UNE APPROCHE INTÉGRÉE POUR LA SYNTHÈSE-PLACEMENT-ROUTAGE DES SYSTÈMES SUR PUCE

Christophe ALEXANDRE Alain GREINER

Département ASIM, laboratoire LIP6, Université Pierre et Marie CURIE
4, place Jussieu, 75252 Paris Cedex 05
{Christophe.Alexandre, Alain.Greiner}@lip6.fr

RÉSUMÉ : Dans cet article, nous présentons les principales motivations qui poussent à réviser le flot de conception classique des circuits intégrés : synthèse-placement-routage. Au fil des années, pour synthétiser, placer et router un design, des solutions algorithmiques performantes ont été développées. Les problèmes nouveaux posés par les procédés de fabrication profondément sub-micronique (résistances des fils d'interconnexion, capacités de couplage, etc.) imposent de modifier l'articulation entre ces étapes.

I – INTRODUCTION

Avec le développement des procédés de fabrication sub-micronique, les temps de propagation des signaux dans les systèmes intégrés sur puce sont de plus en plus largement dominés par les interconnexions : les temps liés à la résistance et capacité distribuées des fils d'interconnexion sont plus importants que les temps de traversée des portes.

Par conséquent, l'utilisation de modèles de charge des interconnexions (Wireload Model : WLM) devient inopérante lors de l'étape de synthèse [2]. Ces modèles, basés sur des estimations statistiques de la capacité des fils d'interconnexions, ne prennent pas en compte la topologie des fils, puisque celle-ci ne peut être connue qu'après le placement et le routage. L'écart entre la charge prédite et celle finalement mesurée est donc beaucoup trop grand. Sur l'exemple de la figure 1, pour un même signal logique mais pour des placements différents, la synthèse classique ne fait pas de différence entre les deux configurations.

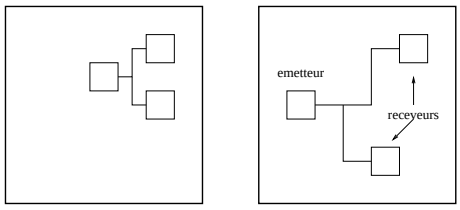


FIG. 1 – deux configurations de placement

Certains outils de CAO essaient de résoudre cette difficulté en prenant en compte le placement des cellules dans la phase de synthèse. Ceci entraîne des allers retours laborieux entre les outils de placement/routage, qui font partie du *back-end*, et les outils de synthèse qui font plutôt partie du *front-end*. Surtout, la connaissance du placement ne suffit pas à évaluer avec précision les temps passés dans les interconnexions, car ces temps dépendent à la fois des capacités ET des résistances distribuées. S'il est possible d'évaluer avec une précision acceptable les capacités des fils à partir de la seule information de placement, l'évaluation des résistances implique la connaissance de la topologie des fils d'interconnexion, et suppose donc qu'on a déjà effectué au moins le routage global.

II – FLOT DE CONCEPTION

Le flot de conception VLSI classique est traditionnellement divisé entre une partie *front-end*, qui se déroule souvent chez l'industriel concepteur du système, et une partie *back-end*, qui elle se déroule souvent chez le fondeur qui réalise la fabrication du circuit. Dans le *front-end*, le point de départ est généralement une description comportementale de niveau RTL, on y trouve ensuite les phases de validation (par simulation ou par des méthodes formelles), de synthèse vers la bibliothèque de cellules proposée par le fondeur, et de génération des vecteurs de tests. Dans le *back-end*, on trouve les phases de placement, routage, extraction et analyse des performances temporelles et des contraintes d'intégrité *électrique*.
Si les contraintes temporelles ou électriques ne sont pas respectées, il faut remonter très haut dans le flot (voir figure 2).
SI on souhaite supprimer ces itérations, il faut résoudre les difficultés suivantes :

1. Il faut disposer dans la phase de synthèse des informations de placement et de routage global afin d'évaluer avec précision les temps passés dans les interconnexions pour pouvoir guider efficacement la synthèse.
2. Dans la phase de placement, pour résoudre les problèmes de chaîne longue, il faut pouvoir insérer des amplificateurs, voire même resynthétiser certaines parties de la net-list.
3. Pendant le routage, pour résoudre les problèmes de sur-densité et de diaphonie, il faut pouvoir déplacer certaines cellules et donc remettre en cause le placement de certaines régions.

