الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Éducation Nationale وزارة التربية الوطنية

جامعة فرحات عباس -سطيف-
UNIVERSITÉ FARHAT ABBAS - SÉTIF -

معهد الإلكترونيك
Institut d'Élèctronique

Thèse de MAGISTER

Option: **Microélèctronique**

Thème

PLACEMENT DES CELLULES DANS UN CIRCUIT INTEGRE PAR RECUIT SIMULE

Présenté par:
Melle: Zohra ZERROUGUI

Soutenue le 02/05/1994 devant le jury:

MM	**A. KHELLAF**	**Président**
	L. SELMANI	**Rapporteur**
	Y. BOUTERFA	**Rapporteur**
	A. MERZOUKI	**Examinateur**
	F. DJAHLI	**Examinateur**
	D. CHIKOUCHE	**Examinateur**

REMERCIEMENTS

*Je remercie M^r **ABD EL HAFID KHELLAF**, maître de conférence à l'institut d'électronique de SETIF, pour l'honneur qu'il me fait en acceptant de présider le jury de cette thèse.*

*Monsieur **A.MERZOUKI**, maître de conférence à l'institut d'électronique de SETIF, messieurs **F.DHAHLI**, **D.CHIKOUCHE**, chargés de cours à l'institut d'électronique de SETIF, ont accepté d'examiner mon travail , qu'ils reçoivent ici l'expression de ma profonde gratitude.*

*Je tiens à remercier, tout particulièrement, M^r **YOUCEF BOUTERFA**, Professeur associé à l'Université de Louvain, qui m'a guidé dans ce travail par ses conseils, ses critiques et son suivi. Qu'il trouve ici l'expression de ma profonde gratitude.*

*Mes plus vifs remerciements vont à M^r **LARBI SELMANI**, chargé de cours à l'institut d'électronique de SETIF et rapporteur de ce travail. Je le remercie profondément pour la confiance qu'il a mis à mon égard, son soutien continu, ses remarques, ses conseils et ses critiques pertinentes et constructives ainsi que pour tous les moyens matériels qu'il a mis à ma disposition. Qu'il trouve ici ma profonde reconnaissance et mon profond respect.*

*Je tiens particulièrement présenter mes vifs remerciements à celui qui m'a initié à la programmation **M^r ABD EL HAK FERHAT HAMIDA** qui, par sa disponibilité, ses conseils, sa compétence, et son aide continue, a apporté une indispensable contribution à la réalisation de ce travail.*

*Mes remerciements vont également à tous les membres du laboratoire de micro-électronique: **F.RADJAH**, **R.REMMOUCHE**, **A.ROUABHI** et **S.BENSALEM**, pour leur aide, leur esprit de bien faire et leur contribution à une ambiance de travail amicale.*

*Il m'est agréable d'exprimer mes remerciements les plus chaleureux à tous mes collègues en particulier **M^elle N.FRAHTA**, **D.MABROUK**, **M^r N.BENOUDJIT** et **A.BOUCHELAREM** pour leur aide et leur amitié qui m'étaient très précieuses.*

*Que messieurs **L.ZIET**, **A AZIZI**, **A.MOUSSAOUI** acceptent toute ma reconnaissance pour leur aide.*

Je souhaite enfin exprimer avec plaisir toute ma reconnaissance à tous ceux qui ont contribué à l'élaboration de ce travail soit par leur aide morale ou matérielle et dont je n'ai pas cité le nom.

SOMMAIRE

INTRODUCTION 1

CHAPITRE I: ENVIRONNEMENT C.A.O

INTRODUCTION 3
I ETAPES DE CONCEPTION D'UN CIRCUIT INTEGRE 3
II UTILISATION DE CALCULATEURS 5

CHAPITRE II: TECHNIQUES DE PLACEMENT DES CELLULES DANS UN CI

INTRODUCTION 7
I.ETAPES DE PLACEMENT 7
I.1. Placement initial 7
I.2. Amelioration du placement 8
II.TECHNIQUES DE PLACEMENT 9
II.1. Placement barycentrique ou par agrégat 9
II.2 Placement par la méthode Min-cut 11.
II.3 Placement relatif par la théorie des graphes 13
II.4 Placement par la méthode G.F.D.R 19
CONCLUSION

CHAPITRE III: TECHNIQUE DE PLACEMENT PAR RECUIT SIMULE

INTRODUCTION 22
I PRESENTATION DE LA METHODE DE RECUIT SIMULE 22
I.1 Principe général de la méthode 22
I.2 Analogie entre recuit réel et recuit simulé 23
I.3 Description de l'algorthme de Metropolis 25
I.4 Principales approches théoriques 26
II.PRINCIPALES APPLICATIONS 30
II.1 Imagerie 30
II.2 Télécommunication 30
II.3 Architécture 30
CONCLUSION

CHAPITRE IV: RECUIT SIMULE APPLIQUE AUX DIFFERENTS TYPES DE PLACEMENT

INTRODUCTION 32
I DIFFERENTS TYPES DE PLACEMENT 34
I.1 Placement pour Gate array 34
I.2 Placement pour standard cell 37
I.3 Placement pour macro cell 40

CHAPITRE V: PROGRAMME DE RECUIT

INTRODUCTION 43
I.PARAMETRES DE CONVERGENCE DE L'ALGORITHME 43
I.1 Température initiale 43
I.2 Loi de décroissance de la température 48
I.3 Durée des paliers 49
I.4 Critère d'arrêt du programme 50
II. DEROULEMENT DU PROGRAMME 50
II.1 Partitionnement 51
II.2 Mouvement 53
III.DEVELOPPEMENT D'UN SECOND LOGICIEL DE PLACEMENT 67
COMPARAISON DES DEUX LOGICIELS 68

CONCLUSION 70
BIBLIOGRAPHIE.

INTRODUCTION

INTRODUCTION

La complexité sans cesse croissante des circuits intégrés **VLSI**, rend leur réalisation manuelle impossible. Il a donc fallu développer des outils informatiques puissants et rapides pour l'élaboration de circuits intégrés. Ces logiciels permettent d'assurer d'une manière interactive la conception du circuit intégré depuis le cahier des charges jusqu'à l'implantation de celui ci. Les outils de **C.A.O** sont de puissance et de complexité diverses, allant du simple outil graphique d'aide au dessin, jusqu'aux chaînes de logiciels qui effectuent automatiquement des tâches très complexes.

Lors de la réalisation d'un circuit intégré, on peut distinguer plusieurs phases dans la mise au point de celui ci:

-**Simulation et conception:** lors de cette étape, le schéma du circuit à réaliser est formé à partir d'éléments de base choisis selon la fonction désirée.

-**Plan et dessin de masques**: c'est une étape fondamentale dans la conception de tout circuit intégré. Pour faire le layout d'un circuit, on doit tenir compte de la technologie utilisée(NMOS, CMOS, BIPOLAIRE...). Chaque technologie est caractérisée par un nombre de masques et des règles de dessin qu'il faut absolument respecter (dimension minimale des pistes conductrices, des zones diffusées,...)

-**Implantation du circuit** : placement des composants et routage des connexions.

-**Vérification du fonctionnement:** ceci est assuré par un simulateur électrique auquel il faut fournir non seulement la topologie du circuit (schéma d'interconnexions des transistors) mais également les dimensions de chaque transistor ainsi que les valeurs des résistances et celles des capacités en chaque noeud.

Les techniques de la **C.A.O** interviennent à tous les stades de la conception, en particulier, lors des étapes de placement des composants et de tracé des connexions.

En **VLSI**, le taux d'intégration est très élevé. Une puce peut comporter des centaines sinon des milliers de cellules que le concepteur doit placer sans introduire d'erreurs et ceci n'est possible que si l'on dispose de logiciels performants. Il existe plusieurs techniques de placement des cellules dans un circuit intégré. Parmi ces techniques: le **recuit simulé**.

Cette technique du **recuit simulé** a montré son efficacité lors des opérations de placement par rapport aux techniques les plus évoluées. Actuellement, le placement des cellules est effectué a base de réseaux neurones **(réseau de kohonen) [1]**, où chaque cellule est représentée par un ensemble de neurones. La disposition des cellules est identique à celle de **Gate array**. Les résultats obtenus n'étant pas satisfaisants, il a fallu introduire le **recuit simulé.**

Notre travail consiste a développer un logiciel de placement automatique des cellules dans un circuit intégré par l'algorithme de **recuit simulé**.

Le premier chapitre de ce document constitue un rappel sur l'environnement **C.A.O** et l'importance de ses outils. Dans le deuxième chapitre, nous présentons les différents algorithmes de placement. Le troisième chapitre est consacré à l'étude théorique de la technique de **recuit simulé** adoptée dans notre travail. Dans le quatrième chapitre, nous développons, d'une manière détaillée l'application du **recuit simulé** aux différentes configurations. Le dernier chapitre est consacré au logiciel développé.

CHAPITRE I

ENVIRONNEMENT C.A.O

CHAPITRE I

ENVIRONNEMENT C.A.O

INTRODUCTION

L'élaboration et la mise au point de nouveaux types de circuits intégrés, éléments de base des appareils électroniques modernes à usage divers, exige des concepteurs une connaissance approfondie des aspects physique, constructif et technologique.

L'élévation du degré d'intégration des circuits intégrés a nécessité l'introduction d'un stade supplémentaire de conception, celui de la conception de la structure, ayant pour but d'élaborer la structure générale du circuit à réaliser sous forme d'un assemblage d'éléments logiques et analogiques.

Les circuits intégrés modernes étant des dispositifs électroniques complexes, leur représentation schématique est donnée à deux niveaux: le premier, le plus détaillé, est celui du schéma électrique qui représente les interconnexions de composants distincts. Le second, plus général, est celui du schéma structurel (appelé encore schéma fonctionnel, organigramme ou bloc diagramme) qui représente les interconnexions d'éléments (portes) logiques distincts et de bascules (pour les circuits numériques) ou d'éléments analogiques (pour les circuits analogiques).

Ces éléments effectuent des opérations logiques (ET-NON, OU-NON et autre) ou analogique (amplification, filtrage et autres) à l'aide desquelles on peut réaliser n'importe quelle fonction numérique, analogique-numérique ou analogique.

I LES ETAPES DE CONCEPTION D'UN CIRCUIT INTEGRE [30]

Les conditions de départ du projet d'un circuit intégré (cahier des charges) contiennent la description des fonctions qu'il doit remplir et les performances exigées de ses principaux paramètres (puissance, rapidité de fonctionnement et autre).

Lors de la conception d'un circuit intégré, la base d'éléments utilisés est choisie parmi les variantes d'éléments logiques ou analogique déjà élaborées. Le stade de conception de la structure d'un circuit intégré comporte:

• **la synthèse de la structure** :au cours de laquelle on construit, à l'aide d'une base d'éléments choisis, un schéma structurel qui assure l'exécution des fonctions définies par le cahier des charges.

• **l'analyse de la structure**: au cours de laquelle on vérifie le fonctionnement correcte de la structure synthétisée dans les différentes conditions de service. Cette analyse est suivie d'un procédé d'évaluation comparative approchée des principaux paramètres de cette structure.

Partant des résultats de la comparaison des paramètres, on choisit une ou plusieurs variantes parmi les plus réussies. Si les variantes obtenues ne sont pas conformes au cahier des charges fixé, on procède à la synthèse de nouvelles variantes de structure.

• **la conception du schéma électrique du circuit à réaliser**.: Ce stade de la conception comporte:

- la synthèse du schéma électrique correspondant à la variante de la structure choisie

- l'analyse électrique du schéma obtenu ayant pour but de déterminer ses principaux paramètres électriques. Lors de cette analyse on effectue aussi une optimisation paramétrique du schéma. Celle ci consiste à déterminer les meilleures valeurs des paramètres électriques des composants du circuit.

En effet , le concepteur est amené à élaborer plusieurs variantes de schémas électriques qui diffèrent par la structure, par la base d'éléments utilisés et donc par les valeurs des principaux paramètres, au cours de la conception . Il choisit celle qui satisfait le plus au cahier des charges du projet.

Le stade suivant, celui de la conception constructive et technologique, comprend les étapes du choix ou de l'élaboration du processus technologique nécessaire à la fabrication du circuit, et l'élaboration de sa topologie conformément au schéma électrique conçu. Les differentes étapes de la concéption sont illustrées sur la figure (Fig I.1).

II.UTILISATION DE CALCULATEURS

La conception des circuits intégrés est caractérisée par une large utilisation de calculateurs électroniques. la nécessité du recours aux ordinateurs s'explique par les possibilités limitées de la simulation d'experiences des circuits intégrés. Du fait que l'obtention des échantillons prototypes expérimentaux de circuits nécessite d'effectuer tout le processus pénible et onéreux de concéption et de fabrication qui peut durer plusieurs mois . C'est pourqoi la simulation d'expérience est généralement remplacée par la simulation sur ordinateur.

En effet, la complexité croissante des circuits intégrés rend leur réalisation manuelle difficile. Il a donc fallu développé des outils informatiques propres à l'élaboration de ces circuits.

La **C.A.O** intervient aux différentes étapes de conception d'un circuit intégré, notamment dans le placement et le routage des cellules. En technologie **VLSI**, le concepteur est amené a placer des centaines sinon des milliers de cellules dans une seule puce en essayant d'optimiser la surface et la longueur des connexions. Ceci n'est possible qu'aux moyens de logiciels performants.

Selon les domaines de fonctionnement, et après avoir défini le schéma bloc du circuit, le concepteur à la possibilité de réaliser son circuit à base des configurations suivantes:

RESEAU PREDIFFUSE (GATE ARRAY): le concepteur utilise des cellules programmables. Une cellule est une entité fonctionnelle, qui peut aller d'une simple porte logique AND jusqu'à une unité de multiplication. Le concepteur peut les personnaliser à une application particulière, en ajoutant les connexions là où il faut.

STANDARD CELL: ce sont des cellules de même hauteur et de largeur différentes. Chaque cellule peut réaliser une fonction bien précise. Des bibliothèques contenant toutes les informations relatives à ces cellules sont à la disposition des concepteurs.

CUSTOM DESIGN: si le volume de production est très élevé et la complexité est modérée, le concepteur se trouve obligé de dessiner lui même ces cellules qui seront de tailles différentes.

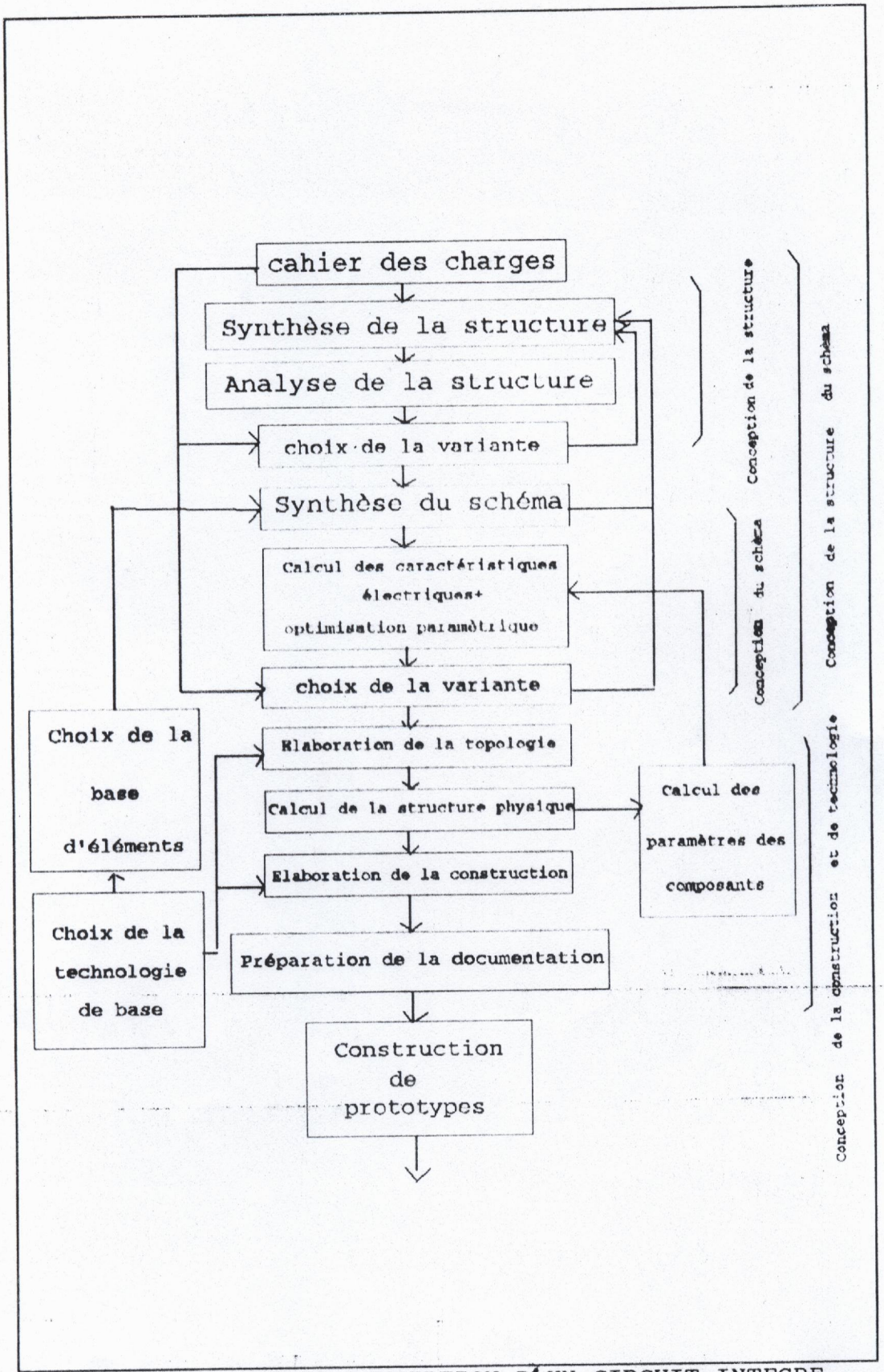


Fig.I.1 ETAPES DE LA CONCEPTION D'UN CIRCUIT INTEGRE

CHAPITRE II

TECHNIQUES DE PLACEMENT DES CELLULES DANS UN CIRCUIT INTEGRE

CHAPITRE II

TECHNIQUES DE PLACEMENT DES CELLULES DANS UN CIRCUIT INTEGRE

INTRODUCTION

Le problème de placement peut être défini de la façon suivante: étant donné **N** éléments avec leur liste d'équipotentielles, il s'agit de trouver un placement optimisé pour ces éléments définis soit par **M** positions **(M>N)**, soit par une surface **S (S ≥ S1 U S2 U S3.....USN**, où les **Si** sont les surfaces de chaque élément). Cette optimisation concerne le tracé ultérieur des connexions: il s'agit de trouver un placement tel que ce tracé soit possible et de bonne qualité.

Pour apprécier la qualité d'un placement **P**, on lui associe une valeur **Ci(P)** appelée coût. La fonction de coût peut tenir compte de plusieurs contraintes, à savoir: la longueur des connexions, la surface et la forme du circuit, le nombre de croisements des connexions ainsi que leur densité. Cependant le critère le plus couramment utilisé est la longueur totale des connexions, car il semble refléter aux mieux toutes les contraintes et faciliter le routage par la garantie de proximité des composants fortement connectés.

