

UNIVERSITY NAME

DOCTORAL THESIS

---

# Thesis Title

---

*Author:*

John SMITH

*Supervisor:*

Dr. James SMITH

*A thesis submitted in fulfillment of the requirements  
for the degree of Doctor of Philosophy  
in the*

Research Group Name  
Department or School Name

December 18, 2017



# Declaration of Authorship

I, John SMITH, declare that this thesis titled, “Thesis Title” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

---

Date:

---



*"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."*

Dave Barry



UNIVERSITY NAME

# *Abstract*

Faculty Name  
Department or School Name

Doctor of Philosophy

**Thesis Title**  
by John SMITH

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...





# *Acknowledgements*

The acknowledgments and the people to thank go here, don't forget to include your project advisor...



# Contents

<b>Declaration of Authorship</b>	<b>iii</b>
<b>Abstract</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>1 Placement</b>	<b>1</b>
1.1 Introduction	1
1.2 État de l'art des algorithmes de placement	2
1.2.1 Les représentations topologiques de plan de masse	2
Les slicing models	3
Les sequence pair	4
Les O-trees	5
Les B*-trees et HB*-trees	6
Les Transitive Closure Graph	7
Sommaires des contraintes des représentations	7
1.2.2 Les méthodes d'optimisation de placement	8
1.3 Implémentation	10
1.3.1 Notre approche de placement	10
1.3.2 Méthodologie de placement	13
Définition d'une topologie de placement	14
Expression des contraintes de placement	15
Algorithme de placement	18
1.4 Exemples	20
1.5 Conclusion	20
<b>2 Problem Definition</b>	<b>21</b>
<b>3 State of the Art</b>	<b>23</b>
<b>4 Academic Tools</b>	<b>25</b>
<b>5 Industrial Tools</b>	<b>27</b>
<b>6 Challenges</b>	<b>29</b>
<b>7 Analog/Mixed-signal Design Approach</b>	<b>31</b>
7.1 Design Flow	31
7.2 Placement in row	32

<b>8</b>	<b>Methodology for the placement</b>	<b>33</b>
8.1	Slicing Tree Construction . . . . .	33
8.2	Devices Variations . . . . .	34
8.3	Margin Tolerances . . . . .	35
8.4	Placement Constraints . . . . .	35
8.5	Placement Choice . . . . .	36
<b>9</b>	<b>Placement Algorithm</b>	<b>37</b>
9.1	Theory . . . . .	37
9.2	Results . . . . .	38
<b>10</b>	<b>Future Work: Routing</b>	<b>41</b>
10.1	Global Routing . . . . .	41
10.1.1	Detailed Routing . . . . .	41
<b>A</b>	<b>Frequently Asked Questions</b>	<b>43</b>
A.1	How do I change the colors of links? . . . . .	43

# List of Figures

1.1	Exemple de placement et sa représentation en <i>slicing tree</i> ou "H" définit une coupe horizontale et "V" une coupe vertical. . . . .	3
1.2	Exemple d'un placement représentable par un <i>slicing tree</i> (a) et d'un placement non représentable (b) . . . . .	4
1.3	(a) Échelons positifs résultant et $\Gamma_+ = ecadfb$ . (b) Échelons négatifs résultant et $\Gamma_- = fcbead$ . . . . .	5
1.4	(a) Une représentation en <i>O-tree</i> et (b) son placement correspondant. . . . .	5
1.5	(a) Placement compact. (b) Représentation en <i>B*-tree</i> du placement compact (a). . . . .	6
1.6	(a) Placement compact. (b) Représentation en <i>TCG</i> du placement compact (a). . . . .	7
1.7	Notre approche permet la génération rapide de plusieurs placements . . . . .	12
1.8	Difficultés à écarter un placement non <i>slicing-tree</i> . . . . .	13
1.9	Evolution des découpes d'un <i>slicing tree</i> . . . . .	14
1.10	(a) Placement représentable par un <i>slicing tree</i> (b) Placement non représentable par un <i>slicing tree</i> . . . . .	15
1.11	Les paramètres réglables des devices, (a) Style de dessin des masques, (b) Variation du nombre de doigts et (c) Variation de la taille des connecteurs. . . . .	16
1.12	Espace de routage vertical et horizontal . . . . .	17
1.13	Noeud hiérarchique vertical avec une organisation en bande . . . . .	18
7.1	Cairo Hurricane AMS Design Flow . . . . .	31
9.1	Evolution of 2 rows from layout 1 and 6 of Table II for different global aspect ratios. These figures have the same scale. . . . .	39
9.2	Layouts of the fully differential transconductor cite19 from Table II. These figures have all the same scale. . . . .	40
10.1	Dijkstra pathfinding algorithm . . . . .	41



# List of Tables

1.1	Tableau récapitulatif de l'état de l'art des représentations de placement pour les circuits analogiques . . . . .	8
1.2	Comparison of area usage and runtime for different approaches on 1-S symmetry placement [Xia+10] . . . . .	12
9.1	Some layout results of the fully differential transconductor . . . . .	39





# List of Abbreviations

**LAH** List Abbreviations Here  
**WSF** What (it) Stands For



# Physical Constants

Speed of Light  $c_0 = 2.997\,924\,58 \times 10^8 \text{ m s}^{-1}$  (exact)



# List of Symbols

$a$	distance	m
$P$	power	W (J s <sup>-1</sup> )
$\omega$	angular frequency	rad



*For/Dedicated to/To my...*





# Chapter 1

## Placement

### 1.1 Introduction

Les systèmes-sur-puce modernes contiennent tant des circuits numériques que des circuits analogiques. La conception de circuits numériques a été particulièrement automatisée par des outils de conception assistée par ordinateur tandis que la conception de circuits analogiques est restée manuelle. Avec l'évolution des nouvelles technologies nanométriques, cela implique un travail manuel long et sujet à l'erreur humaine.

Dans le cadre des circuits analogiques, les parasites du dessin des masques et les contraintes de procédés de fabrication augmentent drastiquement la complexité de la tâche quant au placement des modules du circuit. Il devient essentiel d'introduire des outils de placement dédiés aux circuits analogiques dans le but d'accélérer aussi bien le cycle de conception que l'effort de conception.

La phase de placement pour les circuits mixtes et analogiques a une influence importante sur les performance du circuit. La relation de placement entre l'ensemble des modules, et plus particulièrement celle entre les modules appariés, est importante afin de préserver au plus possible l'environnement adéquate de fonctionnement des modules en tenant compte des parasites engendrés par le dessin des masques.

Dans ce chapitre, nous discuterons de l'étude du problème du placement des circuits mixtes et analogiques.

Dans la section 1.2.1, nous énumérons les principales représentations topologiques du plan de masse de l'état de l'art. Ces représentations utilisent des graphe traduisant des relations de placement entre l'ensemble des modules d'un circuit. Leur spécificité avec leurs avantages et inconvénients et les contraintes qu'elles sont capables de prendre en compte dans le choix d'un placement optimal seront explicités.

Dans la section 1.2.2, nous exposerons la phase d'optimisation que la majorité de l'état de l'art utilise dans le cadre de la recherche d'une solution de placement optimale. Les algorithmes de recuit simulé ont souvent opté pour leur capacité à produire de bons résultats en temps suffisamment court. Ces algorithmes sont

également relativement simples à implémenter et peuvent être facilement enrichis pour des conditions supplémentaires.

Dans la section 1.3, nous présenterons notre approche du problème du placement des circuits mixtes et analogiques. Notre représentation topologique ainsi que notre méthodologie d'optimisation y sera détaillées. Cette section contiendra une partie de notre structure de données utilisée tout au long de la génération du placement routage.

Dans la section 1.4, les détails de la méthodologie de placement seront illustrés par un exemple contenant des contraintes de placement répandues. Les interventions du concepteur seront énumérés avec clarté dans le but de mettre en avant son rôle dans le choix de la solution de placement.

## 1.2 État de l'art des algorithmes de placement

Le problème de placement abordé par les outils de CAO dédiés aux dessins des masques numériques et analogiques consistent à explorer un grand espace de solutions de configurations de placement faisables et non faisables en utilisant une représentation du plan de masse couplée à une optimisation stochastique telle que l'algorithme de recuit simulé. A la différence des circuits numériques, les circuits analogiques doivent prendre en considération des contraintes de placement supplémentaires liées aux parasites du dessin des masques. L'état de l'art des approches de placement pour circuits mixtes et analogiques sera abordé dans les sections suivantes.

### 1.2.1 Les représentations topologiques de plan de masse

Afin de pouvoir générer un placement légal respectant plusieurs contraintes de placement analogiques, la majorité des études récentes emploie une représentation sous forme de graphe. On distingue deux catégories de représentations: les représentations absolues et les représentations topologiques. Les représentations absolues [JJ83] sont principalement dans les anciennes méthodes de placement et consiste à associer à chacun des modules du circuit une coordonnée par rapport à un point de référence. Cette représentation permet la superposition illégale de modules lors de la phase d'optimisation dû au fait qu'il n'existe pas de relation de placement entre les modules. Cette approche doit alors être en mesure d'explorer un très grand espace de solutions comprenant tant des placements faisables que non faisables. Cela se traduit par des temps d'exécution long causés par le grand nombre de mouvements nécessaires pour obtenir un dessin des masques satisfaisant. Il est également possible qu'en plus du long temps d'exécution, il ne garantisse pas toujours de solutions faisables. L'ajustement de la fonction de coût pour éviter les recouvrements de modules peut requérir un effort conséquent.

En opposition aux représentations absolues, les représentations topologiques ne font pas face à des problèmes de recouvrements, elles consistent à définir des

relations de positions entre les modules d'un circuit. Ces représentations sont largement utilisés pour des raisons d'efficacité et de flexibilités à pouvoir satisfaire des contraintes pour un moindre coût en termes de nombre de mouvements comparées aux représentations absolues. On présentera dans les sous-sections suivantes les représentations topologiques les plus utilisées de l'état de l'art.

### Les slicing models

Les *slicing models* [WL86] font parties des premières représentations topologiques utilisées. Ces *slicing models* sont représentés à l'aide d'un graphe appelé *slicing tree*. Un *slicing tree* est un arbre dans lequel chaque noeud interne peut être obtenu par découpes récursives du circuit (Fig. 1.1). A partir de la région totale du circuit représentant la racine de l'arbre, le circuit sera découpé de manière hiérarchique, alternativement de manière horizontal et vertical, jusqu'à atteindre les feuilles de l'arbre représentant les modules du circuit ou bien des espaces de routage. L'ensemble des rectangles terminaux dû aux découpes sera toujours représenté par une feuille.

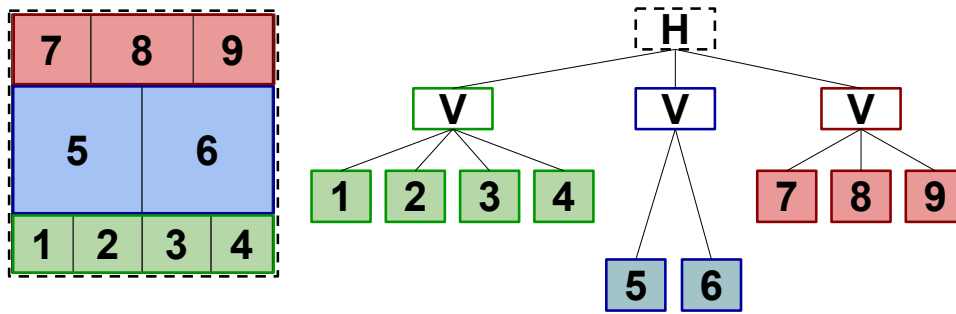


FIGURE 1.1: Exemple de placement et sa représentation en *slicing tree* ou "H" définit une coupe horizontale et "V" une coupe vertical.