Ces besoins imposent de remettre profondément en cause le flot de conception descendant pour permettre de combiner les différents traitements (voir figure 3).
Par contre, les algorithmes mis en œuvre dans chacune des étapes de traitement ne sont pas forcément remis en cause. En effet, actuellement, il existe des solutions algorithmiques optimales pour chaque étage du flot de conception. Il faut maintenant définir une méthodologie et une structure de données permettant aux différents outils d'interagir et de communiquer les uns avec les autres.

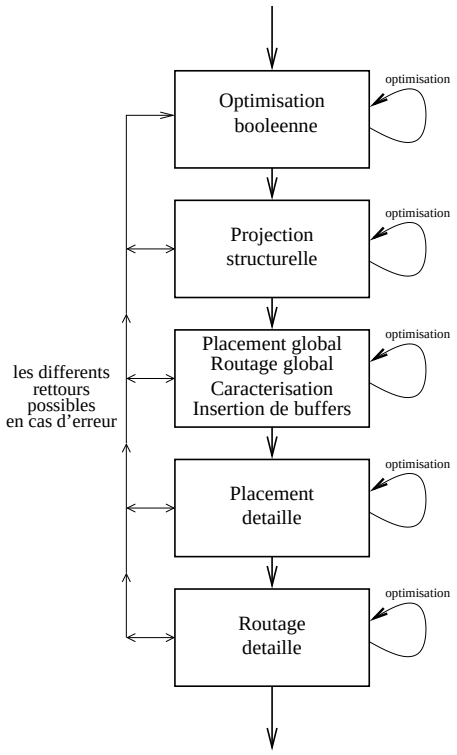


FIG. 2 – flot classique

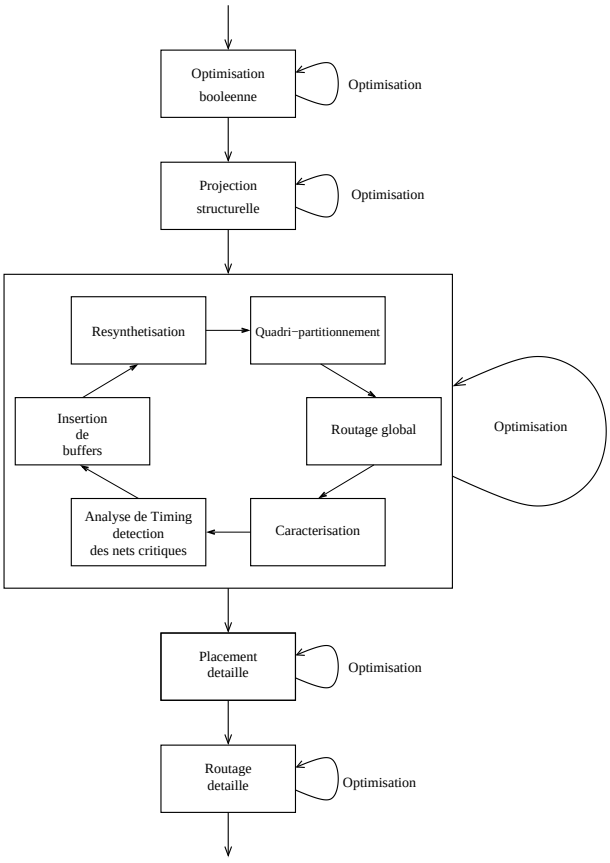
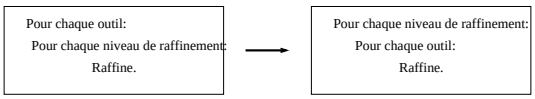


