route system," Proc. 26th Design Automat. Conf., pp. 122-127, June 1989.
- [27] P. R. Suaris and G. Kedem, "A quadrisection-based combined place and route scheme for standard cells," IEEE Trans. Computer-Aided Design, vol. 8, no. 3, pp. 234-244, Mar. 1989.
- [28] T. Nordsström et B. Svenson, "Using and designing massively parallel computer for artificial neural networks, Journal of Parallel and Computing , 14(3), 260-285, 1992.
- [29] T. Nordsström, "Highly Parallel computer for artificial neural networks", PHD thesis, Sweden 1995.

Résumé :

Depuis l'arrivée des circuits VLSI, la complexité des circuits implantés n'est plus limitée par la technologie, mais par le processus de conception. Un concepteur ne peut plus réaliser certains circuits sans l'aide d'outils de CAO tant la complexité des circuits est grande, et ceci pour plusieurs raisons : le temps de conception, l'optimisation et la stabilité des circuits. Notre travail s'inscrit dans le cadre de l'automatisation de la conception des circuits intégrés, à partir de la phase physique qui inclut le placement des cellules et le routage, et application de la logique floue et le parallélisme à ces outils. En effet, nous avons pris en considération, explicitement, deux aspects importants: présence de multiples objectifs contradictoires et utilisation de la connaissance experte dans la prise de décision. Ces aspects peuvent être traités par la technique basée sur la théorie des ensembles flous. Ensuite, nous avons introduit le concept du parallélisme et son application à la conception des circuits intégrés pour la phase de placement et routage.

Mots clés: microélectronique, CAO, placement, routage, Logique floue, parallélisme

المخلص

منذ مجيء تركيبات من نوع **VLSI**، فدرجة العقيد ليست مرتبطة بالتكنولوجيا وإنما بسياق التصور المبتكر لا يستطيع إنجاز بعض التركيبات بدون إيعار من **CAO** وهذا نظرا للتعقيدات الكبيرة للتركيبات، وهذا لعدة أسباب: وقت الإنجاز، التوسع واستقرار التركيبات، العمل المنجز يندرج في إطار أتمتة مرحلة الابتكار للتركيبات منذ الطور الفيزيائي الذي يحتوي على توظيف وإنجاز الاتصال بين الخلايا وتطبيق مبدأ المنطق الضبابي والتوازي لهاتين المرحلتين. بالفعل، لقد أخذنا بعين الاعتبار، بوضوح جانبيين مهمين: وجود مساعي متباينة واستعمال المعرفة المختصة لإنجاز القرار.

هذه المبادئ يمكن تطبيقها بالتقنية المتمثلة في المجموعات الضبابية ثم أدخلنا التصور الموازي وتطبيقه لابتكار التركيبات بالنسبة لتوظيف وإنجاز الاتصال بين الخلايا.

الكلمات المفتاحية: ميكروإلكترونيك، CAO، تركيب، إنجاز الاتصال، المنطق الضبابي، التوازي.

Abstract

Since the arrival of the VLSI circuits, the complexity of the established circuits is not limited any more by technology, but by the process of design. An originator cannot carry out any more certain circuits without the assistance of tools of CAD so much the complexity of the circuits is tall, and this for several reasons: the time of design, the optimization and the stability of the circuits. Our work lies within the scope of the automation of the design of the integrated circuits, starting from the physical phase which includes the placement of the cells and the routing, and application of fuzzy logic and parallelism to these tools. Indeed, we took in consideration, explicitly, two significant aspects: presence of multiple contradictory objectives and use of expert knowledge in the decision-making. These aspects can be treated by the technique based on the Fuzzy set theory. Then, we introduced the concept of parallelism and its application to the design of the integrated circuits for the phase of placement and routing.

Key words: microelectronic, CAD, placement, routing, Fuzzy Logic, parallel