[AJ96] concentre son optimisation sur la surface totale du circuit ainsi que la longueur total des *nets* dont l'amélioration est 5 fois plus courte comparé à la non prise en considération. L'optimisation prend en compte les contraintes de proximité, de modules pré-placés et de symétries. [Pri+97] considère les parasites d'interconnection dans la phase d'optimisation avec une prise en compte simultanée du placement et routage. Ils sont en mesure de maintenir des symétries de groupes de modules. [YW98] étend l'approche de [WL86] afin de pouvoir considérer des modules pré-placés, de contraintes de bordure [YWY99] et de proximité [YWY00]. [Lin+12] améliore la formulation des *slicing trees* en introduisant des conditions de symétries pour des groupes de modules. [Wu+12] introduit la prise en compte des contraintes de sens du courant dans les structures en *slicing tree* en plus des symétries.

Puisque tous les circuits n'ont pas une structure en tranche, le désavantage de cette topologie est que la densité de la solution de placement peut être dégradé ce qui peut être notable lorsque les modules d'un circuit ont des rapports hauteur/largeur très hétérogènes. De plus, la structure des *slicing trees* ne permet pas

de représenter toutes les placement possibles (Fig. 1.2). Suite à ces limitations, des représentations avec des structures *non-slicing* se sont succédé. Les *sequence pairs*, les *B\*-trees*, les *Ordered-Trees - O-tree* et les *Transitive Closure Graphs - TCG* seront présentés dans les sous-sections suivantes.

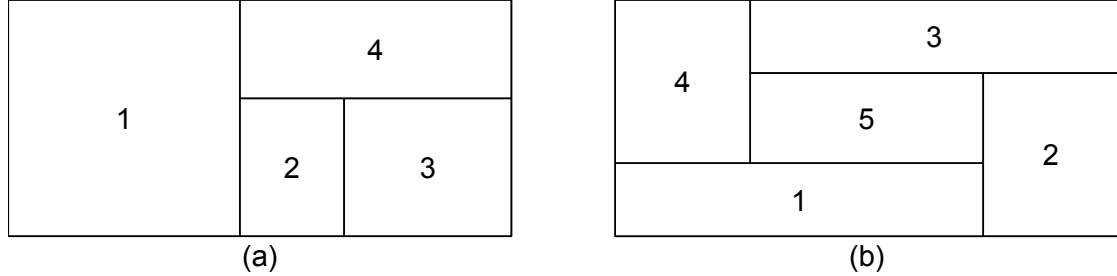


FIGURE 1.2: Exemple d'un placement représentable par un *slicing tree* (a) et d'un placement non représentable (b)

### Les *sequence pair*

Les *sequence pairs* sont utilisés dans le contexte des placements de circuits analogiques pour la première fois par [BL00]. Une *sequence pair* donnée pour ensemble de modules est une paire de séquences contenant l'ensemble des modules. Par exemple,  $(abc, cba)$  est une *sequence pair* pour l'ensemble de module  $\{a, b, c\}$ . Pour un placement donné, la *sequence pair* correspondante est obtenue par la construction des échelons positifs et négatifs de chacun des modules (Fig. 1.7). L'échelon positif (négatif) d'un module contient 3 parties: l'échelon haut-droit (gauche-haut), l'échelon bas-gauche (droite-bas) et la ligne diagonale effectuant la connexion. L'échelon haut-droite (bas-gauche, gauche-haut, droite-bas respectivement) d'un module est  $A$  est dessiné en suivant les règles suivantes:

1. On part de coin supérieur droit (inférieur gauche, supérieur gauche et inférieur droit respectivement) du module  $A$ .
2. On change la direction alternativement haut (bas, gauche et droite respectivement) et droit (gauche, haut et bas respectivement) jusqu'à atteindre le coin supérieur droit (inférieur gauche, supérieur gauche et inférieur droit respectivement) du placement sans croiser:
  - Les bordures des autres modules
  - Les lignes précédemment tracées

En utilisant l'ordre des séquences des échelons positifs et négatifs (de la gauche vers la droite), une paire ordonnée,  $(\Gamma_+, \Gamma_-)$ , de séquences de modules peut être placée. La Fig. 1.7 montre le placement résultant de ses échelons négatifs et positifs avec de ses séquences respectives. A partir d'une *sequence pair*, il est également possible d'en dégager les contraintes géométriques (horizontal ou vertical) entre deux modules.

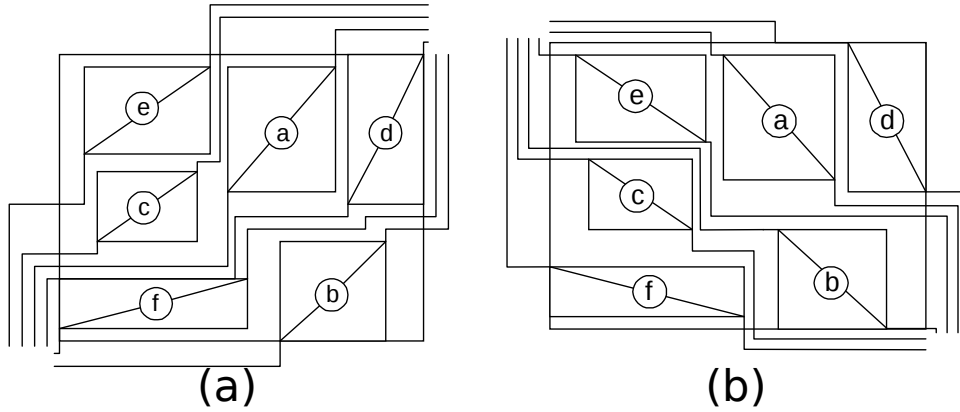


FIGURE 1.3: (a) Échelons positifs résultant et  $\Gamma_+ = ecadfb$ . (b) Échelons négatifs résultant et  $\Gamma_- = fcbead$ .

H. Murata et al. [Mur+96] sont les premiers à introduire le concept de *sequence pair* pour des applications de dessins des masques VLSI durant une période pendant laquelle les *slicing models* étaient populaires mais ne produisaient pas des résultats assez satisfaisant. [BL00] adresse l'utilisation des *sequence pairs* dans le cadre du placement pour circuits analogiques avec une prise en compte de contraintes de symétries et [BM01] améliore leur temps de calcul par la suite. [TYC06] ajoute la considération d'alignement et de contraintes de proximités des modules en plus des symétries. [DXS06] oriente leur attention sur du placement vis à vis du sens du courant tout en tenant compte des symétries. [Shi+10] porte leur attention sur les contraintes de régularités couplés aux contraintes de symétries. [Xia+10] prends en considération des contraintes de congestions pour le routage en évaluant la congestion des placements donnés.

### Les O-trees

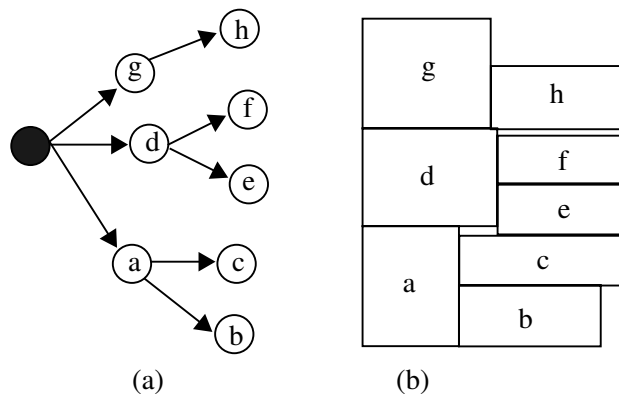


FIGURE 1.4: (a) Une représentation en *O-tree* et (b) son placement correspondant.

Un (*Ordered Tree (O-tree)*) étend le principe des *B\*-tree* à une représentation du plan de masse de type *non-slicing* avec une complexité inférieure à celle des *slicing models*. Étant donné un placement de  $n$  modules, un *O-tree* correspondant possède  $n + 1$  noeuds encodé par  $(T, \pi)$ , où  $T$  correspond à une chaîne de caractères de  $2n$ -bit identifiant la structure de l'arbre et  $\pi$  représente l'ordre des noms des modules sans considérer la racine de l'arbre. Lorsqu'on parcourt l'arbre en profondeur, "0" correspondant à une arête descendante, et "1" à une arête ascendante. Cette méthode permet de réduire les redondances et le temps d'exécution est plus rapide que celui des représentations en *sequence pair*.

Cette méthodologie est introduite par [GCY99], [Pan+00] et [XY09] élargies les types contraintes de symétries considérées.

### Les *B\*-trees* et *HB\*-trees*

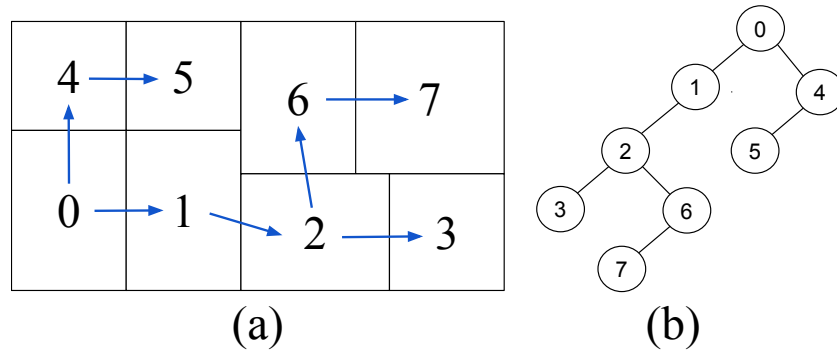


FIGURE 1.5: (a) Placement compact. (b) Représentation en *B\*-tree* du placement compact (a).

Les (*Ordered Binary Trees - B\*-trees*) sont des arbres couramment utilisés pour représenter un placement compact de modules dans lequel tous les modules ne peuvent plus se déplacer vers la gauche et vers le bas. Chaque module d'un circuit est représenté par un noeud dans un *B\*-tree*. La racine d'un *B\*-tree* correspond au module situé dans le coin inférieur gauche, il s'agit du module 0 sur la Fig. 1.5. Le noeud de gauche du module 0 représente le module de droite et adjacent au module 0, il s'agit du module 1. Le noeud de droite du module 1 représente le module au dessus et adjacent au module 0, il s'agit du module 4. La Fig. 1.5 montre un placement et sa représentation en *B\*-tree* correspondante.

Les *B\*-trees* sont introduits par [Bal00] et [Cha+00] dans le but d'utiliser une représentation succédant aux *O-trees*. Cette approche est améliorée en incorporant les contraintes de symétrie par [BMK02] dans le cadre des circuits analogiques. [Mar+05] améliore le temps d'exécution de 20% à 30%. [Str+08] propose une approche déterministe permettant une meilleure reproductibilité du placement.

La représentation des *B\** avec s'enrichie avec l'utilisation des *Hierarchical Ordered Binary Trees - HB\*-trees* [LCL09] dans le but d'ajouter des contraintes hiérarchies et groupes de symétries. [Lin+10] y ajoute la prise en compte de contraintes

de bordure. Cette représentation est ensuite utilisée pour différentes contraintes d'optimisation, telle que pour des contraintes de température [Lin+11] ou de régularité [COC11] et avec de meilleure optimisation de temps d'exécution [Tsa+11].

### Les Transitive Closure Graph

Les *Transitive Closure Graphs* (TCG) décrivent les relations géométriques entre les modules d'un circuit en se basant sur deux graphes: un *Horizontal transitive closure graph*  $G_h$  et un *Vertical transitive closure graph*  $G_v$ . Dans le  $G_h$  (respectivement  $G_v$ ), une arête  $\langle v_i, v_j \rangle$  exprime que le module  $m_i$  se trouve à gauche de (respectivement en bas de) la cellule  $m_j$ . Le poids associé à l'arête dans  $G_h$  (respectivement  $G_v$ ) correspond à la largeur (respectivement hauteur) du module associé. La Fig. 1.6 montre un exemple de placement avec sa représentation TCG.