La recherche d'un placement optimal s'effectue le plus souvent en deux étapes: Création d'un placement initial, puis son amélioration.

I.ETAPES DE PLACEMENT

I.1. Placement initial

Cette étape consiste en fait à dégrossir le problème. En effet, il est extrêmement difficile, voire impossible, de réaliser en une seule étape un placement respectant toutes les contraintes imposées, si l'on ne possède aucune base solide de départ. Le but du placement

initial est de déterminer un placement de qualité suffisante afin de faciliter l'étape d'amélioration itérative.

L'étape du placement initial est généralement réalisée de façon constructive, c'est à dire que l'on ne cherche pas à manipuler simultanément tous les composants, mais on les considère successivement, évaluant à chaque fois la meilleure association place/élément.

Dans une telle approche, un ordonnancement est indispensable et c'est à partir de cet ordre que seront placés les éléments et que les places seront attribuées. On voit apparaître deux types d'ordonnancements possibles:

•**Un ordonnancement topologique**: où l'ordre est établi sur les places disponibles du circuit. Pour chaque emplacement on place l'élément qui respecte aux mieux les contraintes, ou tout au moins celui qui permet d'optimiser la fonction de coût.

•**Un ordonnancement logique**: consiste à définir un ordre de placement sur les composants, cet ordre peut être défini à partir de nombreux critères: le nombre de connexions par rapport aux éléments déjà placés, l'éloignement des plots, le nombre de connexions aboutissant à chaque élément etc...

I.2. Amélioration du placement [4][5]

Lors de cette étape, on applique des perturbations au placement initial qui peut être constructif ou aléatoire, de façon à améliorer globalement le placement. Ces perturbations sont à choisir dans l'ensemble des transformations géométriques autorisées.

Selon la règle de sélection des éléments à perturber et celle de la succession des mouvements, on distingue deux type de méthodes: les méthodes stochastiques et les méthodes déterministes.

•**Méthodes stochastiques**: dans cette approche, les transformations locales (symétrie, rotations, translation) et les transformations globales (échange par paires, translations) sont choisies aléatoirement et appliquées à des éléments choisis au hasard. Une des méthodes stochastiques et celle du **"recuit simulé".**

•**Méthodes déterministes**: dans cette approche, les transformations sont appliquées systématiquement à chaque élément et une perturbation n'est acceptée que si elle conduit à une diminution de la fonction de coût. Un échange peut être soit un échange par paires, soit

un échange mettant en jeu plusieurs éléments.

II. TECHNIQUES DE PLACEMENT

Les règles de sélection et de positionnement des éléments dans un circuit permettent de définir plusieurs techniques de placement. Parmi ces méthodes on peut citer:

II.1 Placement barycentrique ou par agrégat [5]

Le placement par agrégat consiste à placer un composant au centre du circuit, puis à remplir les places contiguës en progressant selon une spirale si l'ordre topologique est complètement défini (fig II.1.a), ou selon des anneaux concentriques si un ordre sur des classes de places a été établi (fig II.1.b).

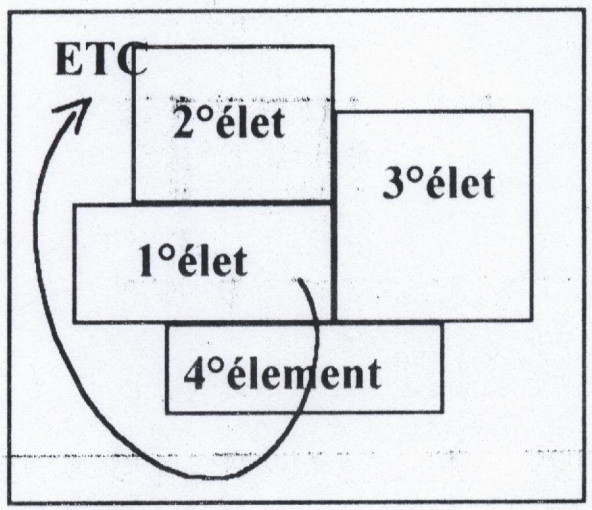


Fig II.1.a Progression selon une spirale

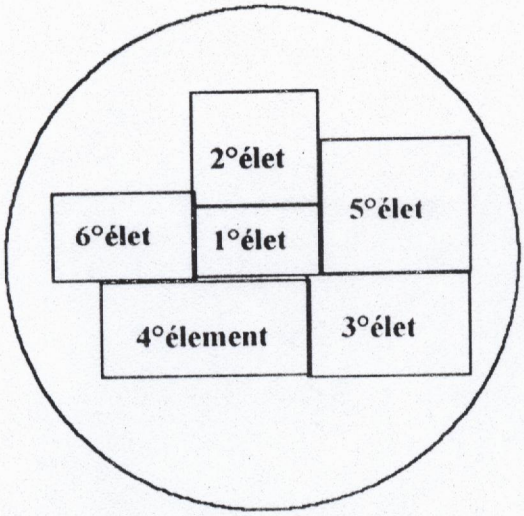


Fig II.1.b Progression selon des anneaux concentriques

La méthode barycentrique consiste à placer successivement les composants, de manière à réduire la force d'attraction qui existe entre les éléments déjà placés et l'élément candidat. Pour cela, chaque nouveau bloc, est placé le plus près possible du barycentre des éléments qui lui sont connectés. Ceci est illustré par la figure (fig II.1.c).

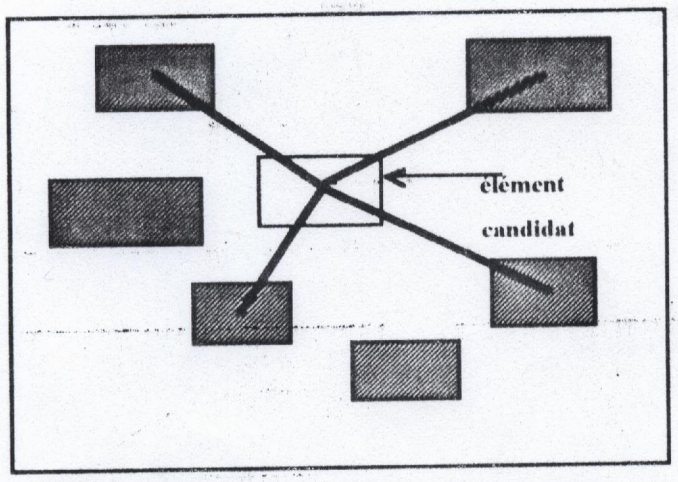


Fig II.1.c Placement constructif barycentrique

II.2. Placement par la méthode Min_cut [6][10][16]

Dans ce cas, il faut trouver une partition efficace pour un circuit constitué de **N** composants reliés par **I** interconnexions qui devront traverser la frontière entre les parties. La résolution du problème est normalement assujettie à la nécessité d'obtenir un équilibre relatif entre les tailles des deux parties: il faut évidemment éviter la solution qui consisterait à placer tous les composants du même côté (Fig II.2.a).

La méthode **Min_cut** est une méthode heuristique qui permet d'obtenir un placement de bonne qualité pour un temps de calcul relativement faible.

5	3	1	4
2			
6			

Fig II.2.a séquence de repartition permettant d'obtenir un placement fin

Cette méthode est basée sur le gain d'un composant qui représente la variation du nombre total d'interconnexions résultant du transfert d'un composant d'un côté à l'autre de la frontière (fig II.2.b). La variable d'équilibre **e**, détermine l'objectif à atteindre en ce qui concerne la taille relative des deux parties. La contrainte d'équilibre est représentée par l'équation suivante:

$$eC - d < |A| < eC + d \qquad (2.1)$$

$$|A| + |B| = C \qquad (2.2)$$

où $|A|$ et $|B|$ sont les cardinalités des deux parties.

Un composant peut se voir assigner une position fixe ou se deplacer librement. Au début d'une itération, les composants libres sont triés en fonction de leurs gains.

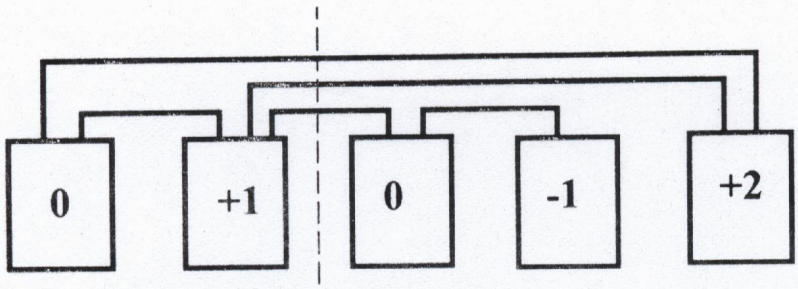


Fig II.2.b Gain des composants pour une répartition donnée

En utilisant la méthode **Min_cut**, on déplace un à un les composants libres selon l'ordre décroissant des gains. Pour empêcher la formation de boucle sans fin, on fait en sorte qu'un composant libre qui a été déplacé devienne fixe pour le reste de l'itération, et qu'il soit inséré dans la liste des composants libres pour l'itération suivante. On doit calculer le total des gains à chaque étape et mettre à jour la valeur des gains de chacun des composants .

Fidduccia et **Mattheyses [17]** proposent une méthode efficace qui permet de gérer l'ensemble des valeurs des gains et montrent que la complexité de cette méthode varie de façon linéaire en fonction de la taille du problème.

Le résultat d'une itération est le déplacement pour lequel le gain était maximal. Signalons que la méthode permet d'effectuer des déplacements de gain négatif, et par conséquent, de sortir de minima locaux.

Les déplacements doivent par ailleurs respecter les contraintes d'équilibre. Si le déplacement du composant dont le gain est maximal entraîne un déséquilibre, on le laisse simplement dans la liste des composants libres jusqu'à ce que l'équilibre permet son déplacement.

Bien que le nombre maximal d'itérations requises pour qu'on obtienne une bonne solution n'ait pas été déterminé de façon formelle, on considère que trois ou quatre itérations par partition sont nécessaires en pratique.

Le principal avantage de la méthode **min_cut** est sa rapidité d'exécution, sa principale faiblesse, en ce qui concerne le placement des composants, provient du fait que la congestion du routage résultant n'est pas prise en considération. Il peut donc arriver que le placement produit soit amélioré aux dépens de la facilité subséquente à effectuer le routage.

II.3. Placement relatif par la théorie des graphes [16][26]

II.3.1. Introduction

Comme pour toute implantation automatique de circuit **VLSI** l'opération nécessite quatre étapes:

- Placement relatif des cellules.
- Placement final des cellules.
- Routage global des interconnexions.
- Routage à canal détaillé des interconnexions.

Une des méthodes les plus utilisées dans le placement relatif est la méthode de placement forcé direct ou encore **"théorie des graphes"**.

Dans cette méthode on prend comme modèle géométrique pour les réseaux et les cellules des rectangles circonscrits. Ce modèle permet une simulation typique lors de la procédure de routage. Pour une bonne estimation de la longueur des connexions, celle ci est donnée par le demi périmètre du rectangle circonscrit. En effet, ceci n'est valable que si le réseau en question possède deux ou trois broches alors que le nombre de ceux ayant plus de trois doit être très faible **(<25%)**.

II.3.2 Formulation mathématique du problème

Un réseau est défini par un ensemble de **m** broches reliées entre elles. **Nk={Pk1,Pk2,Pk3,....,Pkm}** où **Pkj** représente la broche définie à son tour par ses coordonnées **(Xkj,Ykj)**. Chaque réseau est limité par un rectangle appelé **domaine du réseau**, dont le demi périmètre est donné par:

$$HPCR(k) = (V_k - U_k) + (V'_k - U'_k) \qquad (2.3)$$

Uk(uk') et **Vk(vk')** sont respectivement le minimum et le maximum des **Xkj(Ykj)**. Chaque cellule est représentée par un rectangle dit **domaine de cellule**, dont les dimensions sont supérieures à celles de la cellule, lui offrant ainsi un certain degré de liberté (Fig II.3).

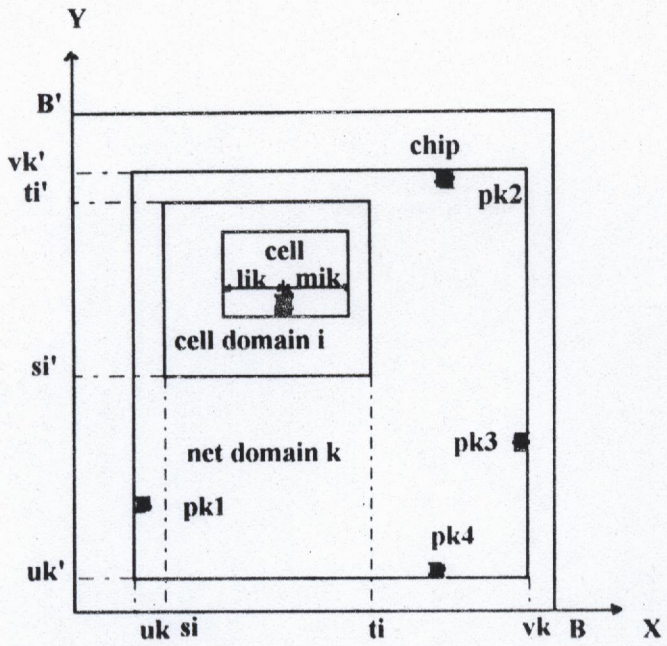


Fig.II.3 Net domain and cell domain

Le problème de placement peut être établi comme suit: pour une surface donnée de la puce de dimensions **B** et **B'**, il s'agit de trouver les positions pour tous les domaines et pour toutes les broches de manière à avoir:

$$\sum_{k} C_{xk}\left(V_k - U_k\right) + C_{yk}\left(V'_k - U'_k\right) = \text{minimum} \qquad (2.4)$$

Cxk et **Cyk** sont les poids respectifs des réseaux donnés par le constructeur. Chaque domaine doit vérifier certaines contraintes représentées par les inégalités suivantes :

$$V_k - U_k \geq d_k \geq 0 \qquad (2.5)$$

dk est la largeur horizontale minimale du domaine de réseau choisie par le constructeur.

$$t_i - S_i \geq W_i \geq 0 \qquad (2.6)$$

wi est la largeur de la cellule, **si** et **ti** sont ses abscisses.

$$U_k - S_i \leq L_{ik} \quad \text{et} \quad V_k - t_k \geq -m_{ik} \qquad (2.7)$$

lik et **mik** sont les positions de la broche relatives à la cellule **i.**

$$U_k - p_i \leq n_{ik} \quad \text{et} \quad V_k - q_i \geq -o_{ik} \qquad (2.8)$$

où **pi**, **qi** coordonnées des cellules fixes, **ni** et **oi** identiques à **li** et **mi**.

le problème de placement à deux dimensions peut être traité en deux problèmes indépendants de directions respectives **X** et **Y**.

II.3.3 Formulation du problème par la théorie des graphes

a/graphe du placement .

Les inégalités **(2.5)**, **(2.6)**, **(2.7)** et **(2.8)** sont facilement modèlisées en un graphe direct **G=(V,A)** appelé graphe de placement. **V** et **A** sont respectivement les sommets et les arcs du graphe. Chacun des côtés verticaux des domaines est représenté par un sommet **vj** $\in$ **V**. Le poids du sommet **w(vj)** est donné par la variable correspondante **(uk, vk, si ou ti)**.

On appelle **sommet libre**, un sommet dont le poids n'est pas encore fixé, contrairement aux cellules **d'entrée-sortie** qui ont un poids fixe (dans notre cas **pi** et **qi**). Chaque paire de sommet **(vj,vk)** $\in$ V est définie par:

$$W(V_k) - W(V_j) \geq b \qquad (2.9)$$

b=(2.5),...,(2.8). La différence des poids est appelée longueur d'onde λ et la valeur limite b est **λmin**. le sous graphe **G** formé par les sommets fixes et les arcs de longueur fixe est appelé **"pad tree"**.

b/problème d'optimisation par la théorie des graphes

Soit un problème de placement modèlisé en un graphe **G=(V,A)**, les valeurs minimales de tous ses arcs **λmin(Ai)** et leurs poids correspondants **c(Ai)** sont connus. Optimiser le placement revient à trouver pour tous les sommets des poids de manière à avoir :

$$\sum_{Ai \in A} c(Ai)\,\lambda(Ai) = \textbf{minimum} \qquad (2.10)$$

II.3.4 Algorithme de la théorie des graphes

a/aperçu sur l'algorithme

L'idée de base de cet algorithme est de trouver pour tous les sommets un poids relatif à celui du sommet gauche de la puce qui permet de minimiser le coût. Le graphe de placement où chaque arc atteint sa valeur minimale est appelé **"spanning tree".**

L'algorithme s'effectue en deux phases: une phase de construction et une phase d'optimisation. Chaque sommet libre forme un arbre.

b/définitions

Pour la compréhension de la méthode de la **théorie des graphes**, certaines définitions de base s'avèrent nécessaires:

Définition 1: Soit un sommet **vi**, un arc direct du graphe est dit branche droite (gauche) si elle démarre (arrive) de ce sommet. On note par **RB(vi)** et **LB(vi)** l'ensemble de ces branches, l'arc indirect correspondant appartient à l'arbre. De même un arc direct est dit liaison droite (gauche) du sommet s'il démarre (arrive) de celui ci. **RL(vi)** et **LL(vi)** sont les ensembles correspondants, l'arc indirect n'appartient pas à l'arbre.

Définition 2: Le degré du sommet d'arbre **g(vi)** est le nombre de branches incidentes à ce sommet.

Définition 3: la quantité **H(T)**, dite **"hull"** de l'arbre **T** ayant **V(T)={..,vi,..}** ensemble de sommets est donnée par:

$$H(T) = \sum_{v_i \in V(T)} \left[\sum_{A_i \in RL(v_i)} c(A_i) - \sum_{A_k \in RR(v_i)} c(A_k) \right] \qquad (2.11)$$

Définition 4: Soit l'arbre T divisé en deux sous arbres **STr** et **STl**, l'arc **Ai** se transforme de branches en liaisons. le gain de la branche Ai droite (gauche) noté par **RG(Ai)** (LG(Ai)) est la valeur de **H(Ai)** du sous arbre droit (gauche).

c/phase de construction

Son but est de construire le **"spanning tree"**. Au début de cette phase, on attribue à tous les sommets la valeur du poids du sommet gauche ou droit de la puce; ceci donne aux arcs des longueurs supérieures ou égales aux valeurs minimales. cette étape s'effectue en quatre procédures: recherche d'arbre propre, déplacement, fusionnement et recalcul des hulls.

•**recherche d'arbre**: La réduction du coût n'est possible que si l'arbre Ti est déplacé vers la droite si son **H(Ti)>0** ou vers la gauche si son **H(Ti)<0**. l'arc choisi est celui dont **|H(Ti)|** est la plus élevée.