FIG. 3 – Nouveau flot

III – APPROCHE INTÉGRÉE SYNTHÈSE/PLACEMENT/ROUTAGE

La méthode générale proposée repose sur une technique de quadri-partitionnement récursif, telle que la boucle de quadri-partitionnement englobe toutes les étapes de placement, routage global, caractérisation, analyse des chaînes longues, détermination des signaux critiques, insertion d'amplificateurs (et éventuellement resynthèse). Le quadri-partitionnement peut être vu comme une méthode de raffinement progressif.
Dans les flots de conception traditionnel, les différents traitements (synthèse / placement / routage) sont exécutés de façon séquentielle. Chaque étape est réalisée par un outil séparé qui contient généralement une boucle d'optimisation. La solution «optimale» pour une étape donnée est souvent obtenue par raffinements successifs au sein de cette étape. Nous proposons donc d'inverser l'ordre d'imbrication des deux boucles : le raffinement n'est plus «interne» à chaque outil, mais englobe les 6 étapes de traitement décrites ci-dessus.



Comme on peut le voir sur la figure 4, la synthèse RTL fournit un point de départ pour le processus d'optimisation. Ensuite, les différents étapes sont les suivantes :

1. A chaque itération de la boucle, on effectue un quadri-partitionnement (basé sur une variante de l'algorithme de Fiduccia-Mattheyses [1]. A chaque itération, le nombre de blocs est multiplié par 4, et la surface de chaque zone est divisée par 4.
2. A chaque itération, on utilise le raffinement du partitionnement pour raffiner le routage global, en cherchant à contrôler et à éviter les zones de sur-densité de signaux.
3. A chaque itération, on utilise les informations (de plus en plus précises) concernant le placement et le routage pour caractériser les signaux, en calculant les temps associés aux RC distribués.
4. A chaque itération, on effectue une analyse statique des chaînes longues, en prenant en compte non seulement les délais associés aux temps de traversée des portes, mais également les délais associés aux interconnexions calculés dans la phase précédente.
5. A chaque itération, on exploite les résultats de l'analyse des chaînes longues pour déterminer quels sont les signaux critiques.
6. A chaque itération, on s'autorise à modifier la *net-list*, en insérant des amplificateurs là où cela est nécessaire, ou même en effectuant une resynthèse locale.

Le circuit est partitionné récursivement jusqu'à obtention de blocs contenant une centaine de cellules.

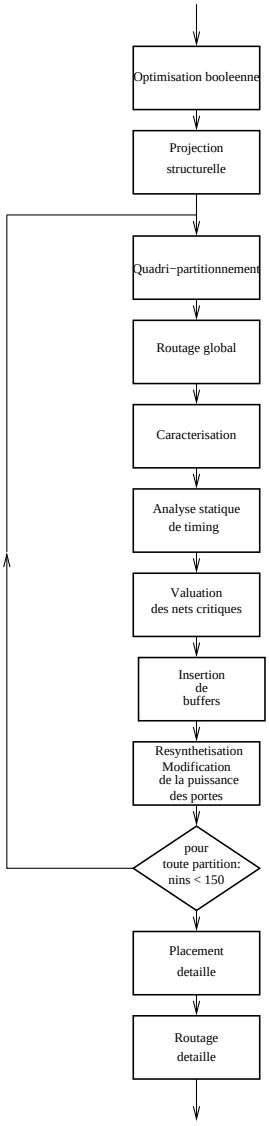


FIG. 4 – détail du flot

On effectue ensuite une phase de placement détaillé qui porte sur des petits nombres de cellules. On utilisera pour cela des techniques de recuit simulé. Pour finir, on utilise le fenêtrage déterminé par le partitionnement et le routage global pour effectuer le routage détaillé.