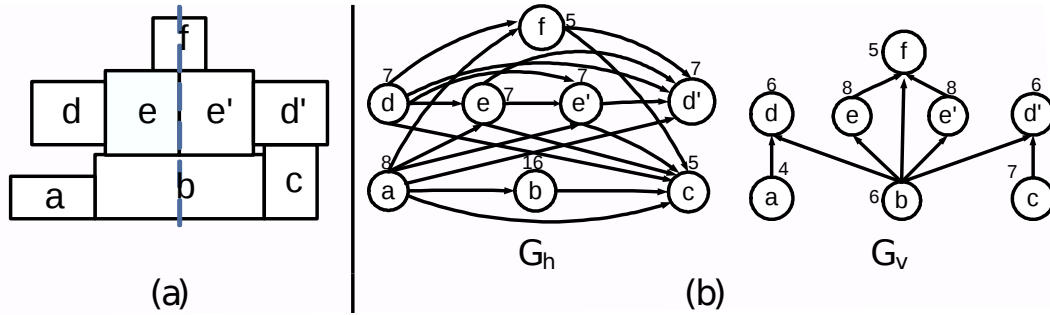


FIGURE 1.6: (a) Placement compact. (b) Représentation en TCG du placement compact (a).

Les *Transitive Closure Graphs* (TCG) sont introduits par [LC01] dans le contexte d'opter pour des solutions différentes des *sequence pairs* et des *B\*-trees*. [ZSJ08] y ajoute la considération des contraintes de symétrie. Les *Transitive Closure Graphs* (TCG) sont étendus au TCG-S par [LC04] et [Lin+05] avec lesquels la phase de compactage est plus performante est inspiré de celle des *sequence pairs*.

### Sommaires des contraintes des représentations

Au cours de ces deux dernières décennies, de nombreuses représentations topologiques ont été utilisées en couplage avec diverses contraintes. L'ensemble des articles mentionnés précédemment sont rassemblés dans le tableau récapitulatif 1.1. A partir d'une représentation du plan de masse considérant les contraintes du concepteur, il en vient ensuite la phase d'optimisation du placement. Elle consiste à explorer l'espace des solutions de placement afin de trouver une solution en un temps raisonnable.

TABLE 1.1: Tableau récapitulatif de l'état de l'art des représentations de placement pour les circuits analogiques

Contraintes	Slicing-tree	Sequence Pair	O-tree	B*-tree et HB*-tree	TCG
Symmétries	[AJ96], [Pri+97], [Lin+12], [Wu+12]	[BL00], [BM01], [TYC06], [Shi+10], [Xia+10]	[Pan+00], [XY09]	[Bal00], [BMK02], [Mar+05], [Str+08], [LCL09], [Lin+10], [Lin+11], [COC11], [Tsa+11]	[ZSJ08], [Lin+05]
Symétrie de groupes	[Wu+12]	[Xia+10]		[LCL09], [Lin+10], [Lin+11]	
Centrage géométrique		[Xia+10]	[XY09]	[Str+08]	
Sens du courant	[Wu+12]	[DXS06]			
Température				[Lin+11]	
Régularité		[Shi+10]		[COC11]	
Proximité	[AJ96], [YWY00]	[TYC06]		[Str+08]	
Régularité				[Bal00]	
Modules pré-placés	[AJ96], [YW98]			[Tsa+11]	
Bordures	[YWY99]			[Lin+10], [Tsa+11]	
Routage	[Pri+97]	[Xia+10]			

### 1.2.2 Les méthodes d'optimisation de placement

La majeure partie des approches de l'état de l'art utilise une représentation topologique parmi celles mentionnées dans la section précédente, avec laquelle on applique la méthode du recuit simulé. Cette méthode introduite par [KGV+83] est une technique probabiliste d'approximation de l'optimum global d'une fonction donnée. C'est à dire qu'il s'agit d'une métaheuristique permettant d'estimer l'optimisation globale dans un grand espace de recherche. Le principe s'inspire du processus de recuit des métaux en métallurgie dans lequel le refroidissement d'un matériau est contrôlé. Cela se traduit par une probabilité d'acceptation dégressive des mauvaises solutions dans l'espace de solutions exploré.

Le fonctionnement de l'algorithme du recuit simulé se déroule par les étapes suivantes:

1. L'algorithme sélectionne une transformation aléatoire prédéfinie sur la solution courante
2. On mesure la qualité de la solution sélectionnée
3. On sauvegarde la nouvelle solution ou bien on décide maintenir la solution courante en se basant sur une probabilité dépendant de la qualité précédemment évaluée



4. Le paramètre de température diminue se traduisant par une augmentation de la probabilité du choix de la meilleure solution et une diminution de la probabilité du choix de mauvaise qualité

Le tirage au sort (étape 1) des transformations est déterminé en fonction de l'approche choisie. Ces transformations dépendent de la représentation utilisée, une transformation revient à perturber le graphe représentant le placement de la solution courante. Il peut s'agir d'une rotation d'un module ou d'un échange de position entre deux modules.

La spécification des contraintes de placement d'un concepteur se définit par la formulation d'une fonction de coût qui nous permet d'évaluer la qualité (étape 2) de la solution sélectionnée. Afin d'illustrer le fonctionnement général d'une fonction de coût, considérons une optimisation sur une contrainte de surface global et d'une longueur de fil, la fonction de coût se présenterait de la façon suivante:

$$Cost(F) = \alpha.A_{normalisé} + \beta.W_{normalisé} \quad (1.1)$$

avec  $\alpha$  et  $\beta$  étant des paramètres de réglages permettant d'influencer l'importance de la contrainte considérée,  $A_{normalisé}$  représentant la surface total occupée normalisée et  $W_{normalisé}$  la longueur de fil totale normalisée.

## 1.3 Implémentation

Parmi l'ensemble des méthodologies de placement analogique, la majorité d'entre elle présente une capacité à produire un placement tenant compte d'une ou plusieurs contraintes du concepteur. L'utilisation de métaheuristique permet de produire des résultats optimisés pour les contraintes considérées dans la fonction de coût. Néanmoins, la complexité des contraintes des circuits analogiques et mixtes peuvent rendre difficile l'ajustement des paramètres de la fonction de coût. Dans la section suivante, nous présenterons l'approche pour laquelle nous avons opté ainsi que les raisons qui nous ont poussés à faire ces choix. Un exemple illustrera notre méthodologie et notre outil de placement.

### 1.3.1 Notre approche de placement

La phase de placement est une étape complexe dans la réalisation d'un circuit tant pour les circuits numériques qu'analogiques mais se différencie par leur problème. Dans le cadre des circuits numériques, le problème consiste à placer un grand nombre de cellules tout en minimisant la surface occupée et la longueur de fils des interconnexions. En règle générale, la phase de placement est découplé de la phase de routage. Quant au problème de placement des circuits analogiques, il consiste à placer un plus faible nombre de cellules tout en respectant les deux contraintes mentionnées précédemment, mais également une multitude de contraintes simultanément pouvant provenir de plusieurs domaines (électrique, mécanique ou bien thermique).

Ces considérations ont amenées à la création de représentations topologiques et méthodes d'optimisation mentionnée dans les sous-sections 1.2.1 et 1.2.2. En étudiant l'état de l'art du placement analogique, nous pensons que certains aspects nécessite une particulière attention ainsi que les limites d'une grande majorité des approches. Ce sont en tenant compte des considérations suivantes que nous avons effectué des choix dans le cadre de notre approche de placement:

- **Interventions du concepteur:** La conception du dessin des masques dans le domaine numérique est une étape particulièrement automatisé par rapport au domaine analogique. Du fait de la complexité du jeu de contraintes à gérer, nous jugeons qu'il est préférable de ne pas rendre l'automatisation du placement et du routage aussi automatique que dans le domaine numérique. Le placement d'un circuit dépend de plusieurs facteurs. L'environnement dans lequel va fonctionner le circuit comprend la prise en compte des effets des procédés de fabrication et des leurs effets parasites. Le facteur qui nous paraît particulièrement important est que l'expérience d'un concepteur peut énormément influencer les choix de placement dans la gestion des compensations des effets parasites. Sans la prise en compte de cette expérience, les outils de placement pour circuits analogiques produiront la plupart du temps des résultats insatisfaisant. Il en vient qu'il est nécessaire que le concepteur ait la possibilité des d'effectuer des interventions significatives. Notre approche s'oriente vers une approche où l'automatisation est moindre

en comparaison à l'état de l'art mais l'influence du concepteur sur le résultat final d'avantage prise en compte.

- **Gestion des contraintes:** On observe dans l'état de l'art qu'une grande diversité de contraintes pour chaque représentation topologique en particulier pour les *B\*-trees*. On remarque que la gestion des contraintes est relativement automatisé, il n'est pas toujours évident de connaître la capacité des paramètres à être modifier. L'ajustement de ces paramètres est généralement fixé suite à de nombreux essais et ne peut être l'unique manoeuvre de jeu du concepteur. Dans notre méthodologie de placement, les contraintes gérées de manière automatique sont uniquement les contraintes de symétries. Nous utilisons une approche en *slicing tree*, cette représentation sera défini par le concepteur. Définir le *slicing tree* revient à laisser le choix aux concepteurs de juger l'importance de chacune des contraintes nécessaires à être prise en compte.
- **Ajustabilité du placeur:** Le recuit simulé associé à une des représentations de l'état de l'art permet de générer des placements suivant en certains nombres de critères. Malgré la qualité des placements produits, il est souvent nécessaire de réaliser des modifications diverses. Il peut s'agir de modification de placement entre deux blocks ou bien un simple repositionnement de certains modules. Dans la mesure où les concepteurs souhaitent réaliser ce type de modifications, il peut s'avérer difficile de réajuster la représentation topologique ou bien encore la position de l'ensemble du circuit pour des raisons de compactions. Dans le cadre du *slicing tree*, le repositionnement ou bien le changement de taille d'un module est facile de part la structure du plan de masse en découpe. Tout comme mentionné dans le point précédent, la définition du *slicing tree* est une tâche réalisée par le concepteur, lui permettant d'avoir un contrôle important sur le placement relatif des modules.
- **Placement déterministe:** Les algorithmes d'optimisation ne permettent pas toujours de générer des placements prévisibles. Certains d'entre eux permettent une certaine reproductibilité à maintenant les jeux de paramètres du recuit simulé en tenant compte de la température de départ et de la fonction de refroidissement. Néanmoins, les aspects aléatoires des probabilités des choix de placement peuvent rendre l'algorithme non déterministe. Le non-déterminisme rend l'évaluation des paramètres d'entrées difficiles ainsi que l'ajustabilité du placeur. L'approche que nous avons adopté permet un calcul des placements valides de manière déterministe. Nous jugeons que cette caractéristique est particulièrement utile dans les phases de debug.
- **Choix multiple de placements:** Le recuit simulé permet de générer un unique placement optimisé respectant au mieux les contraintes données considérées en entrée. Pour les mêmes raisons mentionnées dans les points précédents, cet unique placement sera probablement rarement choisi par tous les concepteurs. Nous pensons qu'il est nécessaire d'avoir la capacité à générer plusieurs placements afin que les concepteurs puissent choisir une solution qui lui conviennent. Certains placements peu compacts pourraient être des

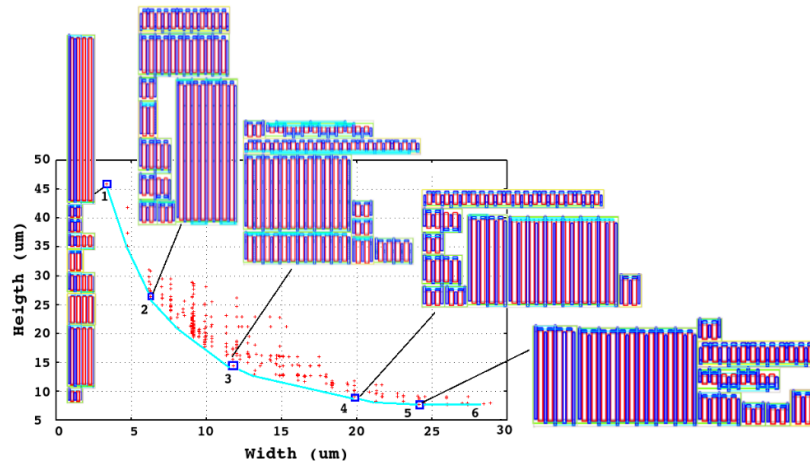


FIGURE 1.7: Notre approche permet la génération rapide de plusieurs placements

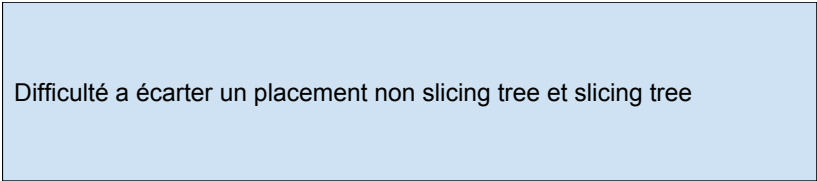
résultats de placements plus satisfaisant aux yeux des concepteurs dans certaines situations. Les placements identifiés par notre outil de placement sont disposés sur un graphe où chaque placement est représenté par un point défini par la largeur et hauteur du placement global.