•**déplacement**: L'arc choisi est déplacé aussi loin que possible. La valeur maximale **ΔX** du déplacement est donnée par

$$\Delta X = \left| \lambda(A_k) - \lambda_{min}(A_k) \right| \qquad (2.12)$$

•**fusionnement et recalcul des hulls**: par fusionnement les liaisons se transforment en branches. La nouvelle valeur de **H(Ai)** est la somme des **H(Aj)** fusionnées.

d/phase d'optimisation

Si la solution optimale n'a pas été atteinte en phase de construction, une phase

d'optimisation s'avère nécessaire.Cette phase ne traite que les sommets ayant **g(vi)=1**. Elle se fait en quatre étapes: recherche d'arc, division, déplacement, fusionnement et recalcul des gains. Toutes ces procédures sont identiques à celles de la phase de construction. La division d'une branche n'est possible que si le **RG(Ai)>0** ou **LG(Ai)<0**.

Critère d'arrêt: on décide de l'arrêt de cette phase lorsque tous les arcs du **"spanning tree"** possèdent des gains droits négatifs et des gains gauches positifs.

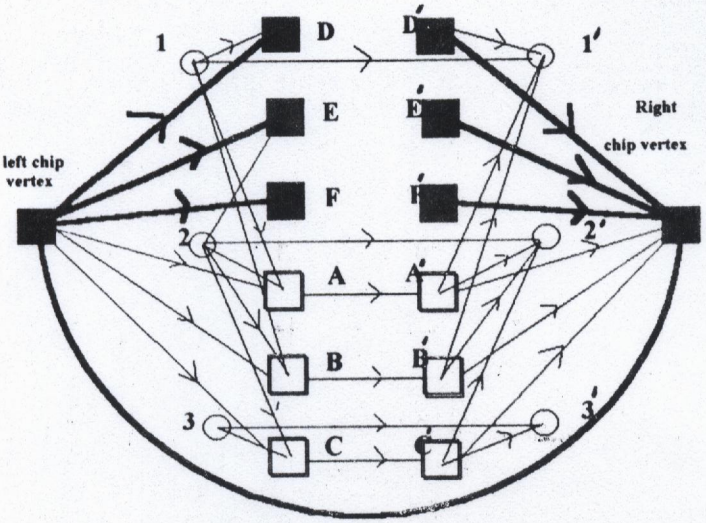


Fig II.4 Exemple d'un placement par la théorie des graphes

La figure II.4 montre un exemple de placement par la théorie des graphes. Cet exemple présente trois cellules **A**, **B**, **C**, trois cellules d'entrée-sortie **D**, **E**, **F** et trois réseaux **1**, **2**, **3**. Le **"pad tree"** est représenté par des traits forts.

En général, pour un circuit ayant **Nn** réseaux, **Nc** cellules, **Ni/o** cellules d'entrée-sortie et **Np** broches, le nombre de sommets **Nv** et d'arcs **Na** du graphe sont donnés par:

$$N_v = 2(N_n + N_c + N_{i/o}) + 2 \qquad (2.13)$$

$$N_a = (N_n + N_c) + 2(N_p + N_{i/o}) + 2(N_c + N_{i/o}) + 1 \qquad (2.14)$$

II.4. Placement par la méthode G.F.D.R [4]

II.4.1 Introduction

La méthode **G.F.D.R.**(generalized force directed relaxation) est une méthode qui utilise des perturbations très complexes. Nous rappelons qu'une **zone de relaxation** est définie comme étant une région où la résultante des forces d'attraction qui s'exercent entre un élément placé dans cette région et les éléments qui lui sont connectés, est nulle. Les forces d'attraction sont constituées par les connexions entre ces éléments.

La méthode **G.F.D.R.** utilise une procédure de recherche non exhaustive où l'ensemble des blocs à échanger n'est pas choisi aléatoirement, mais pris dans des régions bien définies.

II.4.2 Mécanisme des mouvements élémentaires dans la méthode G.F.D.R

La procédure de recherche d'un état optimal est illustrée par un **"arbre"** présenté par la figure (fig II.5), où chaque noeud représente un bloc et chaque segment, un essai de transformation. Le sommet de l'arbre **A**, qui est choisi arbitrairement, est appelé bloc primaire. A partir de là, on détermine la position **"médiane"** de **A** qui est définie comme étant une position qui minimise la longueur du routage associé. Le bloc se trouvant dans cet endroit est dit bloc **médian** (**B** dans notre exemple).

Ensuite, nous déterminons les **ε-1** voisins qui sont les plus proches par rapport à **B** (dans notre cas, **ε = 3**), soit **C** et **D**; on dit aussi que l'ensemble **{B,C,D}** est le voisinage d'ordre **3** de la médiane du bloc **A**. On essaie d'abord les échanges entre les blocs suivants: **A→B** ; **A→C** ; **A→D**. Un échange est accepté s'il y'a une réduction de la longueur totale. Dans le cas où cette réduction se produit plusieurs fois, on en retient la meilleure.

En revanche, si aucun des échanges n'est accepté, on passe aux transformations d'ordre **3** (**λ=3**, **λ** étant le nombre de boîtiers, dans un échange), et pour cela on calcule d'abord la médiane de **B** et son voisinage d'ordre **3**, soit**{E,F,G}**. Ensuite on essaie successivement les transformations **A→B→E** ; **A→B→F** ; **A→B→G**, et on choisit le meilleur échange, c'est à dire celui correspondant à une longueur totale de routage plus faible. Si la transformation choisie conduit à une réduction de la fonction de coût, on effectue alors cette transformation, et l'on choisit un autre bloc primaire. Dans le cas contraire, la même opération effectuée pour le bloc **B**, est faite pour **C** et si c'est nécessaire pour **D** aussi. Ainsi décrite, l'opération de transformation est poursuivie jusqu'à ce que l'échange aboutisse à une diminution de la fonction objectif, sinon jusqu'à ce que le nombre **λ** de blocs impliqués dans un mouvement élémentaire, atteigne le nombre maximal **λ***, introduit comme paramètre d'entrée.

Le choix des paramètres **λ**, **λ***, la configuration initiale et le nombre total de blocs utilisés, a une influence directe sur le type de solution obtenue. Par exemple l'augmentation simultanée du nombre de blocs dans un mouvement élémentaire, et le voisinage d'un bloc médian permettent d'améliorer considérablement la qualité des solutions finales. Les différentes études expérimentales faites sur le choix de ces paramètres sont explicitées à la référence **[4].**

Il est vrai que la méthode d'amelioration itérative utilisant des mouvements élémentaires complexes, aboutit à des solutions finales meilleurs que si l'échange s'effectuaient entre deux blocs seulement. Cependant, en comparaison avec l'algorithme du **recuit simulé**, la méthode **G.F.D.R** est moins puissante. Par exemple, sur **273** essais effectués **27** seulement ont abouti au minimum absolu, alors qu'on choisissant convenablement les valeurs des différents paramètres, et pour le même nombre d'essais, le **recuit simulé** a donné **113** essais qui ont conduit au minimum absolu.

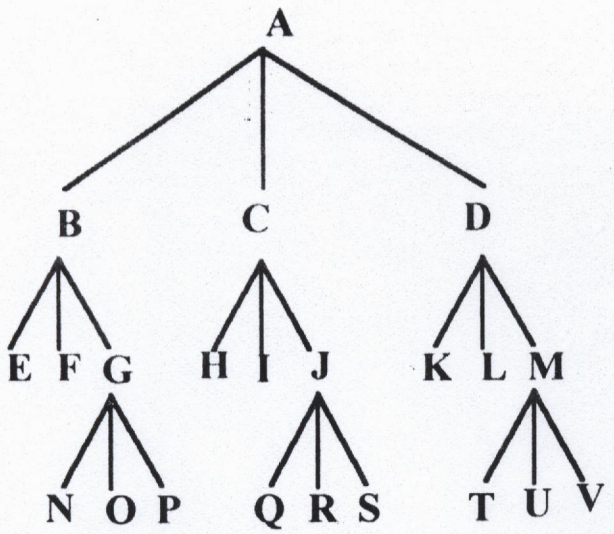


Fig II.6 Mecanisme des mouvements élémentaires dans la méthode G.F.D. R

CONCLUSION.

Les algorithmes présentés permettent de résoudre le problème de placement. Cependant, la méthode du recuit simulé adoptée dans le cadre de notre travail reste la plus utilisée et la plus efficace. Le detail et l'application de celle ci seront developpé dans les chapitres qiu suivent.

CHAPITRE III

TECHNIQUE DE PLACEMENT PAR RECUIT SIMULE

CHAPITRE III

TECHNIQUE DE PLACEMENT PAR RECUIT SIMULE

INTRODUCTION

C'est en 1983 qu'une nouvelle méthode d'analyse et d'optimisation des systèmes complexes a vu le jour: le **"recuit simulé"**. C'est une méthode qui, pour trouver des solutions optimales aux problèmes d'un système complexe, l'assimile à un système physique comportant un très grand nombre de particules en interaction. Par exemple, dans le cas de la **C.A.O.** des circuits intégrés, le passage d'une implantation très désordonnée, à un circuit où les composants sont bien alignés et où les connexions sont rectilignes, présente des points communs avec l'évolution de la matière condensée, d'un état liquide (état **désordonné**), vers un état solide (état **ordonné**). Cette évolution doit se faire d'une façon très lente, par abaissement de la température.

L'utilisation de la **"température"** dans un problème pratique d'optimisation, tel que la conception des circuits intégrés, permet de simuler le phénomène physique, et de considérer la température comme un paramètre de contrôle pour la méthode.

I. PRESENTATION DE LA METHODE DU RECUIT SIMULE

I.1 Principe général de la méthode [4][5][9]

Les problèmes posés aux ingénieurs par les technologies modernes s'expriment de plus en plus fréquemment en terme d'optimisation combinatoire: les objectifs désirés doivent être exprimés sous la forme d'une fonction objectif, appelée aussi fonction de coût, que l'on cherche à minimiser par rapport à un ensemble de paramètres. Cependant, plus le nombre de degrés de liberté augmente, plus il est difficile, voire impossible, d'arriver à l'optimum d'une telle fonction, par une méthode classique. En effet, pour certains problèmes de complexité élevée, le nombre de minima locaux croit exponentiellement avec le nombre de degrés de liberté [5].

Un grand nombre de problèmes d'optimisation appartient à la classe des problèmes

NP-Complets, c'est à dire ceux pour lesquels aucun algorithme ne conduit à l'optimum en un temps polynomial en fonction des dimensions du problème. En pratique, une solution approchée suffit: c'est pourquoi on utilise souvent des méthodes heuristiques qui permettent en un temps raisonnable, d'atteindre des solutions **quasi-optimales**.

Les méthodes classiques d'amélioration itérative entrent dans cette catégorie. Elles consistent, à partir d'une configuration initiale, à effectuer une modification élémentaire, et n'accepter la nouvelle configuration que si cette perturbation entraîne une diminution de la fonction objectif. Il est clair qu'un tel algorithme ne conduit pas, en général, au minimum absolu, mais seulement à un minimum local (fig III.1).

S.Kirkpatrick [18] a montré qu'il existe une analogie entre un système physique et un problème d'optimisation: la nature arrive facilement a résoudre des problèmes d'optimisation ayant un espace de configuration très complexe. Cette analogie a conduit à une nouvelle méthode dite méthode de **"recuit simulé"** qui évite les minima locaux, en autorisant des augmentations de la valeur de la fonction de coût **"hill climbing".**

I.2 Analogie entre le recuit réel et le recuit simulé

L'orsqu'on abaisse progressivement la température d'un système physique, il évolue lentement vers son état fondamental, celui d'énergie la plus basse. En revanche, si l'on abaisse rapidement la température, on fait subir au système une **trempe**: Il atteint un état **métastable**, d'énergie non minimale, correspondant à un minimum local.

Ce passage de la matière d'un état désordonné (état **"liquide"**) vers un état ordonné (état**"cristallin"**) a donné l'idée au chercheurs **[13],[18],[19],[21],[22]** d'utiliser ce phénomène physique en vue d'optimiser des systèmes très complexes.

Dans la technique du **recuit simulé**, on associe une énergie à chacun des états possibles du système et l'on definit un paramètre de contrôle, **"la température"**. La température, dans un problème d'optimisation doit avoir le même rôle que dans un système physique: elle doit conduire vers l'état optimal si elle est abaissée de façon lente et bien contrôlée, et vers un optimum local si elle est abaissée brusquement.

La méthode du **recuit simulé** consiste à simuler l'évolution, en fonction de la température, d'un système obéissant à la statistique de **Boltzmann**: la probabilité pour un système donné, en équilibre thermodynamique à la température **T**, de posséder une énergie

donnée, soit **Es**, est proportionnelle au facteur de Boltzmann : **exp (-Es/kT).**

k: étant la constante de **Boltzmann.**

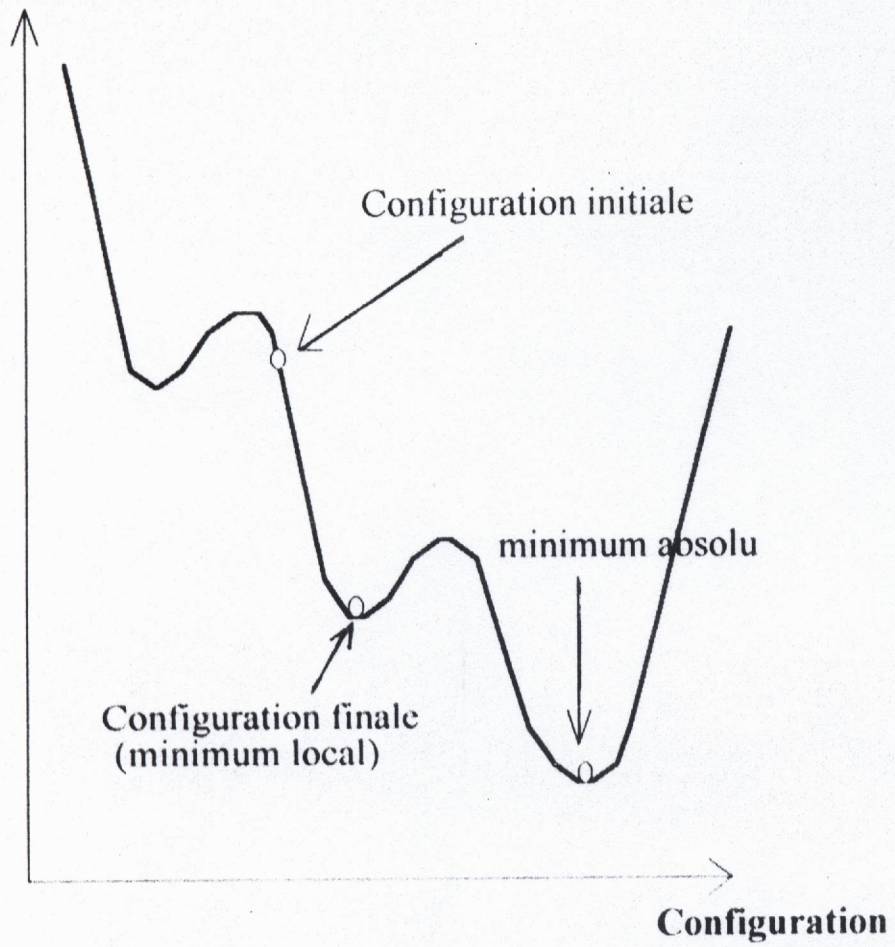


Fig III.1 Optimisation par une méthode itérative classique

I.3 Description de l'algorithme de Metropolis

L'algorithme de **Metropolis [4]**, permet d'engendrer une séquence d'états formant une **chaîne de Markov**, telle que les énergies de ces états soient distribuées selon la statistique de **Boltzmann**. Soit un état **P**, d'énergie **Ei**, auquel on applique une perturbation. Cette modification de l'état initial fait passer le système à un état **j** voisin; le système voit son énergie varier d'une quantité **ΔW**:

$$\Delta W = E_j - E_i \qquad (3.1)$$

La particularité du recuit simulé, par rapport aux méthodes classiques, réside dans le test d'acceptation de **Metropolis** qui est le suivant:

i- si **ΔW<0**, la configuration est acceptée et l'on perturbe de nouveau le système.
ii- si **ΔW>0**, la configuration est acceptée mais avec une probabilité égale à
exp(-ΔW/T).

Le paramètre **T** est la température: quand **T** est élevée, pratiquement toutes les modifications sont acceptées: le système réalise une marche au hasard dans l'espace de tous les états possibles. Au fur et à mesure que la température diminue, **"l'agitation thermique"** s'affaiblit, et les états acceptés se font de plus en plus rares: on dit que le système se **"fige"**. On voit que ce processus d'évolution permet d'extraire un système d'un minimum local.

Pour se ramener à un algorithme déterministe classique, il suffit de fixer la température à zéro: ainsi, le système n'autorise que les baisses des valeurs de la fonction de coût. L'algorithme de **Metropolis** est illusté sur la figure (Fig III.2)

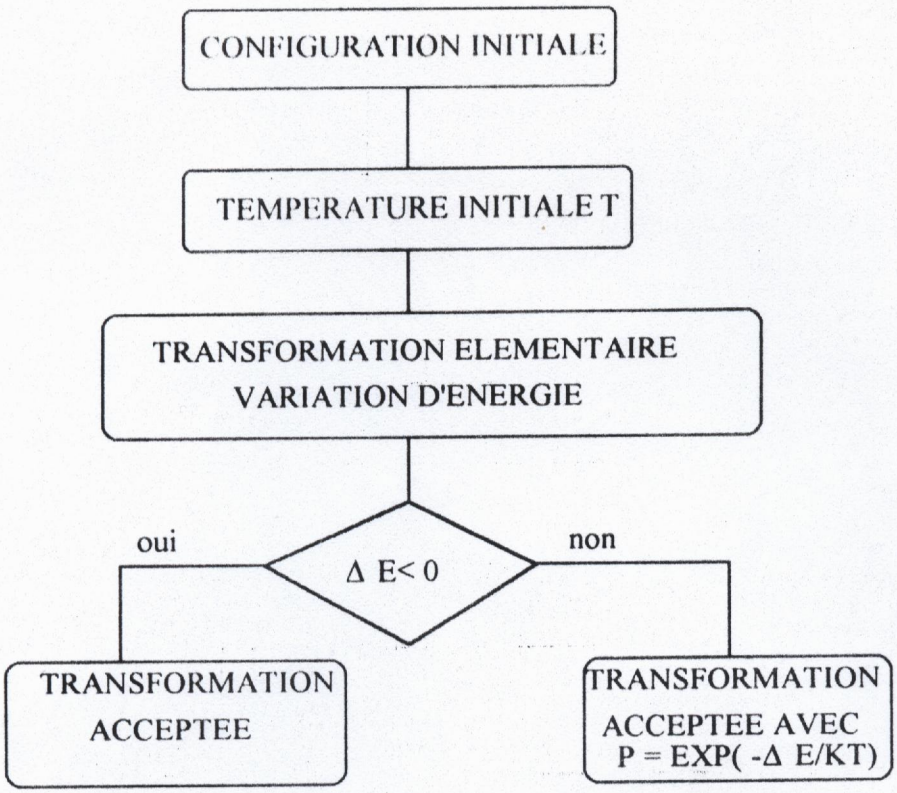


FIG III.2 TEST DE METROPOLIS

I.4 Principales approches théoriques .