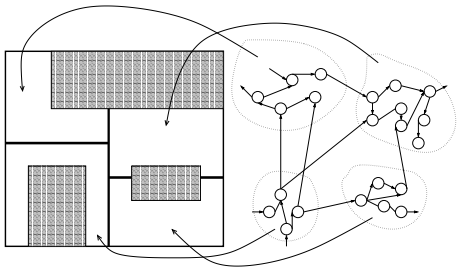


FIG. 5 – premières coupes

On quadri-partitionne donc la *net-list*, en affectant des zones physiques à chaque sous-bloc. Lors de la première itération, les attractions exercées par les entrées-sorties primaires décident de la zone qui va être affectée à chacun des 4 sous-blocs. Il faut estimer assez finement la surface qu'occupera chacune des sous-partitions une fois la synthèse complètement effectuée. Pour permettre l'insertion d'amplificateurs ou la resynthèse, il faut prévoir de la marge, en allouant à chaque sous-bloc une surface supérieure à la somme des surfaces des cellules qu'il contient. Cette marge est un paramètre ajustable, qui peut varier suivant les circuits.

Dans le cas où les modifications apportées changent de façon trop importante l'état d'une sous partition, par exemple si la surface allouée n'est plus suffisante, on peut soit déplacer l'emplacement de la cloison du partitionnement soit relancer le partitionnement. Là encore, le degré de liberté est fourni par la marge. On doit donc disposer de la remise en cause de la position des coupes.

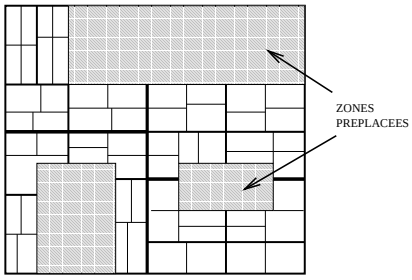


FIG. 6 – après deux étapes de quadri-partitionnement

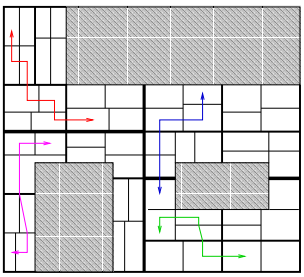


FIG. 7 – routage global

La mise en œuvre de cette approche intégrée suppose l'existence d'une structure de données unifiée, capable de représenter les différentes «vues» du circuit (vue physique, vue structurelle, vue comportementale), et partagée par les différents outils qui interviennent dans la boucle de raffinement. Cette structure de donnée doit permettre de représenter des états «intermédiaires» du circuit, (circuit partiellement placé, ou partiellement routé, et fournir également des mécanismes efficaces de manipulation de la hiérarchie. Les structures de données de la chaîne Alliance [3] ne sont pas adaptées à ces contraintes, et une nouvelle structure de données est en cours de définition au laboratoire LIP6.

IV – CONCLUSION

La prise en compte des contraintes imposées par les procédés de fabrication profondément submicronique oblige à remettre en cause la séparation traditionnelle entre le flot de conception *back-end* et le flot de conception *front-end*. Les solutions proposées par les chaînes de CAO commerciales pour intégrer la synthèse, le placement et le routage ne remettent pas réellement en cause le flot séquentiel entre les différents traitements. Nous proposons donc une approche complètement différente, où la boucle de raffinement progressif par quadri-partitionnement récursif englobe toutes les étapes de placement, routage, analyse temporelle, insertion de buffers et resynthèse. Les implications d'une telle approche sont en cours d'évaluation.

RÉFÉRENCES

[1] A. E. Caldwell, A. B. Kahng, and I. L. Markov *Improved Algorithms for Hypergraph Bipartitioning.* Proc. Asia and South Pacific Design Automation Conf., Jan. 2000, pp. 661-666.

[2] L. Scheffer and E. Nequist *Why Interconnect Prediction Doesn't Work.* ACM Intl. Workshop on System-Level Interconnect Prediction, April 2000, pp. 139-144.

[3] Frédéric Pétrot. *Outils d'aide au développement de bibliothèques VLSI portables.* PhD dissertation, UPMC, July 1994. MASI TH94.06