- **Placement déterministe:** Les algorithmes d'optimisation ne permettent pas toujours de générer des placements prévisibles. Certains d'entre eux permettent une certaine reproductibilité à maintenant les jeux de paramètres du recuit simulé en tenant compte de la température de départ et de la fonction de refroidissement. Néanmoins, les aspects aléatoires des probabilités des choix de placement peuvent rendre l'algorithme non déterministe. Le non-déterminisme rend l'évaluation des paramètres d'entrées difficiles ainsi que l'ajustabilité du placeur. L'approche que nous avons adopté permet un calcul des placements valides de manière déterministe. Nous jugeons que cette caractéristique est particulièrement utile dans les phases de debug.

TABLE 1.2: Comparison of area usage and runtime for different approaches on 1-S symmetry placement [Xia+10]

Data Set	Block No.	1-D Groups	Sequence Pair		Slicing Tree		HB*-tree		B*-tree	
			Area	Time	Area	Time	Area	Time	Area	Time
Bench 1	65	8,12,5	114.9	780	114.9	246	104.7	22	104.9	337
Bench 2	110	16,6,6,12,4	110.4	2824	109.4	726	105.7	43	107.7	387

- **Temps d'exécution faible:** Les contraintes de placement pour les circuits analogiques ainsi que le nombre inférieur de cellules réduit fortement l'espace de solutions en comparaison avec le problème du placement des circuits numériques. Le temps d'exécution du recuit simulé nécessite un certain temps quelque ce soit le type de topologie employé [Xia+10]. La génération de multiple placements de notre approche nécessite des temps de placements inférieurs à ceux de l'état de l'art. Le temps requis pour passer d'un placement à un autre est de l'ordre de l'interactif.



Difficulté à écarter un placement non slicing tree et slicing tree

FIGURE 1.8: Difficultés à écarter un placement non slicing-tree

- **Considérations pour la phase de routage:** Le point le plus crucial dans la comparaison entre notre approche de placement et l'état de l'art. La majeure partie de l'état étudie le problème de placement des circuits analogiques ne manière découplé avec la phase de routage. Or la phase de routage ne peut être totalement ignorée dans la phase de placement. La situation concrète justifiant cela est qu'un placement peut être considéré placé de manière optimiser sur la base de critères de surface occupée. Un circuit trop congestionné ne laissera pas assez de place pour les fils de routage. Contrairement aux circuits numériques, certains fils de routage ne peuvent être tirés au-dessus de certains parties analogiques. Écarter un circuit compact deviendrait alors un travail extrêmement complexe et dénaturerait le résultat de placement initial. De part sa structure, le *slicing tree* n'est pas reconnu comme étant la représentation topologique la plus compacte mais elle est particulièrement flexible pour considérer des espaces dédiées aux fils de routage. En plus de la représentation topologique, nos outils sont uniformes d'un point de vue logiciel. Les objets de manipuler lors de la phase de placement sont réutiliser dans le cadre du routage permettant évitant ainsi des pertes d'informations lors de conversion ou des problèmes couplages entre deux outils développés distinctement.

### 1.3.2 Méthodologie de placement

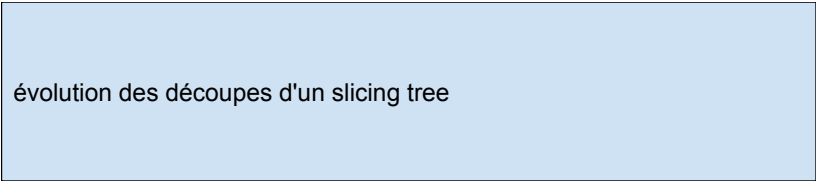
Notre méthodologie de placement consiste à effectuer un placement automatisé tout en impliquant des prises de décisions des concepteurs. L'objectif est de permettre à chaque concepteur de pouvoir appliquer ses propres contraintes sans engendrer une automatisation trop avancée. Cela entraînera des résultats moins optimisés en termes de surface mais en contre partie, le placement seront plus prévisibles et plus simplement ajustables. L'organisation de notre méthodologie se présente de la manière suivante:

- **La topologie en *slicing tree*:** le concepteur définit une position relative des modules.
- **Expression des contraintes de placement:** le concepteur introduit un ensemble de contraintes de placement.
- **Algorithme de placement:** l'outil établit les placements possibles et génère un placement choisi par le concepteur.

### Définition d'une topologie de placement

Lors de la conception d'un système sur puce (SoC), les espaces dédiés aux circuits numériques et analogiques sont distincts afin que chacun des circuits puissent être conçus de manière indépendante en terme de surface de circuit. Les circuits numériques sont connus pour leur structure régulière en bande dans lesquelles les cellules standards sont placées et routées sachant leurs connectivités. C'est avec cette idée de bandes que de nombreux circuits analogiques sont également organisés. Pour les circuits analogiques, ces bandes sont en revanche hétérogènes et dépendent de la hauteur des modules qui les constituent. Ces bandes ne s'étendent pas d'une extrémité à une autre du circuit mais se limitent à un groupe de modules soumis à des contraintes de proximités dû à leurs connectivités. Placer les modules de cette façon permet d'obtenir des circuits réguliers ce qui est un aspect critique pour certains circuits analogiques qui nécessitent d'évoluer dans un environnement le plus semblable possible.

Avec l'objectif de représenter l'organisation en bande, nous avons opté pour le choix de la représentation des *slicing trees*. Mentionné dans la section 1.2.1, un *slicing tree* est une représentation topologique qui consiste à définir un espace divisé par de nombreuses fois verticalement ou horizontalement. Ces divisions d'espace forment alors un ensemble de régions rectangulaires qui chacun peut représenter un module ou bien un espace vide pouvant être utilisé pour la phase de routage. Ces divisions sont organisées de manière hiérarchique et forment un graphe dans lequel chaque noeud hiérarchique est soit une division verticale, soit une division horizontale. Il en revient aux concepteurs la tâche de définir un *slicing tree* et sa définition du *slicing tree* ne sera pas altérée par notre outil de placement. Les seuls perturbations subies seront les écartements de modules afin d'y faire passer les fils de routage. En définissant eux-même la représentation topologique, les concepteurs seront libre de jugé par eux-même l'importance des contraintes de proximité, régularité ou de bordures en fonction de l'environnement dans lequel se trouve le circuit et leurs expériences personnelles en tant que concepteur.



évolution des découpes d'un slicing tree

FIGURE 1.9: Evolution des découpes d'un slicing tree

Fig. 1.9 présente une construction de graphe de *slicing trees* pour un placement donné. On peut y observer les découpes hiérarchiques horizontales et verticales. La représentation en *slicing trees* ne permet pas de représenter tous les placements possibles et un exemple de placement non représentable est donné par la figure 1.10. Dans une certaine mesure, il est possible de représenter ces placements avec un *slicing tree* mais la compaction et la surface du circuit occupées seront dégradées. Cependant, nous jugeons qu'il est plus important de prendre en compte

la faisabilité du circuit en tenant compte de la capacité à espacer les modules d'un placement donné. La représentation en *slicing tree* est adaptée à ces écartements, on considère que chaque découpe vertical ou horizontal puisse être un espace de routage qu'on agrandit en largeur pour une découpe verticale et en hauteur pour une découpe horizontale.

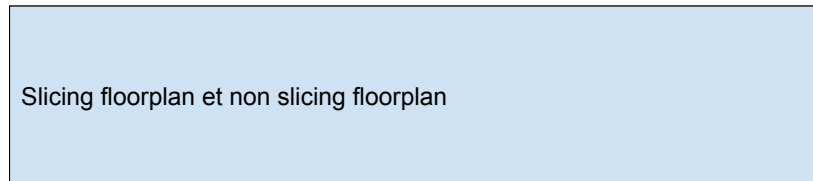


FIGURE 1.10: (a) Placement représentable par un *slicing tree* (b) Placement non représentable par un *slicing tree*

### Expression des contraintes de placement

On décompose les éléments d'un *slicing trees* de la façon suivante:

- les devices
- les espaces de routage
- Les rails traversant
- les noeuds hiérarchiques

On définit un device comme étant un bloc numérique ou analogique. Dans le cadre d'un bloc analogique, il peut s'agir d'une cellule numérique contenant une ou plusieurs cellules standards. Pour un bloc analogique, un device peut représenter un ensemble de transistors permettant de réaliser une fonction analogique de base. Ces devices peuvent être un simple **transistor**, une **paire différentielle**, un **miroir de courant**, un **montage en cascode** ou un **cross-coupled pair**. La raison pour laquelle ces blocs ont été définie est que le comportement électrique de ces blocs analogiques nécessite un dessin des masques dédié comprenant des contraintes géométriques fortes liées au placement et au routage. Ils peuvent également être générés selon des styles dessins des masques tels que par centrage géométrique ou de manière interdigité. Ces blocs analogiques sont générés de manière correcte par construction à partir des règles de dessins de la technologie donnée. De plus, chacun des devices est paramétrable et contient les estimations des parasites liés à son dessin des masques. Pour plus de détails concernant notre librairie de devices analogiques, [You12] décrit entièrement le contenu de ces devices.

Dans le contexte du **slicing tree**, un device est défini par plusieurs attributs:

- Une **coordonnée x, y**: Elle représente la position du coin inférieur gauche du device au sien du plan considéré pour le placement des devices.

- **Une hauteur et une largeur:** Elle définit le rectangle englobant la surface occupée par le device.
- **Une contrainte d'alignement:** Elle positionne le device au sein du noeud hiérarchique, d'avantage de détails seront donnés dans la description d'un noeud hiérarchique.
- **Variation du nombre de doigts des transistors:** Les configurations en nombre de doigts de transistors tolérées par le concepteur.

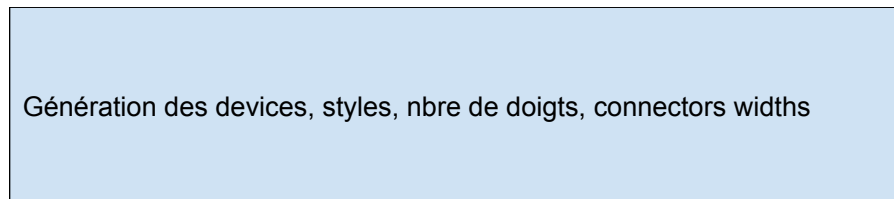


FIGURE 1.11: Les paramètres réglables des devices, (a) Style de dessin des masques, (b) Variation du nombre de doigts et (c) Variation de la taille des connecteurs.