Pour déterminer l'efficacité de la méthode du **recuit simulé**, il faudrait établir les conditions de convergence de l'algorithme et étudier tous ses paramètres et ses variantes.

Beaucoup de travaux théoriques ont été effectués dans ce sens; l'archetype du problème d'optimisation combinatoire est celui du voyageur de commerce: étant donné un certain nombre de villes, l'objectif du voyageur est de passer une fois et une seule par chaque ville, tout en minimisant le chemin total parcouru.

Les paramètres les plus importants qui interviennent dans le fonctionnement et l'efficacité de l'algorithme sont: la topologie de l'espace des configurations, les règles d'acceptation et le programme de recuit.

I.4.1 Espace des configurations

Reprenons l'exemple du problème du voyageur de commerce: si l'on désigne par **N** le nombre de villes, le nombre de tournées distinctes possibles pour le voyageur de commerce est égal à **(N-1)!/2**. Cet ensemble de configurations possibles constitue ce qu'on appelle l'espace des configurations.

A chaque configuration correspond une énergie donnant la longueur de la tournée, formant ainsi un paysage d'énergie qui peut présenter un certain nombres de vallées plus ou moins profondes et plus ou moins proches. Ces vallées correspondent à des minima locaux. Si le paysage d'énergie ne présente qu'une seule vallée, la résolution du problème ne nécessite qu'un algorithme d'itération classique. De ce fait, résoudre un problème d'optimisation par la méthode du **recuit simulé** n'a de sens que si le paysage d'énergie comporte des minima locaux.

L'allure de l'espace des configurations dépend essentiellement du choix de la fonction de coût et des perturbations élémentaires: en effet, deux configurations sont voisines dans l'espace des états si l'on peut passer de l'une à l'autre par une perturbation élémentaire; le choix de ces perturbations détermine donc la topologie de l'espace des configurations. En revanche, la solution finale désirée dépend essentiellement du problème traité.

Il a été montré que la convergence de l'algorithme est soumise à deux conditions, à savoir: la **réversibilité** (le changement inverse d'un changement pèrmis doit être autorisé) et la **connexité** de l'espace des configurations (tout état du système peut être atteint de n'importe quel état après un nombre fini de mouvements élémentaires). Pour le temps de calcul, il a été montré que c'est la topologie de l'espace qui joue un rôle déterminant.

Pour un problème industriel, l'objectif n'est pas de trouver l'optimum, mais seulement une valeur approchée de cet optimum en un temps raisonnable; c'est pourquoi cette méthode a connu des succès importants dans plusieurs applications industrielles **[4][5]**.

I.4.2 Règles d'acceptation

le principe de la méthode thermodynamique repose sur le fait que pour s'extraire d'un minimum local, on autorise les perturbations qui entraînent des augmentations d'énergie avec une probabilité déterminée par la statistique de **Boltzmann**, c'est à dire avec une probabilité égale à **$\exp(-\Delta W/T)$**. **T** et **ΔW** sont respectivement le paramètre de contrôle et la variation d'énergie du système physique après perturbation. A très basse température, cette règle

d'acceptation rend la méthode inefficace, car le taux de configurations acceptées devient très faible.

L'efficacité de la méthode peut être améliorée, à basse température, en utilisant une technique dite de **thermostat** qui consiste à établir une liste des changements possibles, et à calculer pour chacun d'entre eux, l'effet qu'il aurait sur la fonction de coût et l'exponentielle de Boltzmann. Le choix du changement résulte d'un tirage aléatoire, dans la liste, d'une configuration affectée de son poids de **Boltzmann**. Cependant, cette méthode nécessite une capacité de mémoire plus grande et elle n'est envisageable que si le nombre de mouvements élémentaires n'est pas très élevé.

I.4.3 Programme de recuit

Le programme de recuit utilise comme modèle mathématique les **chaînes de Markov**. Un problème d'optimisation combinatoire est caractérisé par l'ensemble des configurations possibles **S** et par une fonction de coût **C : S →R**, qui, à chaque configuration ou état **i**, attribue une valeur réelle **C(i)**. Le but de l'optimisation est de trouver la configuration **i0** vérifiant la relation suivante :

$$C(i_0) = \min[\, C(i) \;/\; i \in S]$$

Tij(θ) est la probabilité de transition de l'état **i** vers l'état **j** à la température θ. Si la température est maintenue constante, cette probabilité est stationnaire et la **chaîne de Markov** est dite homogène. La probabilité de transition peut être définie comme suit :

$$T_{ij}(\theta) = A_{ij}(\theta).P_{ij} \quad \text{si} \quad i \neq j \qquad (3.3)$$

$$\text{et} \quad T_{ij}(\theta) = 1 - \sum_{k} P_{ik}.A_{ik}(\theta) \quad \text{si} \quad i = j \qquad (3.4)$$

Où **Pij** définit la probabilité de perturbation du système d'un état **i** vers un l'état **j**, et **Aij** la probabilité d'acceptation de la configuration **j** quand on est dans la configuration **i** à la température θ.

$$\text{avec} \quad \lim_{\theta \to \infty} A_{ij}(\theta) = 1 \quad \text{et} \quad \lim_{\theta \to 0} A_{ij}(\theta) = 0 \tag{3.5}$$

On suppose que l'espace des configurations est entièrement connecté. Pour engendrer une **chaîne de Markov**, on applique, à partir d'une configuration initiale et à température constante, **n** fois le mécanisme de transition. Lorsque **n** tend vers l'infini, il existe, pour cette chaîne, une distribution stationnaire (ou valeur d'équilibre) unique **q(θ)** dont la i^{eme} composante **qi(θ)** vérifie la relation suivante:

$$q_i(\theta) = A_{iop,i}(\theta) \; / \; \sum_{i=1}^{|k|} A_{iop,i}(\theta) \tag{3.6}$$

iop:numéro d'une configuration optimale.
Les valeurs aux limites sont:

$$\lim_{\theta \to \infty} q_i(\theta) = 1/|S| \quad \text{et} \quad \lim_{\theta \to 0} q_i(\theta) = 1/S \quad \text{si } i \in S_{op} \tag{3.7}$$

$$\lim_{\theta \to 0} q(\theta) = 0 \quad \text{sinon}$$

Sop : ensemble des configurations optimales.

La convergence de l'algorithme dépend d'une façon directe de certains paramètres. Ces paramètres sont aux nombres de quatre, à savoir: la température initiale, la loi de décroissance de la température, la durée des paliers de température et le critère d'arrêt du programme. Tous ces paramètres seront devellopés en détail au chapitre V.

II.PRINCIPALES APPLICATIONS [5].

Le **recuit simulé** a été utilisé dans plusieurs domaines pratiques hors du domaine de l'électronique, en donnant des résultats assez bons. Parmi ces applications on peut citer:

II 1 Imagerie

Une opération très importante en traitement d'images, consiste à reconstituer l'image à partir de données bruitées ou incomplètes. Cette opération, effectuée à l'aide d'une méthode itérative, nécessite un temps de calcul très important, en raison du grand nombre de variables à traiter. Comme solution à ce problème, la méthode du **recuit simulé** a été proposée.

Un autre aspect de reconstitution d'image est utilisé en géophysique. Le problème consiste à établir l'épaisseur des différentes couches géologiques d'un site où on voudrait procéder à une prospection pétrolière: à des endroits bien précis du site on fait propager des ondes sismiques qu'on reçoit, après réflexion, sur plusieurs récepteurs qui enregistrent les ondes réfléchies, souvent bruitées. La fonction objectif, définie à partir des fonctions d'intercorrélation des enregistrements, a été optimisée par la méthode du **recuit simulé.**

II.2 Télécommunications

La méthode du **recuit simulé** a été appliquée comme méthode d'optimisation, pour un problème d'emplacement d'un certains nombre de terminaux et de concentrateurs. La fonction de coût est composée de la somme des longueurs des liaisons entre les terminaux et les concentrateurs, et de la somme des coûts d'implantation de ces concentrateurs. Les résultats obtenus par cette méthode étaient assez satisfaisants.

II.3 Architécture

Le problème est analogue à celui d'un circuit intégré. Ceci revient à chercher une bonne répartition des différentes pièces (cas d'un bâtiment) en minimisant la fonction de coût. Cette fonction doit tenir compte des différentes contraintes auxquelles la répartition voulue doit obéir.

CONCLUSION

Le **recuit simulé** permet d'optimiser des systèmes complexes et peut être appliqué dans differents domaines. Nous l'avons adapté au problème de placement des cellules dans un circuit intégré qui sera developpé aux chapitres qui suivent.

CHAPITRE IV

RECUIT SIMULE APPLIQUE AUX DIFFERENTS TYPE DE PLACEMENT

CHAPITRE IV

RECUIT SIMULE APPLIQUE AUX DIFFERENTS TYPES DE PLACEMENT

INTRODUCTION

Le problème de placement, comme il a été déjà défini, revient à placer un ensemble de modules sur une puce, de manière à minimiser une certaine fonction objectif établie par le concepteur. Le but essentiel est de réduire la surface totale de la puce occupée par les modules du circuit ainsi que la longueur totale des connexions.

Trois types de placement peuvent se présenter: Placement en Gate Array, standard cell et macro cellules.

Gate Array: les cellules sont toutes de même taille, contenant chacune un nombre fixe de transistors (fig IV.1). Chaque cellule peut assurer une certaine fonction logique par spécification des interconnexions entre les différents transistors. Le problème de placement est similaire au problème de placement classique sur **PCB** (printed circuit board).

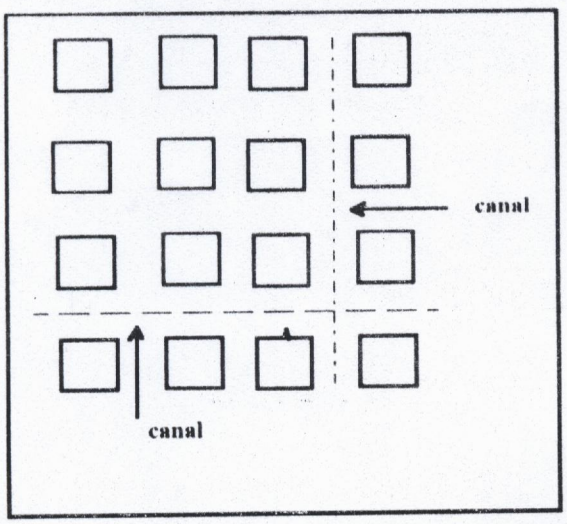


Fig IV.1 Placement en Gate array

Standard cell:, le concepteur choisit ses cellules à partir de bibliothèques, où il trouve toutes les informations concernant la cellule, à savoir: dimensions, performances, caractéristiques électriques et fonction logique qu'elle assure. En général, les cellules standard sont de même hauteur et de largeurs différentes, ceci permet de les rassembler en des rangées (fig IV.2). Les connexions entre cellules ne sont pas toujours entre rangées adjacentes, des cellules dites **"feedthrough" [14]** ayant une broche de chaque côté, servent d'intermédiaire.

Le placement en **Gate Array** et **standard cell** a pour avantage la réduction du temps de conception, mais ceci sera au dépens de la surface de la puce.

Macro cell: ce sont des cellules de tailles différentes (fig IV.3). Leur placement comparé à celui de **Gate Array** et **standard cell** paraît plus difficile, du fait que l'estimation de la surface de routage n'est pas facile à définir.

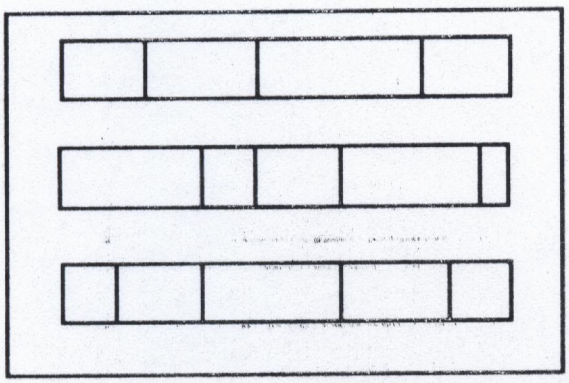


Fig IV.2 Placement des standard cell

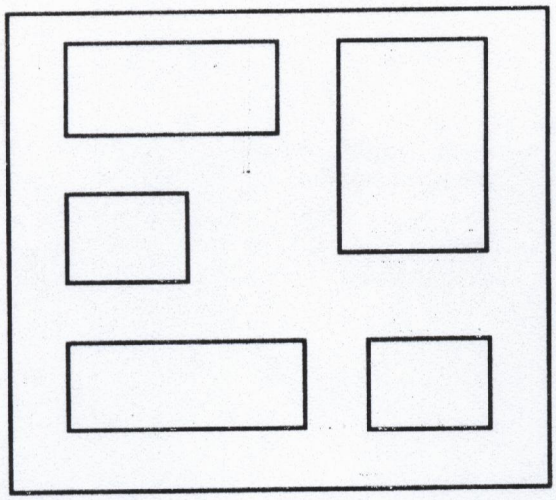


Fig IV.3 Placement des macro/cell

I.DIFFERENTS TYPES DE PLACEMENT

En réalité, pour chaque type de placement, l'algorithme de **recuit simulé** n'est qu'une amélioration d'un autre algorithme déjà établi. Dans cette première partie du chapitre, on essayera de définir ces algorithmes et les améliorations apportées par le recuit simulé appelé généralement: **"Timberwolf"**.

I.1 Placement pour Gate array[12][15].

Le premier algorithme présenté pour le placement Gate Array est selon **Kikpatrick, gelatt et vecchi (K.G.V)[17]**.

I.1.1 Placement selon K.G.V.

Dans cet algorithme, deux types de mouvements sont possibles: permutation entre deux modules et déplacement d'un module vers une place vacante.

La première fonction de coût proposée était basée sur la longueur totale des connexions. Hors, ce choix a conduit à un encombrement des connexions dans les canaux ce qui rendait le routage impossible. Comme solution à cet inconvénient, le concept de **"net crossing-histogram"** a permis une distribution plus ou moins uniforme dans les canaux.

Soit **Si** l'ensemble des places auxquelles les cellules d'un réseau **Ni** seront affectés, **et Ri** le plus petit rectangle entourant les centres de ces cellules appelé **"bounding rectangle"**. **L1,L2,...,Lk** représentent les differents canaux (verticaux ou horizontaux). Pour chaque canal **Li**, **wi** est le nombre total de réseaux dont les **"bounding rectangle"** se croisent avec le canal **Li**.(L'ensemble wi pour tous les canaux verticaux **Li** est appelé **"vertical net crossing-histogram"** et ceux des canaux horizontaux **"horizontal net crossing-histogram"**).

W est une estimation de la longueur, elle peut être calculée par la sommation des demi périmètres des differents **Ri**. Pour des raisons de faisabilité du routage, à chaque canal correspond une valeur seuil **ti**, exprimant ainsi la capacité de câblage qu'il peut supporter.

$$\delta_i = W_i - t_i \qquad \text{si } w_i < t_i \tag{4.1}$$

$$\Delta = \sum_{i=1}^{k} \delta_i^2 \tag{4.2}$$

L'expression de la fonction de coût est la suivante :

$$E = W + \alpha.\Delta \tag{4.3}$$

α est une constante,le terme Δ maintient l'uniformité de la distribution des connexions dans les canaux .

I.1.2 Placement selon Timberwolf.

Le placement selon **Timberwolf** est similaire à celui de **K.G.V**. L'utilisateur a le choix entre deux fonctions de coût: la première est identique à celle utilisée par **K.G.V** et la seconde n'est qu'un raffinement de la première. Dans ce cas l'encombrement est etudié par zone de canal.

On entend par point de coupure **"cut point"**, le point d'intersection du canal vertical et du canal horizontal. Chaque canal est divisé en un ensemble de segments de canaux noté **L1',L2',...,Lk'**(Fig IV.4). La valeur seuil **ti** dans ce cas, est affectée à chaque segment de canal **Li'**. Pour chaque réseau **Nj**, **Qj** est l'ensemble des segments des canaux qui se croisent ou appartiennent au **"bounding rectangle" Rj**.

L'encombrement par segment de canal dans **Qj** introduit par **Nj** est approximativement égal à **Pj/Lj**, où **Lj** est le nombre de segments de canal dans **Qj**, et **Pj** le demi périmètre de **Rj**.

$$W_{ij} = P_j / L_j \quad \text{si} \quad L_j \in Q_j \qquad \text{et } W_{ij} = 0 \quad \text{sinon} \tag{4.4}$$

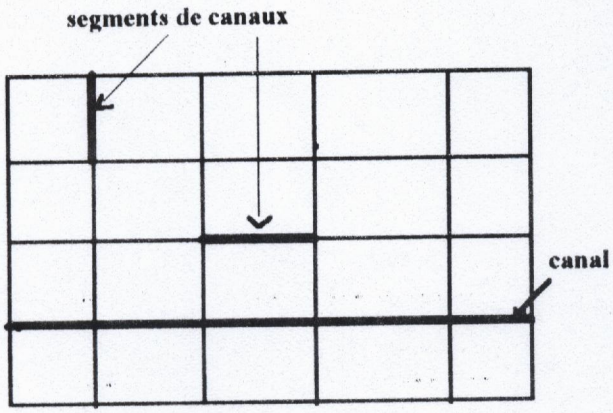


Fig IV.4 Représentation de segments de canaux

La fonction de coût peut être de la forme :

$$E = \sum_{i=1}^{l} W_i + \alpha \sum_{i=1}^{l} \delta_i^2 \tag{4.5}$$

Une autre définition des **"bounding rectangle"** a permis des placements avec plus de précision que celle définie ci dessus. Ces derniers entourent juste les côtés des modules dont les broches font partie du réseau considéré.

I.2 Placement pour standard cell.

Le placement des standard cell est présenté selon deux algorithmes: **Timberwolf** et **Grover**. La différence majeure entre les deux algorithmes réside dans la définition de l'espace solution.

Timberwolf permet un placement avec recouvrement entre modules comme solutions intermédiaires (fig IV.5). Par conséquent, la fonction de coût peut être calculée facilement après chaque mouvement. **Grover** ne le permet pas, ceci rend en général, le calcul de la fonction de coût très difficile. Un des avantages de cette approche est la réduction de l'espace solution.

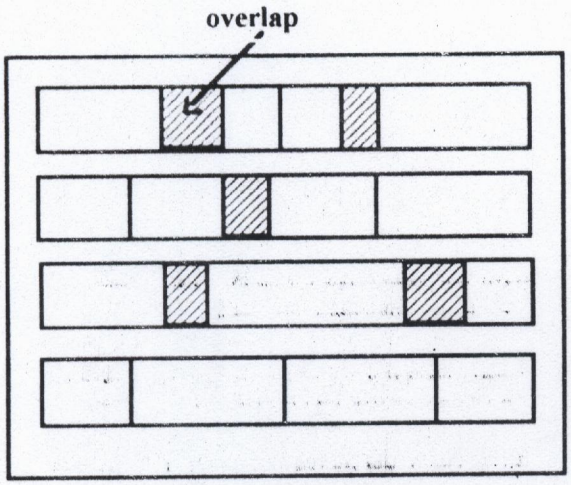


Fig IV.5 Overlaping des modules

I.2.1 Placement selon Timberwolf.