Notre approche de placement consiste à organiser les devices en bande et on cherche également à faire sorte que les devices d'une bande possèdent une hauteur le plus similaire possible. Afin d'obtenir cette configuration, il est nécessaire de considérer plusieurs aspect ratios pour chacun des devices en faisant varier le nombre des doigts des transistors des blocs analogiques tout en préservant leur fonctionnalité électrique. Néanmoins, nous sommes conscient que cette action implique une variation de la capacité source/drain  $C_{jSB}$  pouvant influencer les performances du circuits. [DXS06] et [Wu+14] détaille cette influence de la variation du nombre de doigts d'un transistor. Cependant, dans le cadre de notre approche, notre objectif consiste à effectuer plusieurs boucles itératives sur l'ensemble du design et ainsi resserrer les contraintes du circuit à chaque nouvelle itération afin de tendre vers une solution souhaitée.

Les espaces de routage sont des espaces vides dédiés au positionnement des fils de routage. Au sein d'un **slicing tree**, chaque découpe du circuit est considéré comme un espace de routage rectangulaire redimensionnable dans une seule direction. Dans le cas d'une découpe verticale, l'espace de routage peut être redimensionné de manière horizontale et on appelle ces espaces comme étant des espaces de routage verticaux. Réciproquement pour les découpes horizontales, l'espace de routage peut être redimensionné manière verticale et on appelle ces espaces comme étant des espace de routage horizontaux. Les espace de routage représentant une découpe du **slicing tree** sont implicites et ne nécessitent pas d'être précisé par l'utilisateur. Pour des raisons de clareté, les espaces de routage dûs à une découpe du **slicing tree** ne seront pas représentés sur un **slicing tree**. En revanche, il est possible de considérer des espaces vides volontaires rectangulaires, ces derniers peuvent être utilisés pour des ajustements de position.



Dans le contexte du **slicing tree**, un espace de routage est défini par plusieurs attributs:

- **Une coordonnée  $x, y$ :** Elle représente la position du coin inférieur gauche de l'espace de routage au sein du plan considéré.
- **Une hauteur ou une largeur:** Elle définit l'épaisseur de la découpe (largeur pour une découpe verticale et hauteur pour une découpe horizontale).

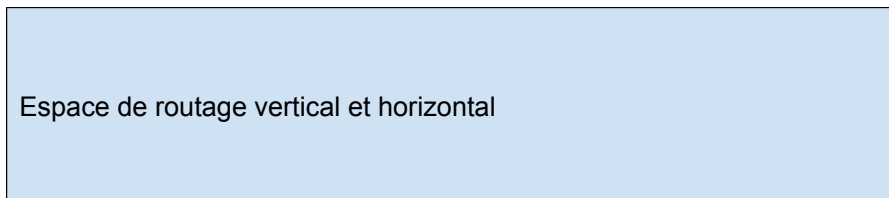


FIGURE 1.12: Espace de routage vertical et horizontal

Les noeuds hiérarchiques représentent les découpes hiérarchiques verticales et horizontales du circuit. Ces noeuds permettent la gestion des contraintes appliquées aux devices tels que les contraintes d'alignement et gèrent également les contraintes de symétries des devices. Chaque noeud hiérarchique possède des noeuds fils pouvant être des devices, des espaces de routage, des rails ou un autre noeud hiérarchique et gère la propagation des contraintes soumis par le concepteur vers les racines du **slicing tree** ainsi qu'aux espaces de routage implicites dûs aux découpes. Les noeuds hiérarchiques propagent également les aspects ratios des devices en **bottom-up** jusqu'à la racine du **slicing tree** qui représente les aspects ratios globales du circuit.

Dans le contexte du **slicing tree**, un noeud hiérarchique est défini par plusieurs attributs:

- **Une coordonnée  $x, y$ :** Elle représente la position du coin inférieur gauche de l'espace occupé par le noeud hiérarchique et ses noeuds fils.
- **Une hauteur et une largeur:** Elle définit le rectangle englobant la surface occupée par l'espace occupé par le noeud hiérarchique et ses noeuds fils.
- **Une contrainte d'alignement:** Elle positionne le noeud hiérarchique au sein du noeud hiérarchique supérieur.
- **Ensemble de noeuds fils:** L'ensemble des noeuds fils est stocké dans un ordre déterminé déterminant le positionnement.
- **Les symétries:** Les noeuds fils symétriques sont stockés au niveau du noeud parent.
- **Paramètre de validité:** La différence de taille entre device est souvent importante. Afin de se rapprocher le plus possible d'une topologie en bande, ce paramètre permet de spécifier la différence de taille autorisée pour considérer la bande comme étant un placement valide.

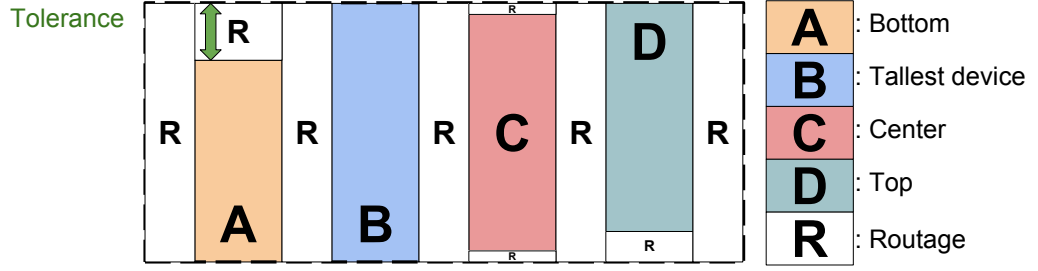


FIGURE 1.13: Noeud hiérarchique vertical avec une organisation en bande

La figure 1.13 présente l'organisation d'un noeud hiérarchique vertical. On y observe l'organisation d'une bande dans laquelle les devices  $A$ ,  $B$ ,  $C$  et  $D$  sont soumis à des contraintes d'alignement. La hauteur et la largeur d'un noeud hiérarchique vertical est déterminé par la formule suivante:

$$Largeur_{BandeVerticale} = \sum_{i=0}^n L_{NoeudFils_i} + \sum_{j=0}^{n+1} L_{EspaceDeRoutage_j} \quad (1.2)$$

$$Hauteur_{BandeVerticale} = \text{Max}\{H_{NoeudFils_0}, H_{NoeudFils_1}, \dots, H_{NoeudFils_n}\}$$

en considérant  $n$  le nombre total de noeuds fils. Chaque espace noté "R" représente un espace de routage et sont dû soit à une découpe verticale soit à une différence de taille avec le noeud fils ayant la plus grande hauteur qui est "B" dans le cas ci-dessus. La taille d'un espace de routage dépend du nombre de fils de routage verticaux et de leur positionnement au sein de l'espace, d'avantage de détails seront donnés dans le chapitre sur le routage. Concernant les contraintes d'alignement à chacun des devices, "A" est aligné vers le bas de la bande, "C" est centré par rapport à la hauteur de la bande et "D" est aligné vers le haut de la bande. Les symétries au sein d'un noeud hiérarchique se traduisent par la garantie que deux fils d'un noeud fils soient de la même taille et possède le même alignement. Sur la figure 1.13, la boîte englobante en pointillé représente l'espace du noeud hiérarchique considéré par les noeuds hiérarchiques supérieurs.

### Algorithme de placement

En considérant  $n$  noeuds pour un noeud hiérarchique vertical (fig. 1.12) avec  $n$  représentant le nombre de noeuds fils, sa hauteur et sa largeur sont calculés à partir de la formule suivante:

$$Height = \max(Height_{Node_1}, Height_{Node_2}, \dots, Height_{Node_n}) \quad (1.3a)$$

$$Width = \sum_{k=1}^n Width_{Node_k} \quad (1.3b)$$

Pour un noeud hiérarchique horizontal, on obtient sa hauteur et sa largeur avec la formule suivante:

$$Height = \sum_{k=1}^n Height_{Node_k} \quad (1.4a)$$

$$Width = \max(Width_{Node_1}, Width_{Node_2}, \dots, Width_{Node_n}) \quad (1.4b)$$

**Définition 2:**

Chaque noeud  $Node_X$  possède  $m$  paires (hauteur, largeur) représentant les tailles que le noeud peut prendre à moins que le noeud ne soit prédéfini. Lorsqu'un noeud possède des dimensions prédéfinies, il ne possède qu'une seule une paire unique (hauteur, largeur) définie par le concepteur:

```

if  $Node_X$  prédéfini: then
  |  $Node_X : [(Height_{preset}, W_{preset})]$ 
else
  |  $Node_X : [(Height_1, Width_1), (Height_2, Width_2), \dots, (Height_m, Width_m)]$ 
end

```

**Algorithm 1:** Possible sizes

L'algorithme évalue toutes les combinaisons possibles si elles peuvent être acceptées par un paramètre de tolérance pour la différence de taille maximum entre le noeud de plus petite taille et de plus grande taille. Chaque noeud sont initialisés avec leur première combinaison:

Combinaison initiale:

$$HierarchicalNode(Height_1, Width_1) = \begin{pmatrix} Node_1(Height_1, Width_1) \\ Node_2(Height_1, Width_1) \\ \vdots \\ Node_n(Height_1, Width_1) \end{pmatrix} \quad (1.5)$$

Une fois que cette combinaison de tailles est traitée, le  $Node_1$  utilise sa combinaison de taille (hauteur, largeur) suivante jusqu'à que l'ensemble des combinaisons du  $Node_1$  soit traité. Lorsque la dernière combinaison du  $Node_1$  est atteinte, le  $Node_2$  incrémente à sa combinaison suivante et  $Node_1$  repasse à sa combinaison initiale et le processus se déroule de la même façon pour les noeuds suivants: L'algorithme principal est le suivante:

**Case: Noeud Vertical**

**for**  $i = 1:\text{Nombre de combinaisons}$  **do**

Déterminer  $(Height_{max_i}, Height_{min_i}, Width_i)^{(1)}$

**if**  $(Height_{max_i} - Height_{min_i}) \leq Height\ tolerance^{(2)}$  **then**

Taille acceptée  $(Height_i, Width_i)$

**end**

**end**

**Case: Noeud Horizontal**

Algorithme identique à l'exception de (1) et (2).

(1): Déterminer  $(Width_{max_i}, Width_{min_i}, Height_i)$

(2):  $(Width_{max_i} - Width_{min_i}) \leq Width\ tolerance$

**Algorithm 2:** Main Placement Algorithm

Le nombre total de combinaisons évaluées est de:

$$Number\ of\ sets = (m_{Node_1} * m_{Node_2} * \dots * m_{Node_n}) \quad (1.6)$$

Le nombre de combinaisons à considérer augmente de manière rapide (factorielle) mais certaines considérations sont à prendre en compte. Au sein d'un **slicing tree**, il est commun d'avoir plusieurs noeuds hiérarchiques ce qui implique que certaines combinaisons seront filtrées si la condition (2) de l'algorithme principale ne respectent pas les paramètres de différence tailles. De plus, un noeud hierarchique peut avoir à gérer à des contraintes de symétries parmi ses noeuds fils. Cela signifie que pour 2 noeuds symétriques, on ne considérera que  $m$  combinaisons au lieu de  $m^2$ .

## 1.4 Exemples

What - gmchamla - Expliquer la structure de la description python - Etapes à suivre avec l'exécution des outils + screen des étapes - Interface graphique, choix des positions et parametres de selections - Différents résultats de placement à montrer

## 1.5 Conclusion

## Chapter 2

# Problem Definition

An automatic placement tool should produce analog device-level layouts similar in density and performance to the high-quality manual layouts. Having the devices for the selected topology sized, they must be laid out in the chip. Considering these devices and their interconnects, the problem is to explore non-overlap placements of given modules. The capability to deal with layout constraints, in order to eliminate unwanted parasitics due to the process variations, is mandatory. cite00 enumerates the most common placement constraints that are respected by modern analog placer. Among them, symmetries and symmetry-island cite3 are the most used constraints. Other constraints like common-centroid cite4, proximity and range are also often considered cite5. Taking into account current and signal path improves performance accuracy cite6. Devices can be placed depending on their thermal impact on the chip cite7. Regularity cite8 and boundary constraints cite9 enhance routability and suppress parasitics induced by extra bends of wires and via cost.

Given a placement of a set of layout devices together with their positions, the layout must then be interconnected according to the netlist in a design rule correct way. Due to the fact that the performances of an analog circuit are critically dependent on layout parasitics, the routing of an analog circuit requires more attention than that of a digital circuit. Analog routing constraints can be critical such as symmetry, topology and wirelength matching.

Recent research focused on using simulated annealing algorithm (SA) in combination with topological representations to respect these placement constraints. Topological representations encode the positioning relations between devices and SA optimizer alters their relative positions. The most popular representations used in analog placement over the last decades are Sequence-Pair cite10, B\*-Tree cite11, Transitive Closure Graph cite12, Ordered-Tree cite13 and slicing floorplans cite15 and they were coupled with some constraints to respect at the same time. This will be discussed in the following section



## Chapter 3

# State of the Art

Analog layout automation tools follow different approaches. Procedural generation tools are about codifying the entire circuit layout using parametric representation by the designer through a procedural language or graphical user interface. Template-based tools incorporate designer knowledge into the optimization task. This approach finds its applications restricted to small circuits or to more complex circuits with the goal of achieving the first cut design. Optimization-based tools use iterative procedure where design variables are updated at each iteration until they achieve an equilibrium point. Overall successful layout generation tools are often optimization-based.

cite20 reviews characteristics that can be taken into consideration to evaluate automation tools:

- *Accuracy and robustness*: Measure of the tool's performance prediction compared to real performance and capacity of the tool to build and test circuits tolerant to manufacturing faults and operating point variations.
- *Automation level*: Time spent to design a circuit with the tool. We can consider the running time (time spent by the tool to provide a solution) and the setup time (time spent by the designer to adequate the problem to the tool).
- *Scope of the tool*: It can be described as a group of analog design problems, which can be solved by this tool. A tool which aims at solving a wide range of design problems will be successful in the long run.
- *Design facilities*: Multi-objective optimization, interactive design, bookkeeping facilities.





## Chapter 4

# Academic Tools

Some academic tools related to analog layout generation will be reviewed in this following part.

Examples of procedural generation tools:

- ALSYN cite21 can synthesize layout from netlist-level description for a range of analog integrated circuits. Designers can incorporate their own reusable knowledge into the synthesis process to explore design space. It also provide a flexible module generator environment and easy technology database access with variable shapes.
- cite22 describes the automatic generation and reusability of physical layouts of analog and mixed-signal blocks based on high-functionality parameterized cells (pCells) that are fully independent of technologies.

Examples of template-based tools:

- LAYGEN cite24 is template based tool that includes placement and routing constraints defined by the designer following a traditionnal analog flow that can generate automatically analog integrated circuit layouts.
- BAG cite23 is a design frame for AMS circuits capable of integrating all steps of the design flow, from architectural-specification definition to correct layout implementation, into procedural analog generators.

Examples of optimization-based tools:

- ANAGRAM cite25 is a full-custom layout of analog cells which is in the style of a macro-cell place and route problem. It can handle layout symmetries, dynamic merging and abutment of individual devices, and flexible generation of wells and bulk contacts
- ILAC cite26 is process-independent tool that automatically generates geometrical layout for analog CMOS cells from a circuit description. It handles typical analog layout constraints such as device matching, symmetry and distance and coupling constraints but also supports user-specified constraints.



## Chapter 5

# Industrial Tools

Some industrial tools related to analog layout generation will be reviewed in this following part.

- Synopsys Helix (ex-Ciranova since 2012) cite27 is a device-level automated placement solution for analog, and mixed-signal IC designs such as PLLs, SerDes, ADCs and PHYs. Helix is fully hierarchical and offers capacity in the tens of thousands of transistors, enabling it to optimize floorplanning and placement for entire custom IC blocks or IP at once.
- Mentor Graphics Pyxis Layout Suite cite28 offers configured options for design engineers who need increased productivity and an automated way of handling the physical layout of their AMS designs. Bundles are offered in a standard configuration, with optional applications that provide increasing levels of automation and functionality to address the designers unique requirements.
- Synopsys Laker cite29 delivers a complete solution for analog, mixed-signal, and custom digital design and layout that is optimized for performance and interoperability for 28-nanometer (nm) and below design flows. It has a built-in understanding of analog layout requirements, so it is much easier to adopt and deploy than competing analog automation tools.
- Virtuoso Layout Suite Family cite30 comprises the layout environment of the industry-standard Virtuoso custom design platform, a complete solution for front-to back custom analog, digital, RF, and mixed-signal design. It preserves design intent throughout the entire physical implementation process, while managing multiple levels of design abstractions from device, cell, and block levels through to the full-chip level.
- Tanner EDA - High Performance Device Generation (HiPer DevGen) cite31 creates building blocks based on an understanding of the functional requirements that are needed to produce a high-quality layout. Retargeting the components to a new technology node simply requires the user to input the manufacturing rules for that technology and regenerate the devices and primitives.



## Chapter 6

# Challenges

Although most of the recent works focus on simulated annealing algorithm [100], we believe that giving more control to the designer and using his interventions to set some constraints yields good analog placement results. Our approach introduces a semi-automatic analog placement approach guided by the designer's preferences. This semi-automatic approach also helps the designer to debug more efficiently and make adjustments easier since he will, himself, control the overall relative placement of the circuit but at the same time, some tiresome and error-prone tasks are automated.

Another critical part in analog design, that has not been considered in most of the previous work, is the routing phase. Conventionally, the execution of placement and routing has been sequential. If the routing is a two-step procedure, the execution of the global routing and detailed routing is also sequential. Simultaneous performance optimization of the placement and the global routing can lead to a more accurate search and this is what we aim to do unlike most of the recent research who only consider the placement alone. The general reluctance of analog designers toward using automatic layout generation tools is because they are skeptical about their reliability and they would like to have full control over the layout generation process. They would prefer the layout tool to be an assistant to them and generating a template for the layout is sometimes better for them.



## Chapter 7

# Analog/Mixed-signal Design Approach

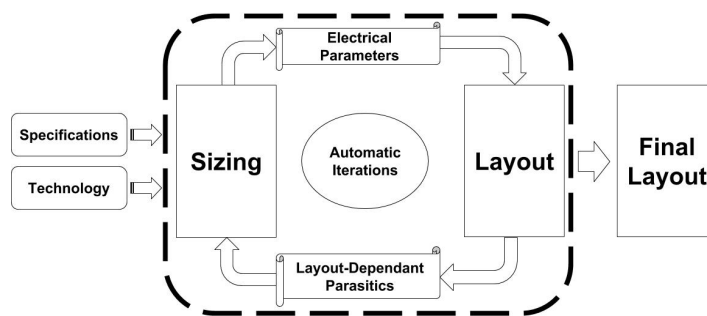


FIGURE 7.1: Cairo Hurricane AMS Design Flow

### 7.1 Design Flow

The Cairo Hurricane AMS (CHAMS) project cite002, developed by the LIP6 Laboratory, proposes a complete flow which would be able to create a library of reconfigurable analog Intellectual Properties, to automatically generate and optimize the layout with little intervention from the designer. The flow provides a reliable and efficient solution taking into account parasitic effect-aware layout generation with enough flexibility to adapt to different designers needs and concerns. CHAMS layout generation tool supports any technology with the new nanometric layout dependent parasitic parameters.

The proposed CHAMS analog design (Fig. 2) is a two way communication between the sizing and layout generation. The idea is that the sizing tool provides the electrical parameters of the transistor such as the width, length, number of fingers, etc... to the layout generation tool. The tool automatically generates the layout from a library of parameterized cells and sends back to the sizing engine the layout-dependent parasitic parameters such as the drain and source areas and perimeters, the stress effect parameters, etc... to re-evaluate the performance. This internal loop is repeated several times, with minimal designer intervention, until the target specifications are met.

Our goal is to develop a tool capable to place and route analog layout respecting designer specifications in a shorter amount of time compared to the fully manual approach. We also aim to have a tool which will be technology

independant and can as well, place and route mixed-signal circuits. At this moment, the bottleneck of our design flow is to handle analog place and route and this is the reason why this thesis is mainly focused on it.

## 7.2 Placement in row

Digital and analog circuits have a dedicated area on a system-on-chip circuit so they can be independently designed within a specific space. Digital circuits are well-known for their regular row structure where standard cells are placed and routed accordingly to their netlist. In a similar way, we plan to organize the analog circuit layout in rows of devices and the analog circuit area should be placed and routed within its dedicated area.

It is common to design analog circuit in rows of devices where the height of each row of devices should be adjustable so it can match its dedicated area. Therefore, we choose the slicing tree representation for several reasons:

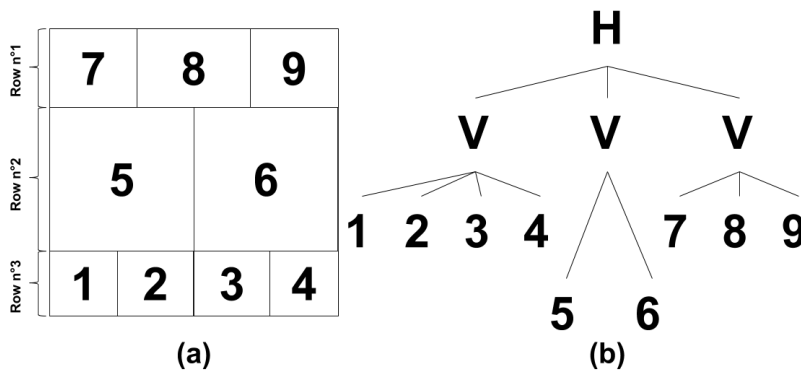
- Slicing trees are a natural choice since they are adequate to the row structure. Rows are represented by horizontal slices which will be divided into vertical slices determined by the area of each device.
- Unlike most of modern analog placement methods, our placement strategy is semi-automatic and will be guided by the designer's preferences. Slicing trees are easy to handle and let the designer choose the overall topology. That shows some advantages that will be explained in the following section.



## Chapter 8

# Methodology for the placement

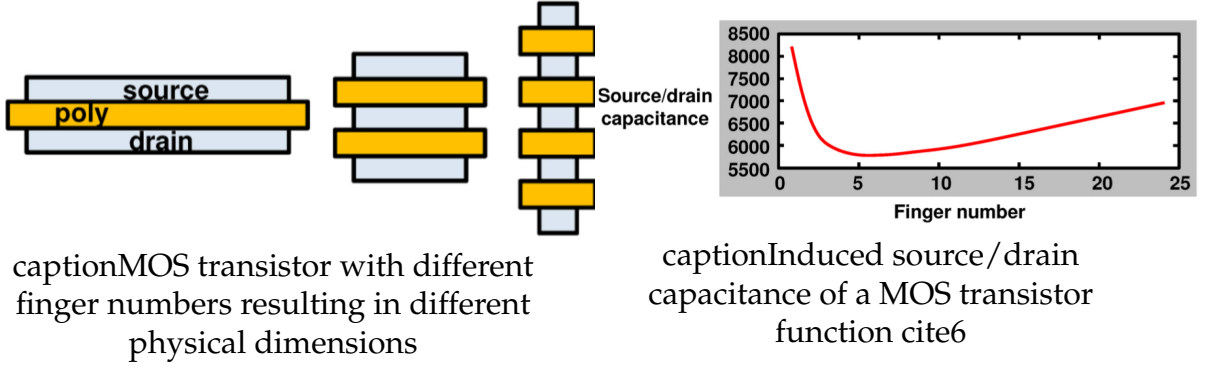
Modern analog automation tools are able to generate analog layout respecting various placement constraints and they mainly use simulated annealing approach. Nevertheless, such approach might not produce predictable and easy-to-adjust results. We believe that giving more control to the designer will generate analog layout, easier to predict compared to optimization-based approach, but also easier to adjust in case modifications are required. Our analog/mixed-signal placer is semi-automatic: the device generation is automated and generated correct-by-construction and guided by the designer's constraints over the circuit. The following sections will describe the interventions from the designer.