Le placement selon **Timberwolf** pour les **standard cell** s'effectue en deux phases. Lors de La première phase, au fur et à mesure que la température décroît, les modules se déplacent à l'intérieur et entre rangées. Le recouvrement est permis. Quand la température atteint une certaine valeur, la deuxième phase commence. Trois types de mouvements sont possibles:

M1: déplace le module vers une place vacante.

M2: permutation entre deux modules.

M3: changement d'orientation du module.

Les mouvements du type **M1** permettent aux modules un déplacement d'une rangée à une autre. Le recouvrement est en général, introduit par les mouvements de type **M2**. Le choix de **M1** est quatre fois plus probable que celui de **M2**.

Une fenêtre rectangulaire **R** dite **"range limiter"** définie à quelle paire de modules les mouvements de types **M2** s'appliquent, et le maximum de déplacement permis par les mouvements de type **M1**.

Au début du traitement, la fenêtre prend la taille de la puce, puis décroît proportionnellement au logarithme de la température. La permutation entre modules n'est autorisée que si leurs centres figurent à l'intérieur de la fenêtre.

La deuxième phase commence lorsque la température atteint le point où les mouvements de type **M1** ne seront plus permis. Les mouvements de type **M3** ne sont autorisés que si la solution obtenue par **M2** est rejetée. La Fonction de coût est définie comme suit:

$$E = C_1 + C_2 + C_3 \qquad (4.6)$$

C_1 , C_2 et C_3 sont respectivement, le coût total des connexions, le taux de recouvrement et la somme des déviations des rangées de leurs longueurs désirées (ne doivent pas dépasser la longueur de la puce) .

$$C_1 = \sum_{i=1}^{m} (\alpha_i W_i + \beta_i h_i) \qquad (4.7)$$

wi et **hi** sont respectivement la largeur et la hauteur de **Ri**, αi et **ßi** sont les poids horizontaux et verticaux du réseau **Ni** (si $\alpha i = \beta i = 1$ pour tous les réseaux, **C1** sera l'estimation typique de la longueur des connexions obtenue par la sommation des demi périmètre des **Ri**).

$$C_2 = \sum_{i \neq j} (O(i,j) + \alpha)^2 \qquad (4.8)$$

o(i,j) est le taux de recouvrement entre les modules i et j. α est un paramètre d'offset. Il assure la convergence de **C2** vers zéro lorsque la température tend vers zéro.

$$C_3 = \sum_{r} \beta | L(r) - d(r) | \qquad (4.9)$$

avec **d(r)** la longueur désirée de la rangée **r**, et **L(r)** la somme des largeurs de tous les modules appartenant à **r**. Notons, qu'en fin de traitement, les termes **C2** et **C3** doivent s'annuler et la fonction de coût se réduit à **C1**.

La température décroît selon la règle **Tk = r(k).Tk-1**. La valeur de **r(k)** croît de **0.8** jusqu'à une valeur maximale de **0.94** puis décroît à **0.1**. A chaque température, le nombre de mouvements effectués est donné par **K.n**, où **n** est le nombre total de modules et **K** une constante choisie par le concepteur.

Comme exemple de test d'un système large (**800 à 2700 modules**) **Timberwolf** comparé à **CIPAR** (développé par microsystèmes américains), a permi une réduction de **45% à 66%** dans la longueur des connexions sur **75** millions de mouvements en un temps de **6.5** heures CPU sur un **IBM 3081**.

I.2.2 Placement selon Grover

L'algorithme de **Grover** ne permet pas de recouvrement. Les modules sont soit placés au début ou à la fin de la rangée, ou encore insérés entre eux. Dans les deux cas un décalage des modules vers la droite ou vers la gauche peut se produire. La fonction de coût utilisée correspond au premier terme de celui de **Timberwolf** avec **αi = ßi =1.**

I.3 Placement pour macro cell

I.3.1 Algorithme de Jespen Gelatt .

Pour les macro cell, Trois types de mouvements sont possibles:

M1: translation horizontale ou verticale du module.
M2: rotation de 90° dans chaque direction.
M3: reflexion horizontale ou verticale .

Le recouvrement est introduit par les deux premiers types de mouvements. La fonction de coût est une généralisation de celle utilisée par **K.G.V** , plus un terme pour le recouvrement. Pour chaque ligne de la grille tracée à la surface de la puce, **L**, **nl** représente le nombre de réseaux qui la traversent et pour chaque module **A** croisant **L**, **IA** représente la longueur d'intersection (hauteur ou largeur de A). **XL** est la somme des longueurs d'intersections de tous les modules se croisant avec L. Le premier terme de la fonction de coût est:

$$W = \sum_{r} (nl + \beta . Xl)^2 \qquad (4.10)$$

ß étant une constante, **XL** permet de réduire l'encombrement des connexions dans les canaux. **ß** doit être choisie de manière à ce que la contribution de **XL** soit très faible par rapport à celle de **nL**. Le second terme de la fonction de coût est :

$$O = \sum_{i \neq j} o(i,j) \qquad (4.11)$$

L'expression finale de la fonction coût sera alors comme suit:

$$E = W + \alpha . O \qquad (4.12)$$

I.3.2 Timberwolf.

L'algorithme de placement des macro cell selon **Timberwolf**, est en principe, similaire à celui de **Jespen** et **Gelatt**. Seulement, **Timberwolf** permet une certaine flexibilité aux dimensions des modules, ce qui lui donne la possibilité de choisir un rapport de forme adéquat, permettant la réduction de la surface de la puce. La fonction de coût est identique à celle de **Jespen** et **Gelatt**.

D'après la description des différents types de placements et des expressions de leurs fonctions de coût, on peut en conclure que l'opération de placement n'est en réalité qu'une préparation à l'opération de routage. Un placement n'est jugé bon que si ce dernier permet un routage facile, de bonne qualité et qui se déroule dans les meilleurs conditions.

Le placement est représenté par sa fonction de coût. Cette fonction doit respecter certaines contraintes. Elle peut avoir une forme générale pour tous les types de placements. Cette forme est la suivante:

$$E_t = E_l + \alpha E_{rr} + \beta E_{rv} \qquad (4.13)$$

Et: est l'énergie totale .

Le premier terme de la fonction de coût **El**, représente la longueur totale des connexions. Pour des facilités de calcul, la distance entre deux boitiers est prise égale à la distance de **Manhattan** (distance en L).

Le second terme **Err[25]** correspond à l'énergie de recouvrement réel, qui présente une violation des zones interdites des composants. On entend par zones interdites, les zones où ne doivent pénétrer ni d'autre composants, ni des connexions afin d'éviter tous les risques de contact électriques et les phénomènes parasites. Généralement, ces zones correspondent à des régions qui se situent autour des plots des composants. Ce terme doit être obligatoirement nul en fin de programme.

Le dernier terme **Erv** est l'énergie de recouvrement virtuel. Un tel recouvrement n'est pas interdit du point de vue placement, mais il indique seulement que certains composants sont trop rapprochés les uns des autres. Une des contraintes qui pourra conduire à un bon placement, est d'essayer de séparer les différents boîtiers représentant les

composants d'une distance permettant le passage d'au moins un minimum de connexions.

α et **ß** sont les poids respectifs donnés à l'énergie de recouvrement réel et virtuel. Elles prennent généralement les valeurs suivantes:

$$\alpha=0.9Eli \qquad \beta=0.1Eli$$

Eli est l'énergie de la longueur initiale. (ces valeurs résultent d'un choix empirique).

Afin de réduire le nombre de **"cross-over"**, enjambement d'une connexion par une autre, l'idée d'introduire un quatrième terme au niveau de la fonction de coût s'avère nécessaire. Ce terme a pour but de minimiser le nombre de croisements entre les connexions.

Comme on le voit, la fonction de coût peut devenir assez complexe, et seule l'expérience permet d'obtenir une fonction suffisamment complexe, c'est à dire qui doit autant que possible saisir les qualités souhaitées avec un compromis entre l'effort de calcul et la qualité de la solution sans que cet effort de calcul soit pour autant inutilement accru.

Il est important de noter, que le passage d'un type de placement à un autre type est très facile, c'est à dire les trois types de placement peuvent se réduire à un seul type. Par exemple, il y'a des concepteurs qui préfèrent concevoir leurs circuits selon **Gate Array** sans se préoccuper du type de composants qu'ils placent. Ceci est faisable en affectant à la cellule de la matrice les dimensions du plus grand composant. Le concepteur peut gagner dans le temps de conception, mais ceci sera au dépens de la surface de la puce, surtout si la différence dans les dimensions des composants est énorme.

CHAPITRE V

PROGRAMME DE RECUIT

CHAPITRE V

PROGRAMME DE RECUIT

INTRODUCTION

Afin de développer un logiciel de placement des cellules dans un circuit intégré, nous avons utilisé l'algorithme du **recuit simulé**. Le programme de recuit considère le problème de placement comme un processus purement aléatoire où Chaque composant est représenté par un boîtier rectangulaire, repéré par les coordonnées de son centre de gravité.

Après avoir défini la fonction objectif qui répond aux contraintes imposées par le concepteur, Le programmeur doit définir les paramètres qui contrôlent la convergence du programme. Ces paramètres sont au nombre de quatre, à savoir :la température initiale, la loi de décroissance de la température, durée des paliers de température, et le critère d'arrêt du programme.

I.PARAMETRES DE CONVERGENCE DE L'ALGORITHME

Les quatre paramètres contrôlant la convergence de l'algorithme sont les suivants:

I.1.Température initiale [1][2][3].

D'après la définition de la probabilité d'acceptation établie par le test de **Metropolis**, représentée par la probabilité de **Boltzmann exp(-ΔE/T)**. Une grande proportion de perturbations est acceptée si la température est assez élevée. Ceci permet à toutes les configurations du système d'être obtenues avec la

même probabilité. Au fur et à mesure que la température décroît, le taux d'acceptation diminue.

En réalité, il est très difficile de déterminer l'ordre de grandeur d'une température élevée ou basse. Au niveau du programme, l'évaluation de la température initiale est déterminée d'une façon purement empirique. Une méthode très pratique permet de calculer la valeur de la température initiale avec une estimation plus ou moins bonne.

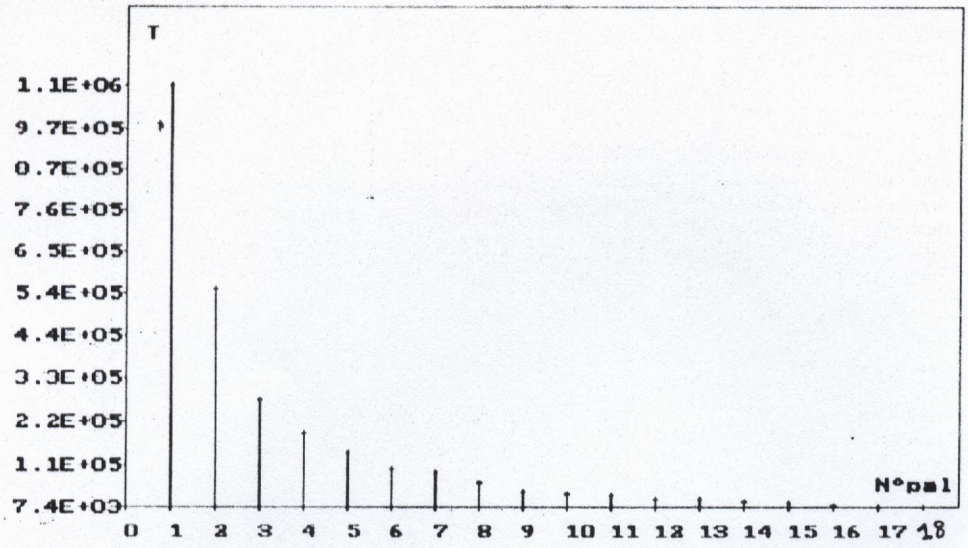
Afin d'acquérir une certaine connaissance du paysage d'énergie, on effectue un certain nombre de mouvements aléatoires et on calcule la moyenne des variations de la fonction sur ces mouvements **(Enermoy/nombre de mouvements)**. La température initiale est alors calculée comme:

$$T = \{ - \textbf{Enermoy} / \ln(\textbf{Acceptini}) \} \qquad (5.1)$$

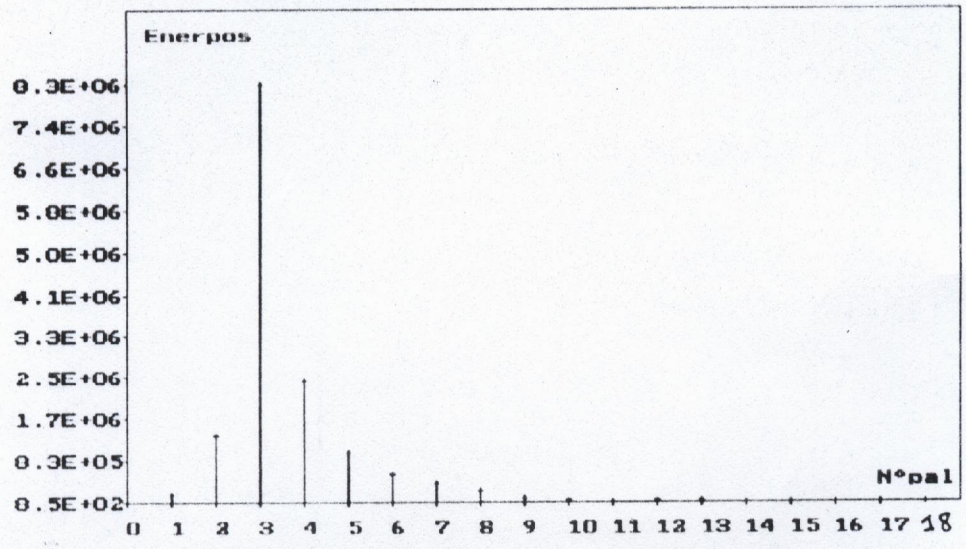
Acceptini :représente le taux d'acceptation initial.

Le programme comporte des histogrammes évaluant la variation de la température en fonction du nombre de paliers. Ces histogrammes permettent de définir le nombre d'essais adéquat pour la détermination de la valeur de la température initiale. C'est dans ce propos que plusieurs essais ont été faits. Pour un circuit contenant **20** cellules et la taille de l'echantillon prise égale à **5**, on montre sur la figure (Fig V.1) la variation de la température pour **2000** mouvements. Pour ce même nombre de mouvements, la figure (Fig V.2) montre le principe de l'algorithme de **Metropolis**, qui permet des augmentations dans la fonction energie. L'optimum est atteint au palier **18** (Fig V.3).

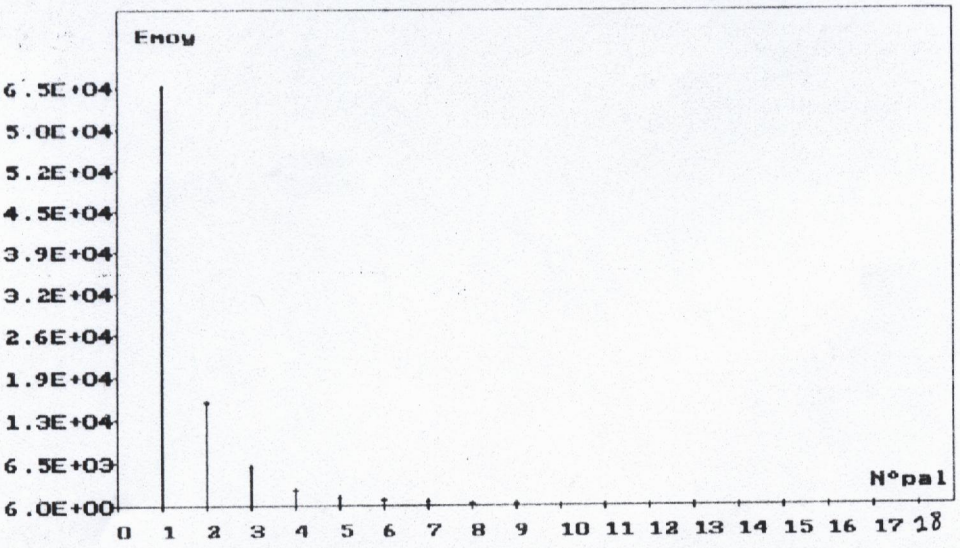
Cette même opération a été refaite avec réduction du nombre de mouvement à **5**, les résultats obtenus pour les variations de la température, l'energie positive et moyenne sont représentées sur les figures V.4, V.5 et V.6. L'optimum de la fonction est atteint au palier **33**. Pour une valeur de mouvements intermédiaire prise égale à **100**, cas des figures V.7, V.8 et V.9, l'optimum est atteint au palier **18**.



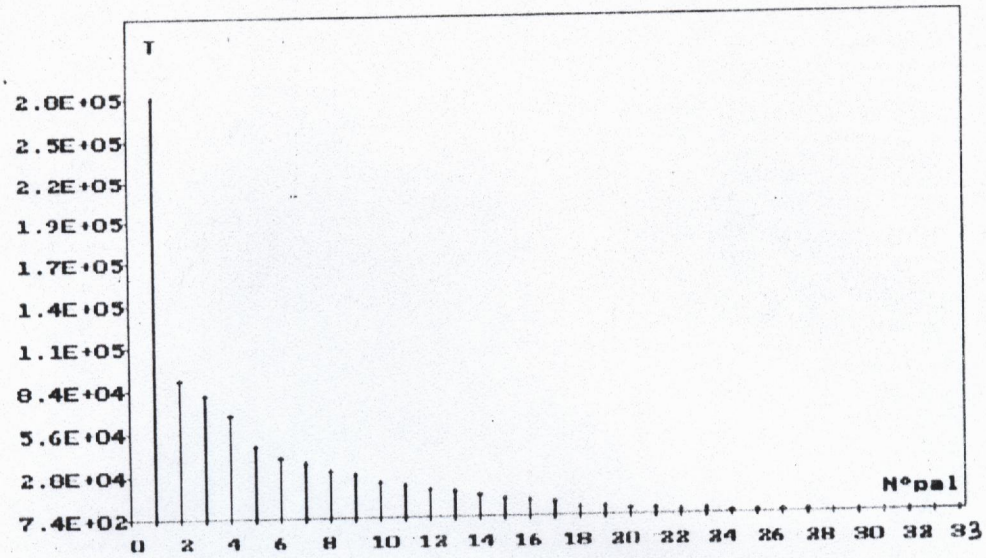
FigV.1 histogramme de la temperature en fonction du nombre de paliers



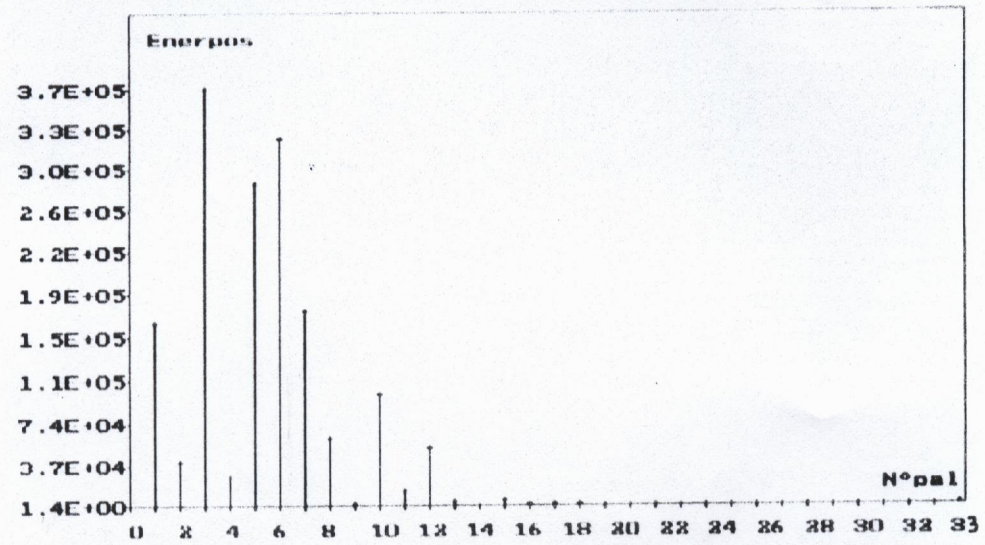
FigV.2 histogramme des variations positives d'energie



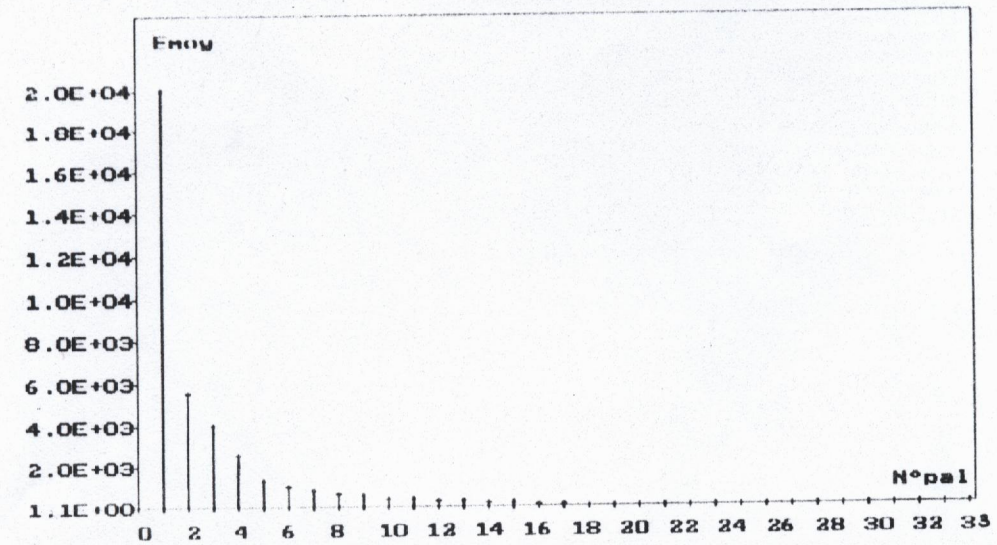
FigV.3 histogramme des états d'energie moyenne totale



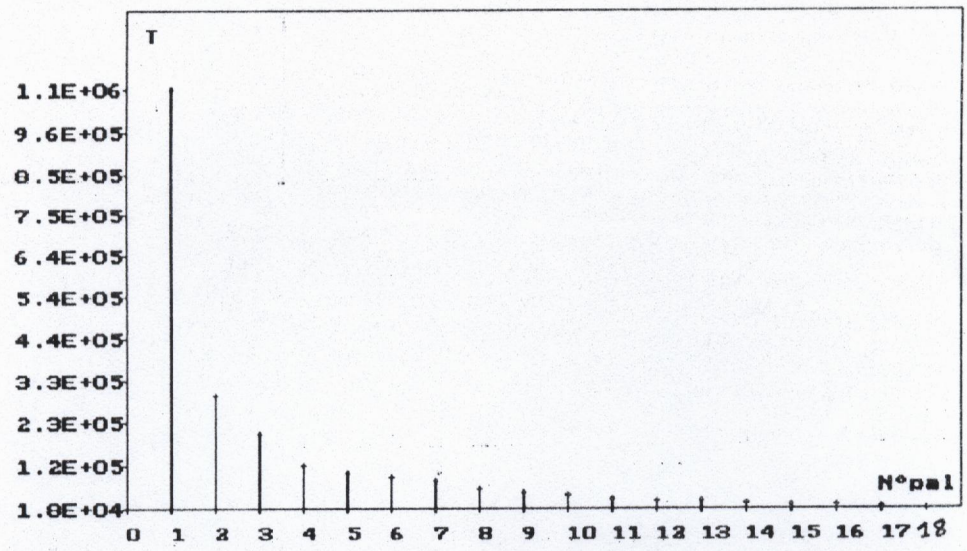
FigV.4 histogramme de la temperature en fonction du nombre de paliers



FigV.5 histogramme des variations positives d'energie



FigV.6 histogramme des états d'energie moyenne totale



FigV.7 histogramme de la temperature en fonction du nombre de paliers

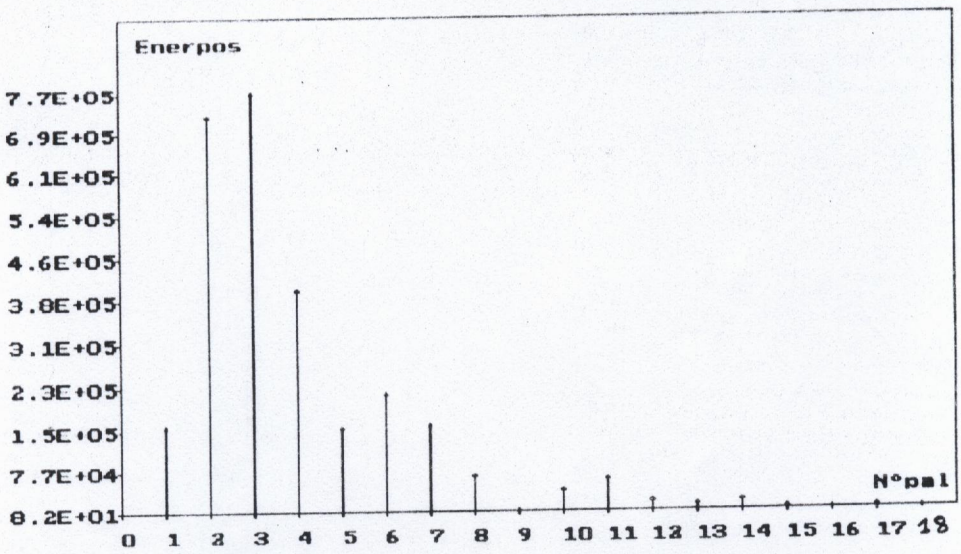
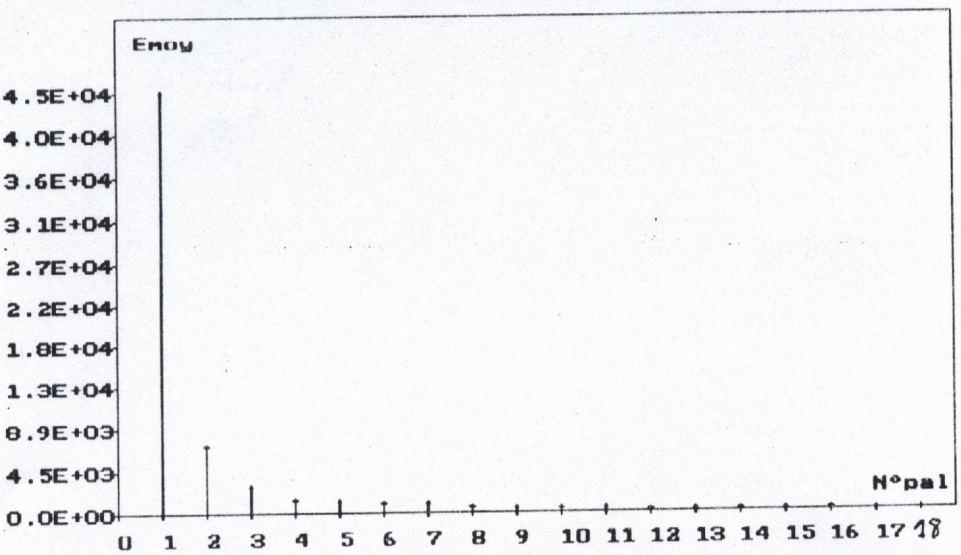


Fig V.8 histogramme des variations positives d'energie



FigV.9 histogramme des états d'energie moyenne totale

De la comparaison des résultats obtenus à partir de ces trois essais, découle que plus le nombre de mouvements est élevé, plus il est facile d'avoir une idée plus précise sur l'espace solution. Ceci se traduit par le bon choix de la température initiale. Cette valeur doit être assez élevée, ce qui permet d'atteindre l'optimum en un nombre moyen de paliers.(cas du troisième essai: nombre de mouvements =**100**, nombre de paliers **18**). La valeur de la température initiale est stockée dans une variable définie au niveau du programme par **Temp0**.

I.2.Loi de décroissance de la température [3][19].

La structure de l'algorithme est telle qu'à chaque changement de palier, le pas du mouvement et la température changent. En effet, lorsque la température décroît, le taux d'acceptation des variations positives de la fonction est plus faible selon le critère de **recuit**. Or on considère que l'efficacité de l'algorithme est meilleure lorsque le taux est sensiblement constant.

Par ailleur, l'amplitude des mouvements influe également sur ce taux, puisque pour les mouvements d'amplitude plus faible, les variations de la fonction sont plus importantes. Selon la formule du taux d'acceptation pour les variations positives: **$p=\exp(-\Delta E/T)$**, ce taux doit augmenter lorsque le pas diminue.

Plusieurs lois de décroissance de la température ont été proposées **[3][5]**. Dans le cadre de ce travail, nous avons choisi celle proposée par **P.Siarry** qui nous a semblé la plus appropriée: l'introduction d'un facteur de décroissance de température permet de rendre l'opération de **recuit** efficace sans que cette derniére n'exige un espace memoire énorme comme celle du **thermostat** illustée au chapitre III.

La température est multipliée par un coefficient dont l'expression est:

$$\mathbf{Coeff} = \mathbf{E_{min}} / \mathbf{E_{moy}} \qquad (5.2)$$

Où **Emin** est le minimum obtenu sur le palier en question et **Emoy**, la valeur moyenne de la fonction sur ce palier.

Par ailleur, l'existence d'une liaison directe entre la décroissance de la

température et du pas, nous a conduit a introduire un autre facteur de decroissance du pas donné par la relation suivante:

$$\mathbf{Coeff = -T * \ln(p_0) / E_{moy}} \qquad (5.3)$$

Pour chacun de ces deux coefficients, on fixe des domaines de validité de ces lois. On vérifie donc que les coefficients ne doivent pas dépasser certaines limites. En outre, par constitution, la loi du pas ne peut être mise en oeuvre que lorsque la valeur de ce dernier est suffisamment petite, c'est pourquoi on ne l'utilise qu'à partir d'un palier donné.

Les limitations du coefficient de température dans un certain domaine permettent d'éviter les baisses trop brusques de la température au début et les stagnations trop longue à la fin.

I.3.Durée des paliers [3][4][5] .

La longueur d'un palier de température est fixée, empiriquement, par le nombre de configurations acceptées. Ce nombre ne doit pas dépasser **12 fois** le nombre total de boîtiers ou composants utilisés.Il ne doit pas également dépasser le nombre total de perturbations essayées, qui doit être inférieur ou égal à **100 fois** le nombre de boîtiers.

Au niveau du programme, les deux conditions de changement de palier sont définies par les deux variables **Arret1** et **Arret2** telles que:

$$\mathbf{Arret1 = 12 * Nmax} \qquad (5.4)$$

$$\mathbf{Arret2 = 100 * Nmax} \qquad (5.5)$$

Nmax représente le nombre maximal de boîtiers .

I.4.Critère d'arrêt du programme .

On décide de l'arrêt du programme, lorsqu'on juge qu'il n'y a plus d'amélioration notable dans la fonction énergie. Pour cela, trois conditions liées par des **"ou"** doivent être vérifiées:

- Le nombre de mouvements acceptés sur le dernier palier est inférieur à un certain nombre de mouvements minimum effectués par palier.

- Le nombre de paliers a atteint un certain nombre maximum autorisé par le programme. (Ces deux nombres sont fixés par l'opérateur).

- Le pas est trop petit, c'est à dire le mouvement devient insignifiant.

II.DEROULEMENT DU PROGRAMME

Le programme de **recuit** démarre d'un état où d'une certaine configuration initiale. Cette configuration peut être très désordonnée ou proche de l'état optimum. Ceci explique l'importance du bon choix de la valeur de la température initiale. L'utilisateur a le choix entre deux états de départ :

- Un état complètement arbitraire, engendré à l'aide d'un générateur de configurations aléatoires.

- Une configuration définie par l'opérateur: dans ce cas, il doit fournir l'emplacement de tous les boîtiers sur le circuit.

Une configuration est définie par l'ensemble des positions des différents boîtiers où chacun est représenté par ses coordonnées. L'opérateur doit fournir au programme son domaine de recherche, c'est à dire, fixer les bornes supérieures et les bornes inférieures qui limitent le déplacement des différents boîtiers, ainsi que le pas effectué à chaque mouvement.

Au niveau du programme, on se contente de mouvements de translation. La valeur initiale que peut prendre le pas est définie par:

$$\mathbf{Bornesup[I] - Borneinf[i] / 2} \qquad (5.6)$$

Bornesup[I] et **borneinf[I]** sont respectivement les bornes supérieure et inférieure pour chaque variable.

Ce choix concernant le pas, permet d'assurer que la totalité de l'intervalle peut être balayée par chaque variable au cours du premier palier et que l'on ne laisse pas de **"zone d'ombre"**.

On considère aussi, que lorsque l'on n'a aucune idée de la position de la variable correspondante, le plus sage est de partir du milieu de l'intervalle défini par:

$$\mathbf{Bornesup[i] + Borneinf[i] / 2} \qquad (5.7)$$

II.1.Partitionnement [3][4].

Une fois le domaine de recherche et le pas fixés, le programme commence par effectuer des mouvements sur les variables. D'après **Poivey[3]**, on augmente l'efficacité de l'algorithme en effectuant les mouvements sur des groupes de variables et non pas sur l'ensemble. Cette méthode est dite " **partitionnement".** On appellera échantillon les variables choisies.

La taille de l'échantillon est fixe. Les variables qui le constituent sont tirées de façon aléatoire. Toutefois, une variable ne peut être tirée à nouveau que lorsque toutes les autres variables ont été tirées un même nombre de fois.

Par exemple, une variable tirée pour la troisième fois, ne peut être tirée que lorsque toutes les autres variables ont été tirées au moins trois fois. Cette contrainte permet à la méthode d'explorer l'espace des variables sans privilégier aucune direction.

Lorsque le nombre total des variables est un multiple de la taille de l'echantillon, le problème est relativement simple. Il suffit d'effectuer des tirages sans remise jusqu'à épuisement des variables non tirées. Le même processus peut être répété indéfiniment.

Dans le cas contraire, la situation est plus complexe. Il arrive un moment où le nombre de variables restantes ne permet pas de constituer un echantillon complet. On doit donc adjoindre à ce premier groupe de variables c'est à dire l'échantillon incomplet, un second groupe permettant de compléter l'echantillon en excluant les variables du premier groupe pour qu'il n'y ait pas de doublés.

Il est donc necéssaire de pouvoir identifier à tout instant les variables déjà tirées de celles non tirées. Cette distinction se fait par la simulation de pointeur, représentée au niveau du programme par le tableau **Index[I]**. Ce tableau contient les numéros assignés à chaque variable. De même le programme comporte une fonction tirage, destinée à extraire les variables de façon aléatoire.

Pour valider la technique proposée par **Poivey**, plusieurs essais ont été faits en jouant sur le nombre de cellules et la taille de l'échantillon. Ces résultats montrent que la fonction energie n'atteint son optimum que si la taille des cellules est un multiple de celle de l'echantillon. Cette taille doit être moyenne par rapport à celle du nombre de cellules, par exemple pour **20** cellules, la taille de l'echantillon prise est égale à **10**, **5** et **4**, tous des multiples de **20**. Le meilleur résultat est obtenu pour celui de **4** en un temps raisonnable.

Pour une taille de l'echantillon égale à **6** ou **3**, l'energie n'a pas diminué, malgré le nombre élevé des mouvements effectués. Ceci confirme la technique de **Poivey**. Ces résultats mettent également en évidence l'augmentation du temps d'exécution en fonction du nombre de cellules. les differents résultats sont représentés sur le tableau ci dessous.

N	ech	Tinit	Tfinal	Einit	Efinal	mvts	t(mn)
5	1	0.179E08	0.701E-04	0.470E+8	0.645E+3	70938	15
5	2	0.218E08	0.836E+8	0.142E-02	0.142E-2	67059	7
5	5	0.178E08	0.470E+8	0.645E+3	0.645E+3	42680	4
10	1	0.941E08	0.128E+8	0.658E-04	0.658E-4	135457	10
10	2	0.173E08	0.121E-02	0.941E+8	0.289E+3	103826	10
10	5	0.851E08	0.941E+8	0.289E+3	0.289E+3	88401	11
10	10	0.156E09	0.941E+8	0.941E+8	0.289E+3	75219	12
20	1	0.836E08	0.238E-03	0.940E+7	0.497E-4	146079	15
20	3	0.836E08	0.250E+0	0.836E+8	0.217E+8	187855	22
20	4	0.157E08	0.116E-02	0.836E+8	0.934E-2	77883	10
20	5	0.157E08	0.174E-02	0.836E+8	0.754E-2	76608	12
20	6	0.356E08	0.106E+1	0.836E+8	0.704E+8	246554	41
20	10	0.139E09	0.456E-02	0.836E+8	0.295E-1	70294	14
40	5	0.149E08	0.137E-03	0.836E+8	0.778E-3	165272	40

N: nombre de variables;
ech: taille de l'echantillon;
Tinit: température initiale;
Tfinal: température finale;
Einit: valeur de l'energie initiale;
Efinal: valeur de l'energie finale;
mvts: nombre de mouvements effectués;
t(mn): temps d'exécution en minutes.

II.2.Mouvement.

Lorsque l'on effectue un mouvement, on n'est pas sûr que celui ci sera accepté. Il faudra donc prévoir une méthode qui permette au programme de se ramener à sa configuration initiale juste avant le mouvement. C'est pourquoi l'opérateur est amené à faire avant chaque mouvement, une copie des positions des différentes variables. Une fois la copie faite, on procède au mouvement sur les variables echantillon.

Le mouvement se fait en ajoutant à certaines variables le pas affecté d'un coefficient compris entre **-1** et **1** que l'on appellera **"Amplitude".**

L'acceptation ou non du mouvement que l'on vient d'effectuer est déterminée par le test de **Metropolis**. Si la variation de la fonction est négative, le mouvement est automatiquement accepté. Dans le cas où celle ci est positive, le mouvement est accepté avec une probabilité $P=\exp\{-\Delta E/T\}$.