(a) placement (a) and its slicing tree representation (b) where "H" stands for horizontal cut and "V" for vertical cut

### 8.1 Slicing Tree Construction

In order to organize devices in row, we use the slicing tree structure (Fig. 3) which will be described by the designer and not automatically generated by the placer. Our placer will only assist the designer so it generates the slicing tree according to the designer's constraints. A slicing tree is a slicing floorplan that represents an area that has been divided multiple times either vertically or horizontally, forming a set of rectangular regions representing the place filled by each device. These slices are organized hierarchically so they form a graph where the hierarchical nodes are either horizontal or vertical cuts. Fig. 3 shows an example of a slicing tree representing a circuit of 9 devices organized in three rows.



## 8.2 Devices Variations

Our placement approach consists in organizing devices in row, that is to say to have rows of devices with similar height, and the overall analog part's height would be a multiple of standard cell's heights from the digital circuit part. To obtain a row of devices, the height of each device needs to be similar and this is the reason why, we need to consider several possible aspect ratios for each device by varying the number of fingers (Fig. 4) like in cite6 so we can find heights of devices that match a given height.

However varying the number of fingers of a MOS transistor implies a variation of the source/drain capacitance  $C_{jSB}$  which can impact the circuit performance. This variation can be calculated using the formula (8.1):

$$C_{jSB} = \frac{A.C_j}{\left(1 - \frac{V_{BS}}{\phi_j}\right)^{m_j}} + \frac{P.C_{jsw}}{\left(1 - \frac{V_{BS}}{\phi_j}\right)^{m_{jsw}}} \quad (8.1)$$

with:

- $A$  and  $P$ : source/drain Area and Perimeter
- $C_j$  and  $C_{jsw}$ : Bottom and Sidewall junction capacitances
- $\phi_j$ : built-in junction potential
- $m_j$  and  $m_{jsw}$ : depend on the doping profile of the junction

Considering a given MOS transistor, its channel width and length is decided by the sizing step and cite6 presents the following formula of  $C_{jSB}$  variation based on the number of fingers:

$$C_{jSB}(F) = (\alpha.W.L + \alpha.W.\delta_{gap} + 2.\beta.\delta_{gap}) + 2.\beta.F(L + W.\delta_{gap}) + \frac{W}{F}(\alpha.\delta_{gap} + 2.\beta) \quad (8.2)$$

with:

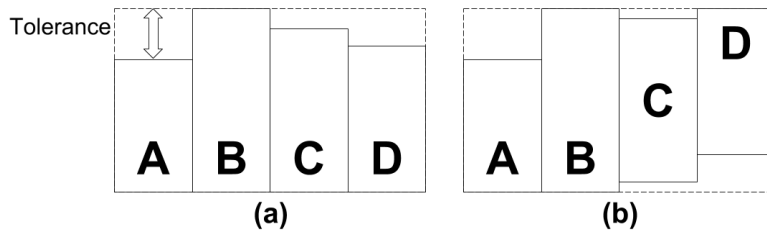
- $F$ : number of fingers
- $W$  and  $L$ : channel Width and Length
- $\alpha = \left(1 - \frac{V_{BS}}{\phi_j}\right)^{m_j}$

- $\beta = \left(1 - \frac{V_{BS}}{\phi_j}\right)^{m_{jsw}}$
- $\delta_{gap}$ : gap separating 2 fingers

As cite2 mentioned, changing the number of fingers of a device can considerably change the source/drain bulk capacitance (Fig. 5) and therefore the device properties. Nevertheless, as mentioned in section 4.1, our placement phase is part of an internal loop, which is repeated until a result meets the required performances (Fig. 2). Also, it is the task of the designer to decide the admissible range of shapes for every device which would limit the number of finger variations.

### 8.3 Margin Tolerances

We aim at creating rows and it is important to define what we consider as a row of devices. We introduce a tolerance margin, which represents the difference between the smallest and the tallest device's height in a vertical node. Fig. 6 (a) shows an example of devices organized in row where the area A, B, C and D represent devices. Devices A and B show respectively the smallest and highest height inside the row. This difference of height is compared to the tolerance parameter to establish if this is considered as a legal row. We apply the same concept to the horizontal node but instead of using the height as a comparison, we use the width. It is up to the designer to adjust this tolerance margin and it will impact on the number of accepted possibilities. Different margin tolerances can be considered at each hierarchical nodes. At the same time, having a small tolerance will reduce the waste of space induced by the slicing tree representation.



(a) and alignment (b) in slicing tree representation

### 8.4 Placement Constraints

As said in section 3.1, it is the task of the designer to describe the slicing tree with the help of our placer. This will directly impact on the topology of the circuit. This means he will have the total control over the relative relation between the blocks. Having the designer building his own slicing tree also means that he will have control over placement constraints such as proximity, boundary, current/signal path and regularity constraints based on his knowledge and preference.

Among the most common constraints, symmetries can be respected with the appropriate slicing tree organization. We also consider alignment constraints inside of a slicing tree in horizontal or vertical node. In Fig. 6 (b), we have an

example of possible alignments: devices A and B are aligned to the bottom of the row, device C is centered and device D is aligned to the top of the row. In horizontal slices, devices can be aligned to the right, center and left of the row.

## 8.5 Placement Choice

Similar to cite17, once all the possible variations are set for each device, we evaluate the accepted variations at hierarchical nodes based on the margin tolerances. These accepted variations are propagated from the bottom of the slicing tree to the root. After this bottom-up propagation, the designer processes the different placements based on height, width or global ratio criteria. He can choose the most optimized placements according to the Pareto front curve like in cite18, but he is free to choose any other possible placement that would eventually have more white space if the circuit can afford more space.

## Chapter 9

# Placement Algorithm

### 9.1 Theory

**Definition 1:**

Considering  $n$  nodes for a Vertical hierarchical node (Fig. 6) with  $n$  representing the number of children, its height and the width are calculated with the following formulas:

$$Height = \max(Height_{Node_1}, Height_{Node_2}, \dots, Height_{Node_n}) \quad (9.1a)$$

$$Width = \sum_{k=1}^n Width_{Node_k} \quad (9.1b)$$

For a Horizontal hierarchical node, we have:

$$Height = \sum_{k=1}^n Height_{Node_k} \quad (9.2a)$$

$$Width = \max(Width_{Node_1}, Width_{Node_2}, \dots, Width_{Node_n}) \quad (9.2b)$$

**Definition 2:**

Each  $Node_X$  has  $m$  pairs (height, width) representing the size that it can take, unless if the node is preset. When a node is preset, it has a single size height/width defined by the designer:

```

if  $Node_X$  is preset then
|    $Node_X : [(Height_{preset}, W_{preset})]$ 
else
|    $Node_X : [(Height_1, Width_1), (Height_2, Width_2), \dots, (Height_m, Width_m)]$ 
|   with  $m$ , number of sizes
end

```

**Algorithm 3:** Possible sizes

The algorithm evaluates all the possible sets if they can be accepted based on the tolerance margins parameter for the height and the width of the hierarchical node. Once this set of pairs height/width sizes is processed,  $Node_1$  considers then its second size  $(Height_2, Width_2)$  until all its possible sizes are considered. Once the last size is reached for  $Node_1$ ,  $Node_2$  passes to its second size and  $Node_1$  back to its first possible size and so on. The main algorithm is the following one:

Initial set:

$$HierarchicalNode(Height_1, Width_1) = \begin{pmatrix} Node_1(Height_1, Width_1) \\ Node_2(Height_1, Width_1) \\ \vdots \\ Node_n(Height_1, Width_1) \end{pmatrix} \quad (9.3)$$

**Case: Vertical Node**

**for**  $i = 1: \text{Number of sets}$  **do**

Determine  $(Height_{max_i}, Height_{min_i}, Width_i)^{(1)}$

**if**  $(Height_{max_i} - Height_{min_i}) \leq Height\ tolerance^{(2)}$  **then**

Accept size  $(Height_i, Width_i)$

**end**

**end**

**Case: Horizontal Node**

Same algorithm except for (1) and (2).

(1): Determine  $(Width_{max_i}, Width_{min_i}, Height_i)$

(2):  $(Width_{max_i} - Width_{min_i}) \leq Width\ tolerance$

#### Algorithm 4: Main Placement Algorithm

The total number of sets to be evaluated is:

$$Number\ of\ sets = (m_{Node_1} * m_{Node_2} * \dots * m_{Node_n}) \quad (9.4)$$

The number of sets to consider may seem to grow at a fast rate but some considerations need to be taken into account. Inside of a slicing tree, it is very common to have several hierarchical nodes which implies that some sets will be filtered by the condition (2) of the main placement algorithm (Algorithm 2) for not respecting the margin tolerances. Moreover, a hierarchical node can contain symmetries between its children nodes. It means that for 2 symmetrical nodes, only  $m$  possible sizes will be considered instead of  $m^2$ .

## 9.2 Results

Our placement algorithm was implemented in C++ programming language on a Intel(R) Core(TM) i5-4590S CPU @ 3.00 GHz workstation with 6 GB RAM. To illustrate the capability of our algorithm, we experiment it on a fully differential transconductor cite19, designed under a technology CMOS 130 nm.

The fully differential transconductor is composed of a total of 32 devices and we consider 2 possible variations for each device. The slicing tree takes into account 11 symmetries for this circuit and tolerance margins are set in a way to have reasonable amount of accepted possibilities. Our algorithm found 384

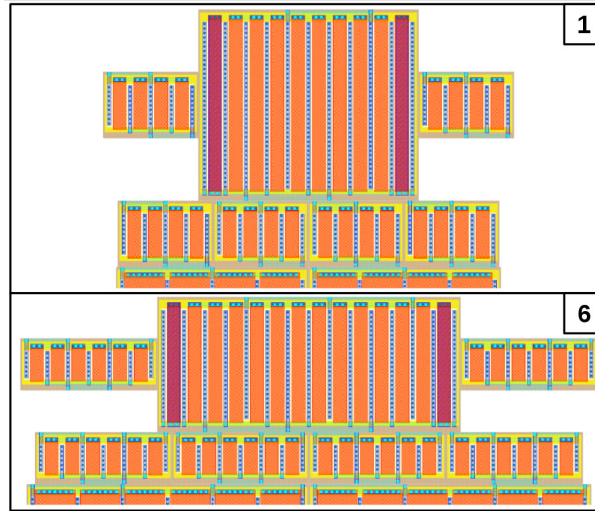


FIGURE 9.1: Evolution of 2 rows from layout 1 and 6 of Table II for different global aspect ratios. These figures have the same scale.

possible placements in less than 1 second, some of them are listed on Table II and can be seen on Fig.8.

Table II shows the characteristics of those layouts with their total area, their width, their height and the percentage of the circuit occupation in the total area. Fig. 7 and Fig. 8 illustrate the layouts described in Table II and show the evolution of the rows for different global aspect ratios. The designer can choose his final placement based on his experience and preference. The placement results are plotted on a graph with heights and widths as axis and can be selected interactively to be placed in a few seconds.