Si la quantité $(-\Delta E/T)$ est supérieure à **50**, on considère directement que la probabilité est nulle. Ceci permet d'éviter des débordements dans le calcul de l'exponentielle.

Lorsque le test conclut qu'il y'a acceptation du mouvement, il faut que les nouvelles valeurs des variables soient effectivement stockées dans les tableaux destinés pour cette fonction.

Le programme developpé arrive à minimiser la longueur des connexions, et de corriger le problème d'**overlaping** (recouvrement) quand celui ci se présente. Cette correction est assurée par un simple test effectué une fois les positions des différentes cellules fixées.

Le programme compare les centres de gravités d'une cellule à celle qui se situe juste après. Si cette différence est inferieure ou égale à la largeur de la cellule, ceci indique la présence d'un recouvrement. Pour l'éliminer, le programme déplace la deuxième cellule d'une quantité **D** qui correspond à la surface de recouvrement, qui n'est autre que la différence entre la largeur de la cellule et celle des centres de gravité des deux cellules juxtaposées.

Cette technique risque de donner un placement avec des cellules placées l'une contre l'autre. Comme solution à ce second problème, le programme doit déplacer la cellule d'une quantité **D+d**, où **d** la largeur minimale d'un canal. Celle ci est proportionnelle au nombre minimal de pistes qu'un canal peut supporter. Ce nombre est imposé par le concepteur.

La figure (Fig V.10) montre un exemple de placement d'un circuit à **160** cellules, **réseaux identiques**. Le placement présente un recouvrement entre les cellules qui a été corrigé comme le montre la figure (Fig V.11).

Un autre exemple de placement pour un circuit comportant **80** cellules est illustré sur la figure (Fig V.12). Ce placement ne présente pas de recouvrement.

Le programme présente un aspect visuel, il permet à l'utilisateur de suivre le déroulement du programme sur écran. De même le programme ne nécessite aucun fichier extérieur de données. Il fournit un fichier de résultats principal sous forme de fichier de réels, et un fichier de chaînes de caractères pour la description de l'algorithme, rassemblés dans un seul fichier texte.

Les résultats d'un essai de placement pour un circuit contenant **20** cellules par réseaux et une taille d'échantillon prise égale à **5** sont récapitulés sur le tableau ci dessous (TAB V.1).

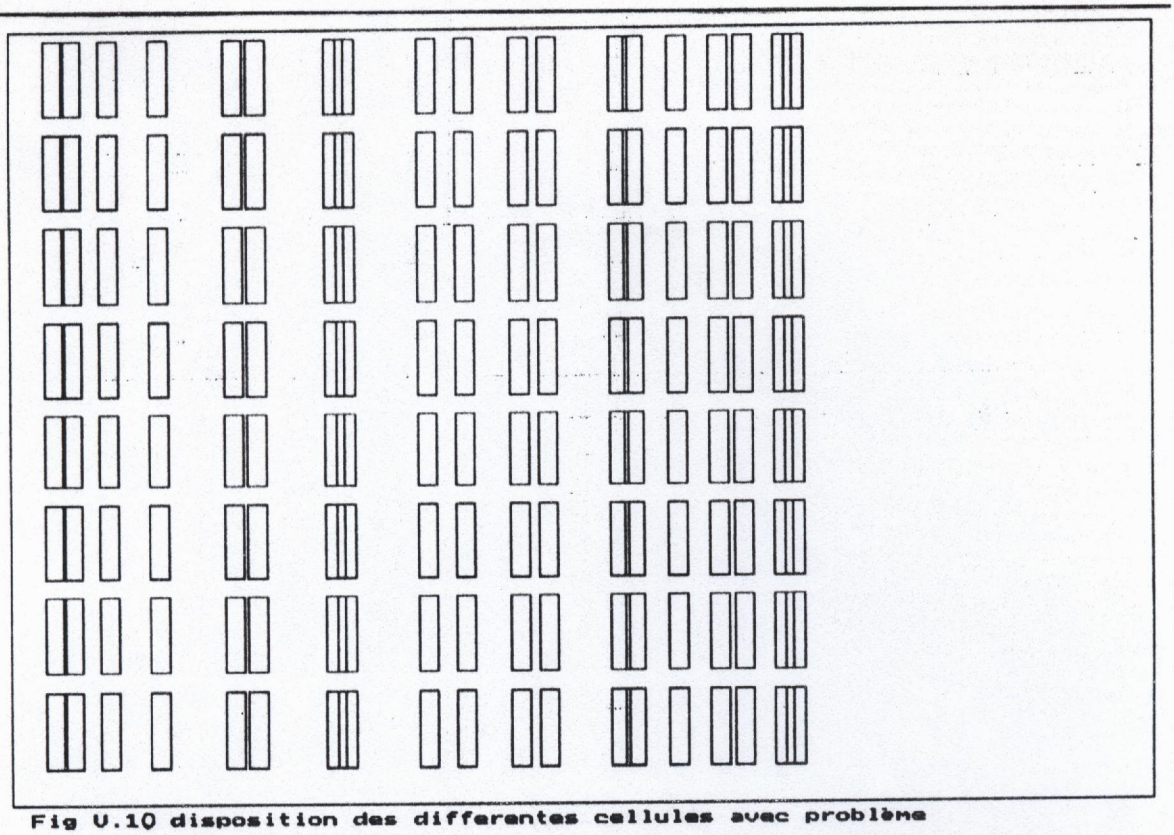


Fig V.10 disposition des differentes cellules avec problème d'overlapping cas d'un circuit à160 cellules reseaux identiques

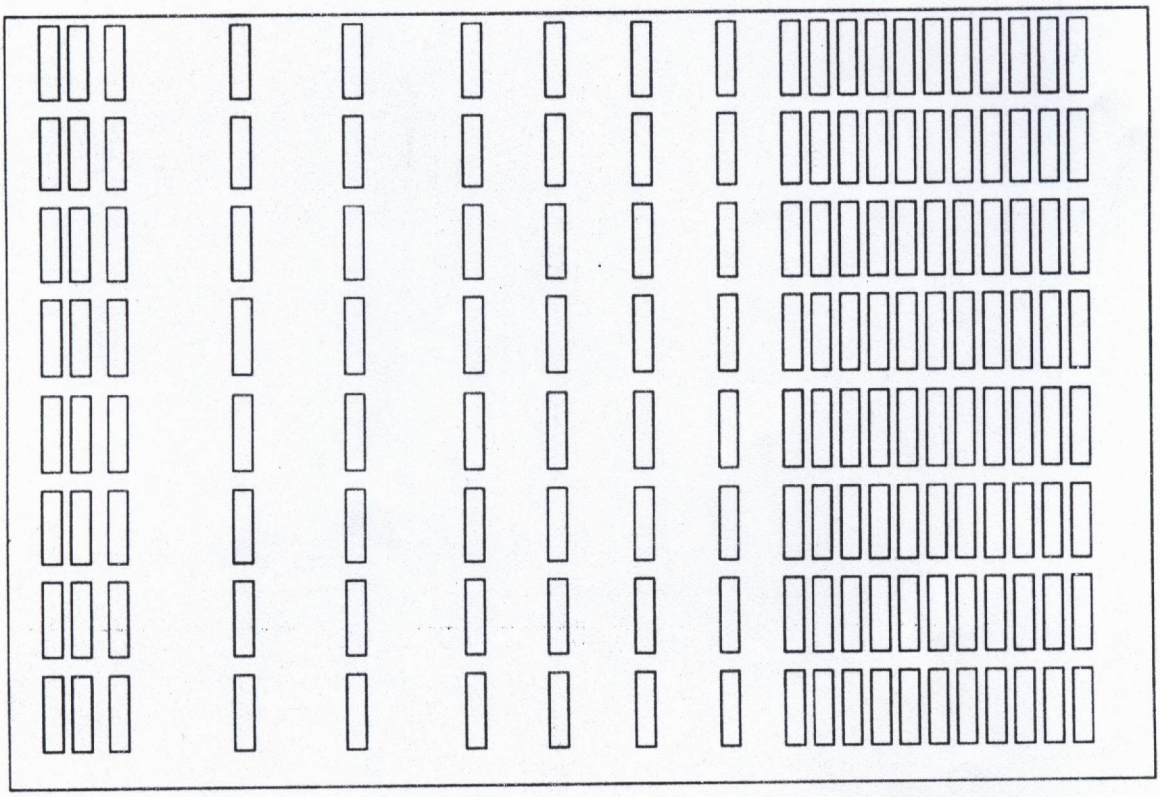


Fig V.11 disposition des differentes cellules après correction du problème d'overlaping

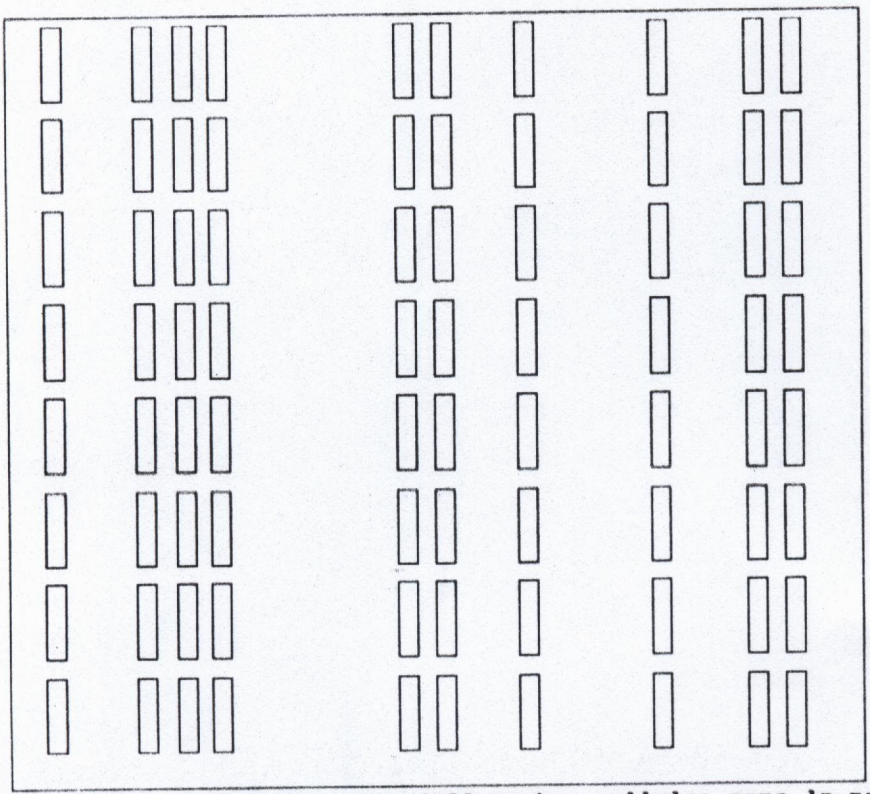


Fig V.12 disposition des differentes cellules sans le problème d'overlapping cas d'un circuit à 80 cellules réseaux identiques

P	Temp	Coeft	Pas	Coefp	<E>	<dE(>0)	MB	MHA	MHR	Pm	EMin.
1	2.1E+06	49.0	2.0E+01	80.0	7.8E+04	0.0E+00	20	0	1980	1	3.82E+04
2	1.0E+06	65.7	1.6E+01	80.0	2.6E+04	0.0E+00	24	0	1976	2	1.71E+04
3	6.8E+05	55.1	1.3E+01	80.0	1.2E+04	0.0E+00	15	0	1985	3	6.49E+03
4	3.8E+05	61.6	1.0E+01	80.0	4.1E+03	0.0E+00	25	0	1975	4	2.54E+03
5	2.3E+05	82.7	8.2E+00	80.0	2.0E+03	0.0E+00	12	0	1988	5	1.69E+03
6	1.9E+05	85.4	6.6E+00	80.0	1.5E+03	0.0E+00	12	0	1988	6	1.28E+03
7	1.6E+05	85.4	5.2E+00	80.0	1.0E+03	0.0E+00	9	0	1991	7	8.80E+02
8	1.4E+05	90.0	4.2E+00	80.0	8.2E+02	0.0E+00	10	0	1990	8	7.76E+02
9	1.3E+05	89.2	3.4E+00	80.0	6.7E+02	0.0E+00	20	0	1980	9	5.98E+02
10	1.1E+05	81.7	2.7E+00	80.0	5.2E+02	0.0E+00	17	0	1983	10	4.26E+02
11	9.2E+04	90.0	2.1E+00	80.0	3.6E+02	0.0E+00	11	0	1989	11	3.40E+02
12	8.2E+04	88.3	1.7E+00	80.0	3.1E+02	0.0E+00	18	0	1982	12	2.73E+02
13	7.3E+04	69.4	1.4E+00	80.0	2.2E+02	0.0E+00	24	0	1976	13	1.50E+02
14	5.1E+04	90.0	1.1E+00	80.0	1.4E+02	0.0E+00	9	0	1991	14	1.37E+02
15	4.5E+04	90.0	8.8E-01	80.0	1.2E+02	0.0E+00	16	0	1984	15	1.13E+02
16	4.1E+04	90.0	7.0E-01	80.0	1.0E+02	0.0E+00	17	0	1983	16	9.47E+01
17	3.7E+04	89.7	5.6E-01	80.0	8.4E+01	0.0E+00	22	0	1978	17	7.55E+01
18	3.3E+04	87.6	4.5E-01	80.0	6.8E+01	0.0E+00	28	0	1972	8	5.96E+01
19	2.9E+04	84.3	3.6E-01	80.0	5.1E+01	0.0E+00	33	0	1967	19	4.28E+01
20	2.4E+04	80.1	2.9E-01	80.0	3.5E+01	0.0E+00	38	0	1962	20	2.81E+01
21	2.0E+04	84.0	2.3E-01	80.0	2.4E+01	0.0E+00	35	0	1965	21	2.06E+01
22	1.6E+04	79.4	1.8E-01	80.0	1.7E+01	0.0E+00	30	0	1970	22	1.39E+01
23	1.3E+04	74.8	1.5E-01	80.0	1.1E+01	0.0E+00	39	0	1961	23	7.95E+00
24	9.8E+03	80.7	1.2E-01	80.0	5.8E+00	0.0E+00	35	0	1965	24	4.69E+00
25	7.9E+03	88.0	9.4E-02	120.0	3.5E+00	0.0E+00	22	0	1978	25	3.05E+00
26	6.9E+03	90.0	1.1E-01	120.0	2.8E+00	0.0E+00	11	0	1989	26	2.74E+00
27	6.2E+03	90.0	1.4E-01	120.0	2.7E+00	0.0E+00	4	0	1996	27	2.66E+00

TAB V.1 Principaux résultats obtenus par le programme.

P: Numéro du palier.
Temp: Température correspondant au palier P.
Coeft: Coefficient de décroissance de la température.
Pas: Valeur du pas.
Coefp: Coefficient du pas.
<E>: Valeur moyenne de l'energie.
<dE(>0)>: Moyenne des variations positives de la fonction energie.
MB: Nombre de mouvements acceptés avec baisse de la fonction energie.
MHA: Nombre de mouvements acceptés avec élévation de la fonction énergie.
MHR: Nombre de mouvements refusés.
Pm: Numéro du palier où l'optimum est atteint.
Emin: Valeur minimale de l'énergie atteinte.

A cause du processus de génération aléatoire engendré par la fonction **RANDOM** de turbo pascal, le programme refait l'opération de placement un nombre fini de fois, fixé par l'utilisateur.

Les résultats ne sont délivrés de manière plus ou moins detaillée que pour le premier essai. Pour le reste des essais, le programme se contente de donner la valeur de certains paramètres seulement, à savoir: le numéro du palier où l'optimum est atteint, la valeur de cet optimum, nombre de mouvements effectués et la valeur de la fonction énergie au **10000** mouvement.

Dans notre cas, le nombre d'essais est fixé à **5**. Les résultats obtenus sont représentés ci dessous. On remarque que les résultats pour les différents essais sont du même ordre de grandeur.

Essai 2

semence	paliers	optimum	essais	E à 10000
0	1/2	6.8E+03	4000	1.6E+05

Essai 3

semence	paliers	optimum	essais	E à 10000
0	1/1	6.9E+03	2000	1.6E+05

Essai 4

semence	paliers	optimum	essais	E à 10000
0	3/3	2.2E+03	6000	1.6E+05

Essai 5

semence	paliers	optimum	essais	E à 10000
0	4/4	8.7E+02	8000	1.6E+05

De même, le programme délivre un fichier complet relatif à l'opération de placement. Indiquant à l'utilisateur le détail sur les différentes techniques exploitées par celui ci. Un prototype de ce fichier est représenté ci dessous.

les résultats obtenus :

**********623
Fonction objectif: s1,(X(i+1)^2-X(i)^2)^2
****

nombre de variables : 20
taille des echantillons: 5
*** initiale optimale ***
fonction 1.60000E+05 0.00000E+00

données sur tableau :

variable	1	2	3	4	5
minimum	0.0	0.0	0.0	0.0	0.0
initial	20.0	20.0	20.0	20.0	20.0
maximum	40.0	40.0	40.0	40.0	40.0
optimum	1	1	1	1	1
variable	6	7	8	9	10
minimum	0.0	0.0	0.0	0.0	0.0
initial	20.0	20.0	20.0	20.0	20.0
maximum	40.0	40.0	40.0	40.0	40.0
optimum	1	1	1	1	1
variable	11	12	13	14	15
minimum	0.0	0.0	0.0	0.0	0.0
initial	20.0	20.0	20.0	20.0	20.0
maximum	40.0	40.0	40.0	40.0	40.0
optimum	1	1	1	1	1
variable	16	17	18	19	20
minimum	0.0	0.0	0.0	0.0	0.0
initial	20.0	20.0	20.0	20.0	20.0
maximum	40.0	40.0	40.0	40.0	40.0
optimum	1	1	1	1	1

recuit simulé

semence aléatoire : 0

programme de recuit

* taux initial d'acceptation : 0.10
* température initiale : 5.3E+05
*decroissance de la temperature : trempe
*changement de palier: 12Nmax mvts acceptes ou 100Nmax
*critère d'arret :50 paliers

programme de discrétisation

* pas initial : 20.0
*décroissance du pas : coeff:Max(0.8,Min(1.2,-T*Ln(0.5)/dE(>0)))
*amplitude des mvts : aléatoire [0,1]

RESULTATS

*nombre de mvts tentés : 12000
*nombre de mvts accéptés dE<0 : 72
*nombre de mvts accéptés dE>0 : 0
*nombre de mvts refusés : 11928

****

***** initial * final *****
energie 1.600E+05 * 1.990E+02--> au palier 6/ 6
temp. 5.302E+05 * 4.527E+04
pas 2.000E+01 * 5.243E+00

****

Les differentes positions

variable	1	2	3	4	5
Résultat	20.25126	20.12898	20.08340	20.10313	20.19273
fenetre	50.35	50.32	50.21	50.26	50.48

variable	6	7	8	9	10
Résultat	20.25126	20.08497	19.88453	19.65283	19.66472
fenetre	50.63	50.21	49.71	49.13	49.16

variable	11	12	13	14	15
Résultat	19.62929	19.58183	19.67114	19.76597	19.89069
fenetre	49.07	48.95	49.18	49.41	49.73

variable	16	17	18	19	20
Résultat	20.14544	20.19511	20.15200	20.08585	20.03416
fenetre	50.36	50.49	50.38	50.21	50.09

L'organigramme principal des différentes procédures suivies par le programme est illustré sur la figure (fig V.13).

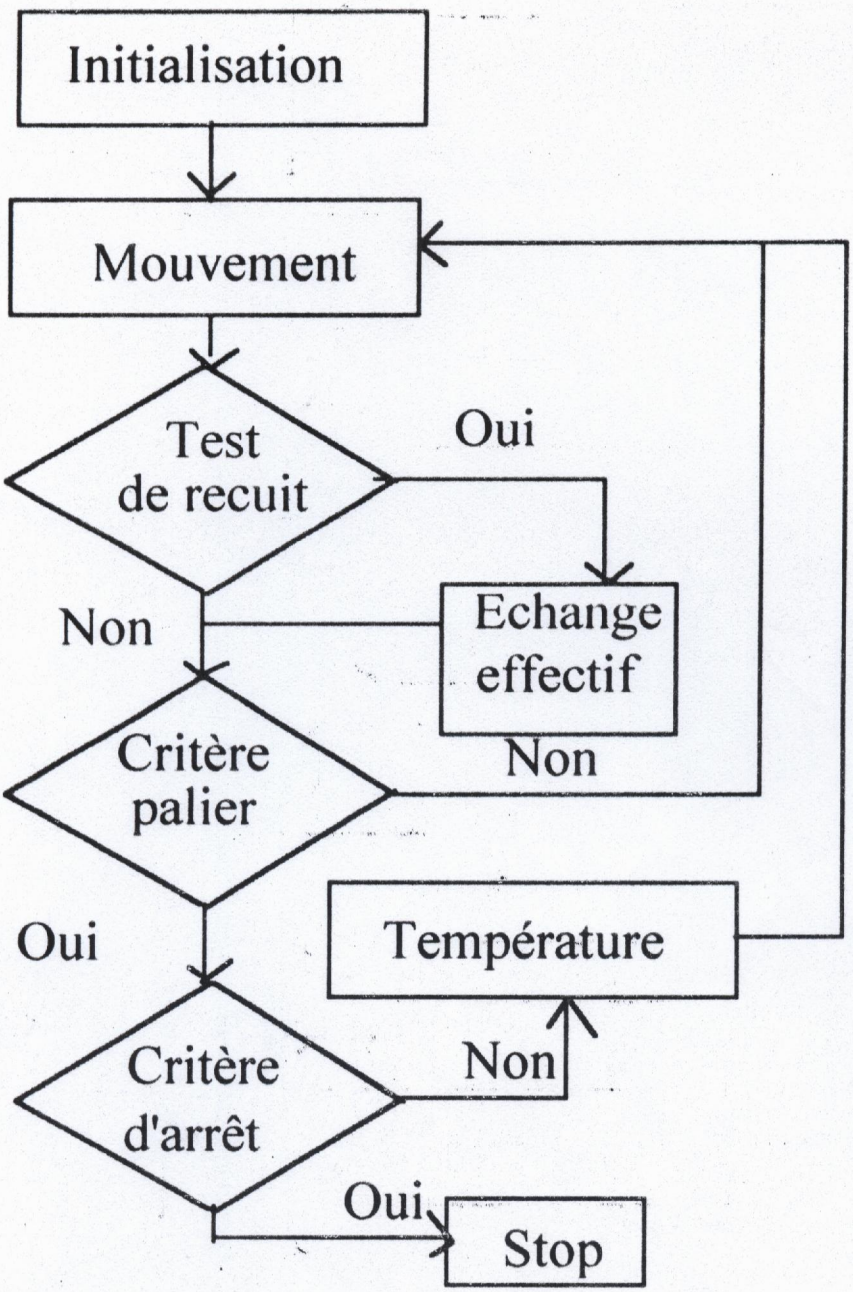


Fig V.13 Organigramme principal

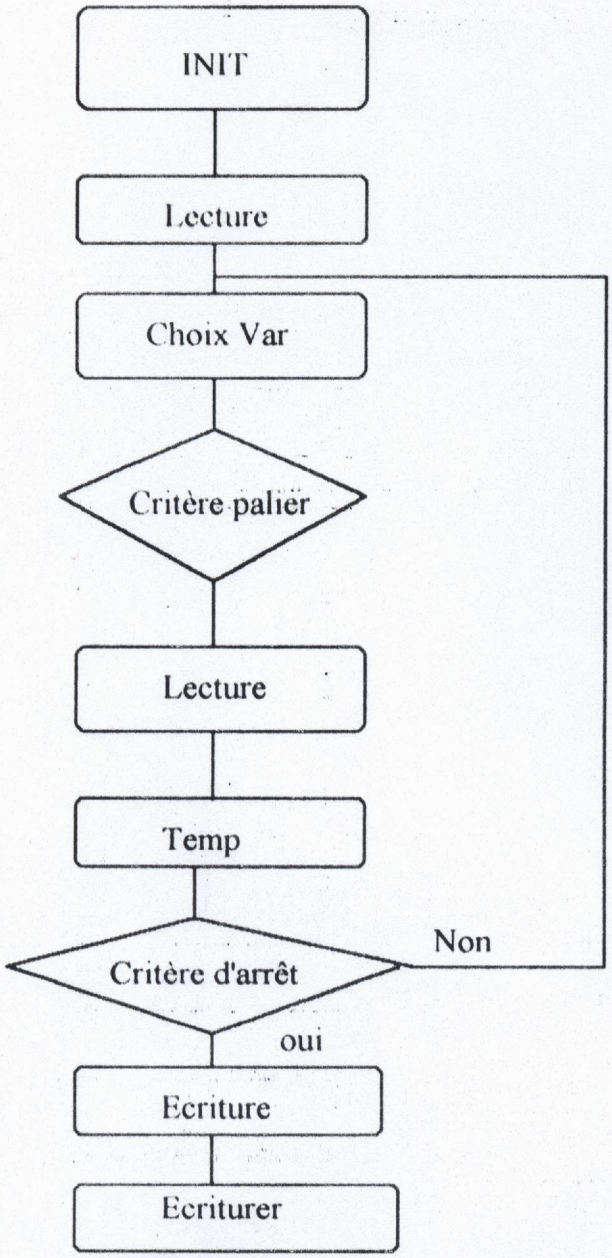
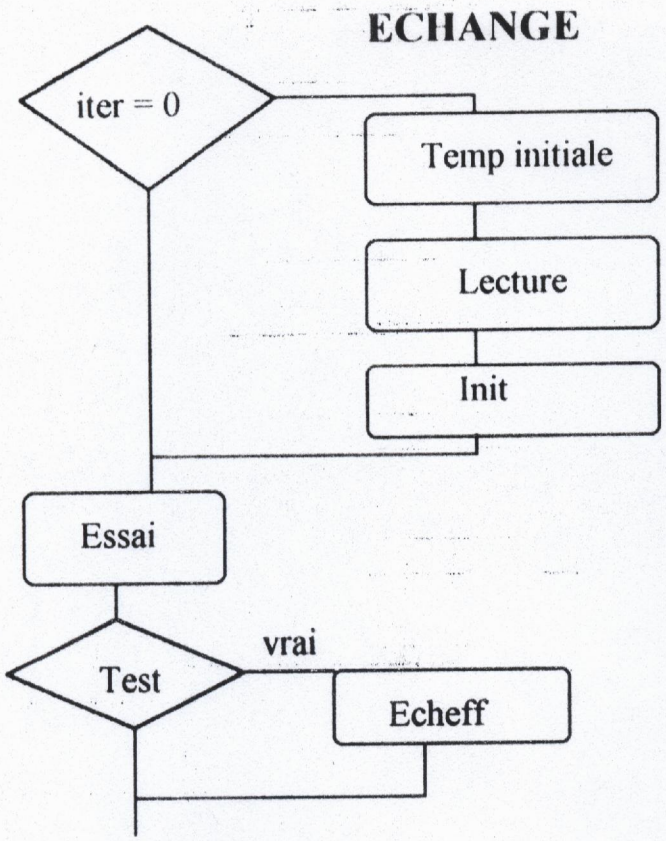
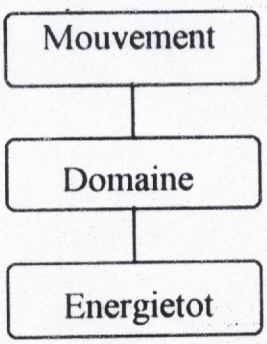


FIG V.14 PROCEDURES PRINCIPALES


ECHANGE
iter = 0
Temp initiale
Lecture
Init
Essai
Test
vrai
Echeff

ESSAI


Mouvement
Domaine
Energietot

III DEVELOPPEMENT D'UN SECOND LOGICIEL DE PLACEMENT.

Dans le même contexte, un second logiciel de placement des cellules dans un circuit intégré a été développé. Ce logiciel est basé sur l'algorithme de **recuit simulé** et de ce fait il utilise les mêmes techniques utilisées par le premier, à savoir: température initiale, longueur des paliers, test de **Metropolis** et critère d'arrêt.

Ce logiciel est spécifique au placement **Gate array**, où les cellules sont de même taille. Il optimise la longueur des connexions. Comparé au premier, l'optimisation est meilleure. Ceci est dû aux types de mouvements effectués par chacun d'eux.

Le premier, comme il a été déjà mentionné, effectue des mouvements de translation sur le même axe. Ceci risque de provoquer un placement plus ou moins long dans le cas où le nombre de cellules appartenant au même réseau est élevé.

Le second,même si ce nombre est assez élevé, essaye de les placer selon une matrice, en permettant aux cellules un placement dans les deux directions X et Y. Deux types de mouvements sont possibles: déplacement des cellule vers une place vacante et permutation entre elles. La longueur des connexions est calculée par la sommation des demi périmètres des **"bounding rectangle"** des différents réseaux.

Le programme démarre d'un état purement aléatoire engendré par la fonction **RANDOM** de turbo pascal. Au début de l'opération de recuit, la taille des **"bounding rectangle"** est celle de la puce. Au fur et à mesure que la température décroît, cette taille diminue jusqu'à ce que les **"bounding rectangle"** n'entourent que les cellules appartenant au réseau.

Contrairement au premier logiciel, la première étape effectuée par ce programme est la détermination des listes des equipotentielles d'une manière automatique. L'utilisateur doit seulement indiquer au programme les indices des

différentes broches correspondant à l'ensemble des réseaux.

Le programme présente a cette étape une procédure qui assure la vérification des différents réseaux introduits. Cette vérification concerne la présence de **circuits fermés**.

Vu le nombre très élevé de broches, l'utilisateur risque d'introduire des erreurs. Ces dernières peuvent provoquer des circuits fermés. Le programme avertit l'utilisateur par affichage d'un message d'avertissement sur écran tel que: " **ATTENTION CIRCUIT FERME AU RESEAU I** " **I** indique l'indice du réseau en question.

Pour la détermination de la valeur initiale de la fonction énergie, le programme effectue l'opération de recuit un nombre fini de fois. Cette valeur correspond à la moyenne des différentes valeurs des énergies calculées.

Le programme présente un fichier d'entrée. La première ligne du fichier donne les dimensions de la puce. Les lignes ultérieures correspondent aux différentes cellules. Chaque élément de la ligne est une broche de la cellules définie par l'indice du réseau auquel elle est connectée.

A la fin de l'exécution, le programme affiche la valeur de la fonction coût du meilleur score obtenu et indique les positions des différentes cellules définies par les coordonnées de leurs centres de gravité.

COMPARAISON DES DEUX LOGICIELS.

Malgré le type de mouvement limité utilisé par le premier programme, ce logiciel peut être adapté à n'importe quel type de contraintes par introduction des termes représentant ces contraintes au niveau de l'expression de la fonction de coût. Ceci fait de lui un programme assez général.

Le déroulement du programme sur écran n'est offert que par le premier, le second affiche directement les résultats ce qui empêche l'utilisateur d'avoir une idée sur les variations des différents paramètres de l'opération de recuit tels que la température, le pas et le nombre de mouvements effectués.

Etant donné que les deux logiciels utilisent l'algorithme de recuit simulé, connu par sa rapidité. Les temps d'executions de ces deux logiciels sont comparable et ne dependent que de la taille du circuit à simuler et du nombre de mouvements à effectuer pour la determination de la valeur initiale de la fonction de coût.

CONCLUSION

CONCLUSION

La complexité sans cesse croissante des circuits intégrés **VLSI**,a rendu la conception assistée par ordinateur indispensable. Celle ci intervient à tous les stades, notamment lors des étapes de placements des composants et le tracé des connexions.

Notre travail rentre dans ce cadre. Nous avons développé une étude théorique et comparative sur les différents types d'algorithmes de placement. Grâce à sa simplicité de mise en oeuvre et ses performances dans le traitement des problèmes complexes d'optimisation, l'algorithme du recuit simulé a été choisi et adapté afin de résoudre le problème de placement.

Le logiciel développé permet l'optimisation de la surface occupée par les cellules, la minimisation de la longueur des connexions qui est la contrainte de base pour le placement et la résolution du problème d'**overlaping** quand celui ci se présente.

Grâce à son aspect interactif, le programme permet à l'opérateur d'acquérir une certaine expérience qui peut le guider dans la recherche de l'optimum par le bon choix des paramètres contrôlant la convergence de celui ci. Il est a noter que le logiciel peut être adapté à n'importe quel type de contrainte, il suffit seulement d'introduire les termes représentant celle-ci au niveau de la fonction coût.

Lorsque le nombre des cellules à placer par réseau devient très élevé, le logiciel risque de donner des circuits plus ou moins long. Afin de remédier à ce problème, un second logiciel basé sur le même algorithme a été développé. Il utilise le principe des **"bounding rectangle"** pour le calcul de la longueur des connexions.

Par intégration du logiciel de routage à canal déjà développé, comme procédure au niveau du programme, celui ci pourra devenir un outil complet assurant le placement et le routage. de ce fait on aura éliminé une tâche manuelle fastidieuse où le risque d'erreurs est inévitable.

En réalité,cette etude ne présente qu'un début d'un travail de recherche de développement de logiciels de placement- routage puissants et rapides. En effet, le

placement des **macro cell** constitue un problème pour les chercheurs **[27]** et Le recuit simulé adapté, considéré comme étant sériel, est incapable de le résoudre **[8]**.

Suite à cela , plusieurs perspectives peuvent être envisagées:

- Parallèliser l'algorithme de recuit simulé (Hypercube) **[7][23]** en association avec la technique de partitionnement. Ceci pourra constituer une solution pour le placement des **macro cell**.

- Automatisation du placement initial: ceci permettra de démarrer d'un état très proche de l'état optimum ce qui se traduirait par un gain assez important en temps.

- Paralleliser l'opération de placement routage:ceci permettra d'éviter le problème de connexions tels que les croisements des pistes et débordement des canaux une fois le placement effectué.

BIBLIOGRAPHIE

[1]. **Aide du placement VLSI par reseaux de KOHONEN**; mémoire d'ingenieur, Université catholique de louvain;1992.

[2]. Jonathan rose and al."**Temperature measurement and equilibrium dynamics of simulated annealing placements**"; IEEE Trans on computer aided design vol 9 N°3 , pp 253_257; T; 1990.

[3]. P.Siarry gourevitch antoine and al."**Le recuit simulé**"; projet de recherche de seconde année laboratoire E.P.A.P école centrale de PARIS; 1989-1990.

[4]. R.H.J.M. Otten and L.P.P.P Van ginneken;"**The annealing algorithm**"; LONDON 1989.

[5]. Ouldali Belhachemi ;"**Recuit simulé et implantation des circuits électroniques:application à la technologie microélectronique hybride**"; Thèse de doctorat de l'université de PARIS , Juillet 1989.

[6]. Peter Kamyal suris and al."**A quadrisection based conbined place and route schema for standard cells**"; IEEE Trans on computer aided design , vol N°3 pp 234-243; 1989.

[7]. J.Brouwer and al."**A parallel algorithm for simultaneous placement and routing using hierarchy** "; IEEE European design automation conference, Brussels 1992.

[8]. M.D.Durand Columbia University;"**Accuracy VS.Speed in placement**";IEEE Design and test of computers , pp 9_33; 1989.

[9]. D.F.Wong,H.W.Leong,C.L.LIU;"**Simulated annealing for VLSI design**";USA 1988.

[10]. Peter R.Suaris and al."**An algorithm for quadrisection and its application to standard cell placement**"; IEEE Trans on circuits and systems, vol 35 N°3 pp 294_303; 1988.

[11]. Ralph M.King and al."**ESP:Placement by simulated evolution**";IEEE Trans on computer aided design, vol 8 , N°3, pp 245_255;1989.

[12]. Van Laarhoven;"**Simulated annealing:theory and application**";1987.

[13]. Patrick.Siarry and all."**Thermodynamic optimisation of block placement**";IEEE Trans on computer aided design, vol 6 N°2 pp 211_221;1987.

[14]. Masami Murakata,Atsushi Tanaka and al."**A standard cell placement algorithm width equalisation**";IEEE Trans on computer aided design ,pp 374_377JAPAN;1986.

[15]. Carl Schen and al."**The timberwolf placement and routing package**";IEEE Of solid state circuits, vol 20 N°2 , pp 510_521;1985.

[16]. Berndx.Weis and al."**A graph theoretic approach to the relatif placement problem**";IEEE Trans on computer aided design, 1988.

[17]. Yvon.Savaria;"**Conception et vérification des circuits VLSI**";Ecole polytechnique de MONTREAL;1988.

[18]. S.Kirkpatrick,C.D.Gelah Jr,M.P.Vecchi;"**Sence**";1983.

[19]. P.Siarry and al."**An applicate of physical methods to the computer aided design of électronic circuits**";janvier 1984.

[20]. F.Romeo,A.L.Sangiovanne and al.IEEE int.conf on computer design; Port chester 1984.

[21]. P.Siarry;"**La méthode du recuit simulé:Application à la conception des circuits électroniques**";Thèse PARIS 6; novembre 1986.

[22]. V.Cerny;"**Optimisation theory and application**";1985.

[23]. Chia-Chuntsai and al."**Planning strategies for area routing**";IEE proceeding G pp 338_347; 1992.

[24].Vsing,C.Y.R.Chen;"**Technique for 1-dimentional VLSI layout generation**"; IEE proceeding G, vol 139 N°6;Décembre 1992.

[25]. Sy.Yen Kuo;"**Yor:Ayield optimizing routing algorithm by minimizing critical areas and vias**";IEEE Trans on computer design pp 525_529 TAIWAN 1992.

[26]. E.S.Kunh and al."**Two-dimentional compaction for placement rafinement**";IEEE Trans on computer aided design vol 7 N°3 pp 136_139; 1989.

[27]. Samuel,Winner and al."**Analysis of strategies for constructive general block placement**";IEEE Trans on computer aided design, vol 7 N°3 pp 371_377 1988.

[28]. Ren.Song Tsay and al."**Proud:A sea of gates placement algorithm**";IEEE