TABLE 9.1: Some layout results of the fully differential transconductor

Layout	Area ( $\mu m^2$ )	Width ( $\mu m$ )	Height ( $\mu m$ )	Occupation (%)
1	4600	84	54	68
2	5116	91	57	63
3	5603	94.2	59	58
4	6162	105	59	52
5	6648	109	61	49
6	7105	107	67	46

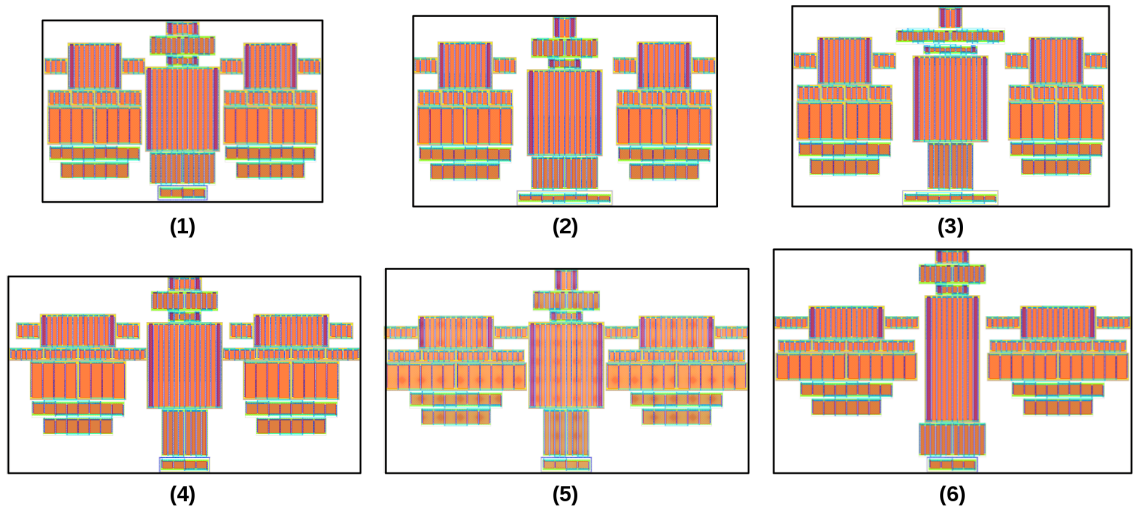


FIGURE 9.2: Layouts of the fully differential transconductor cite19 from Table II. These figures have all the same scale.



## Chapter 10

# Future Work: Routing

Once the placement phase is performed with the slicing tree, it is then used for the routing phase. For our analog routing approach, we will divide this step into two phases: a global routing phase and a detailed routing phase. To be able to route mixed-signal circuit, it is required to build a router capable of handling both digital and analog routing constraints. At the moment, our router can handle digital constraints. Considering analog constraints for the routing phase will be part of the future work of this thesis:

### 10.1 Global Routing

The global routing phase consists in giving a general idea of where the wires are going through. To do so, the slicing tree is going to be converted into a graph of relation between rectangular areas which represent a device or a routing channel. The goal is to establish by which areas each wire is going to pass by.

The search of the shortest paths are performed using Dijkstra algorithm. It allows to find the shortest point from a given source to all points by propagation through a graph (Fig. 9) where the distance relation between areas of the circuit are described. Knowing where the wires are going through, we will separate devices in order to leave enough space for wires to be placed by the detailed routing phase.

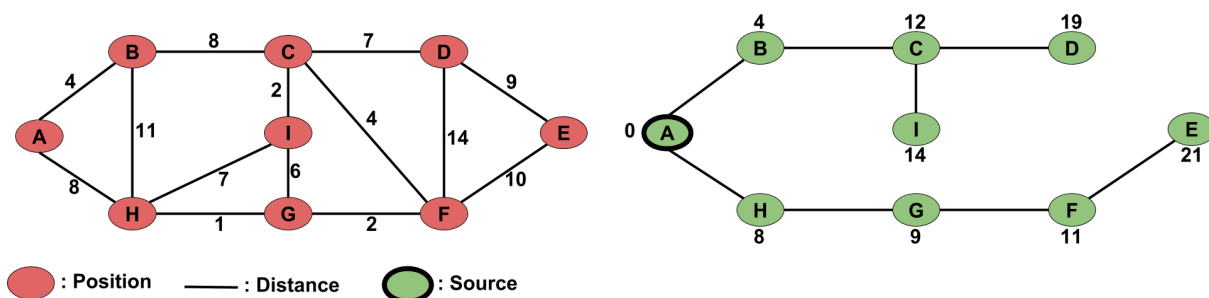


FIGURE 10.1: Dijkstra pathfinding algorithm

#### 10.1.1 Detailed Routing

The goal of detailed routing is to assign route segments of signal nets to specific routing tracks, vias, and metal layers with given global routes of those nets. For

high performance analog circuits, constraints and objectives have to be considered. For the differential signal paths, the matching between parasitics on symmetrical nodes is often more important than their absolute values. For a given set of circuit specifications, several valid routing solutions can be found but with different yield, manufacturability and testability performances. Topology-matching and current-path constraints are commonly imposed on the critical, yet asymmetry nets with the same number of bends, vias, and wirelength to reduce current mismatches, sensitive nodes and noisy nodes that should not interact

## Appendix A

# Frequently Asked Questions

### A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.



# Bibliography

- [AJ96] T. Abthoff and F. Johannes. "TINA : Analog Placement Using Enumerative Techniques Capable of Optimizing Both Area and Net Length". In: *EURO-DAC '96 with EURO-VHDL '96*. 1996, pp. 398–403.
- [Bal00] Florin Balasa. "Modeling non-slicing floorplans with binary trees". In: *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*. 2000, pp. 13–16.
- [BL00] F. Balasa and K. Lampaert. "Symmetry within the sequence-pair representation in the context of placement for analog design". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 19 (2000), pp. 721–731.
- [BM01] Florin Balasa and Sarat C Maruvada. "Using Non-slicing Topological Representations for Analog Placement". In: 84.11 (2001), pp. 2785–2792.
- [BMK02] Florin Balasa, Sarat C Maruvada, and Karthik Krishnamoorthy. "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints". In: *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. 2002, pp. 497–502.
- [Cha+00] Yun-Chih Chang et al. "B\*-Trees: a new representation for non-slicing floorplans". In: *Proceedings of the 37th Annual Design Automation Conference*. 2000, pp. 458–463.
- [COC11] Pang-Yen Chou, Hung-Chih Ou, and Yao-Wen Chang. "Heterogeneous B\*-trees for analog placement with symmetry and regularity considerations". In: *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press. 2011, pp. 512–516.
- [DXS06] Long Di, Hong Xianlong, and Dong Sheqin. "Signal-Path Driven Partition and Placement for Analog Circuit". In: *Design Automation, 2006. Asia and South Pacific Conference on*. 2006, 6–pp.
- [GCY99] Pei-Ning Guo, Chung-Kuan Cheng, and Takeshi Yoshimura. "An O-tree representation of non-slicing floorplan and its applications". In: *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*. ACM. 1999, pp. 268–273.
- [JJ83] D.W. Jepsen and C.D. Gellat Jr. "Macro placement by Monte Carlo annealing". In: *in Proc. IEEE Int. Conf. on Comp. Design*. 1983, pp. 495–498.
- [KGV+83] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

- [LC01] Jai-Ming Lin and Yao-Wen Chang. "TCG: a transitive closure graph-based representation for non-slicing floorplans". In: *Proceedings of the 38th annual Design Automation Conference*. ACM. 2001, pp. 764–769.
- [LC04] JM Lin and YW Chang. "TCG-S: orthogonal coupling of P\*-admissible representation with worst case linear-time packing scheme". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23.6 (2004), pp. 968–980.
- [LCL09] Po-Hung Lin, Yao-Wen Chang, and Shyh-Chang Lin. "Analog placement based on symmetry-island formulation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.6 (2009), pp. 791–804.
- [Lin+05] Jai-Ming Lin et al. "Placement with symmetry constraints for analog layout design using TCG-S". In: *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. ACM. 2005, pp. 1135–1137.
- [Lin+10] Cheng-Wu Lin et al. "Performance-driven analog placement considering boundary constraint". In: *Proceedings of the 47th Design Automation Conference*. ACM. 2010, pp. 292–297.
- [Lin+11] Mark Po-Hung Lin et al. "Thermal-Driven Analog Placement Considering Device Matching". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.3 (2011), pp. 325–336.
- [Lin+12] Mark Po-Hung Lin et al. "Augmenting slicing trees for analog placement". In: *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on*. IEEE. 2012, pp. 57–60.
- [Mar+05] Sarat C Maruvada et al. "Deterministic skip lists in analog topological placement". In: *ASIC, 2005. ASICON 2005. 6th International Conference On*. Vol. 2. IEEE. 2005, pp. 834–837.
- [Mur+96] Hiroshi Murata et al. "VLSI module placement based on rectangle-packing by the sequence-pair". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.12 (1996), pp. 1518–1524.
- [Pan+00] Yingxin Pang et al. "Block placement with symmetry constraints based on the O-tree non-slicing representation". In: *Proceedings of the 37th Annual Design Automation Conference*. ACM. 2000, pp. 464–467.
- [Pri+97] Juan A. Prieto et al. "A Performance-Driven Placement Algorithm with Simultaneous Place&Route Optimization and for Analog and IC's". In: *Proceedings of the 1997 European conference on Design and Test*. IEEE, 1997, p. 389.
- [Shi+10] Nakatake Shigetoshi et al. "Regularity-Oriented Analog Placement with Diffusion Sharing and Well Island Generation". In: *ASPDAC '10 Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE, 2010, pp. 305–311.

- [Str+08] Martin Strasser et al. "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions". In: *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press. 2008, pp. 306–313.
- [Tsa+11] Hui-Fang Tsao et al. "A corner stitching compliant B\*-tree representation and its applications to analog placement". In: *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*. IEEE. 2011, pp. 507–511.
- [TYC06] Yiu-Cheong Tam, Evangeline FY Young, and Chris Chu. "Analog Placement with Symmetry and Other Placement Constraints". In: *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*. 2006, pp. 349–354.
- [WL86] D. F. Wong and C. L. Liu. "A New Algorithm for Floorplan Design". In: *DAC '86 Proceedings of the 23rd ACM/IEEE Design Automation Conference*. IEEE, 1986, pp. 101–107.
- [Wu+12] Po-Hsun Wu et al. "Performance-driven analog placement considering monotonic current paths". In: *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*. IEEE, 2012, pp. 613–619.
- [Wu+14] Po-Hsun Wu et al. "Exploring Feasibilities of Symmetry Islands and Monotonic Current Paths in Slicing Trees for Analog Placement". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.6 (2014), 879–892.
- [Xia+10] Linfu Xiao et al. "Practical placement and routing techniques for analog circuit designs". In: *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*. 2010, pp. 675–679.
- [XY09] Linfu Xiao and Evangeline FY Young. "Analog placement with common centroid and 1-D symmetry constraints". In: *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*. IEEE. 2009, pp. 353–360.
- [You12] Stéphanie Youssef. "Aide au concepteur pour la génération de masques analogiques, réutilisables et optimisés, en technologie CMOS nanométrique". PhD thesis. Université Pierre et Marie Curie-Paris 6, 2012.
- [YW98] F.Y. Young and D.F. Wong. "Slicing floorplans with pre-placed modules". In: *Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on*. 1998, pp. 252–258.
- [YWY00] Fung Yu Young, DF Wong, and Hannah Honghua Yang. "Slicing Floorplans with Range Constraint". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19.2 (2000), pp. 272–278.
- [YWY99] Fung Yu Young, DF Wong, and Hannah Honghua Yang. "Slicing Floorplans and with Boundary and Constraints". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18.9 (1999), pp. 1385–1389.

- [ZSJ08] Lihong Zhang, C-J Richard Shi, and Yingtao Jiang. “Symmetry-aware placement with transitive closure graphs for analog layout design”. In: *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*. IEEE. 2008, pp. 180–185.