系統晶片設計平面規劃之研究
Floorplanning for System-on-a-chip Design

計畫編號：NSC 89-2215-E-009-117
執行期間：89年8月1號至90年7月31日
主持人：張耀文副教授 交通大學資訊科學系

一、英文摘要

Due to the increasing complexity in SOC design, hierarchical design and IP modules are widely used. This trend makes module floorplanning/placement much more critical to the quality of a circuit design. In this project, we develop efficient, flexible, and effective representations for floorplan designs and explore their applications to the placement of hard, soft, preplaced, rectilinear, and symmetry modules. In deep submicron technology, communication between different blocks has significantly increased, and interconnect delay has become the dominant factor in total circuit delay. Therefore, in this project, we also combine interconnect planning with floorplanning.

二、中文摘要

由於 SOC 設計複雜度的與日俱增，階層化設計與 IP 模組已被廣泛地使用。此趨勢使得模組的平面規劃/置放對電路設計品質的影響變得更為重要。在本計畫中，我們發展有效、具彈性，且能處理硬、軟、預先置放、 任意直線及對稱模組的平面表示法。而在深次微米的技術下，模組間訊號連通量正大幅地增加，且連線延遲變成決定整體電路延遲的最主要因素；因此，本計畫亦探討連線與平面的整合規劃。

三、背景和目的

1. Background

Due to the growth in design complexity, the circuit size is getting larger. To cope with the increasing design complexity, hierarchical design and IP modules are widely used. The trend makes module floorplanning/placement much more critical to the quality of a design.

A fundamental problem to floorplanning/placement lies in the representation of geometric relationship among modules. The representation profoundly affects the operations of modules and the complexity of floorplan/placement design process. It is thus desired to find an efficient, flexible, and effective representation of geometric relationship for the floorplan/placement design.

- **Efficiency**: The representation should be easy for implementation, and its corresponding primitive operations such as insertion, deletion, and search should be cheap. Further, the encoding cost for the representation and transformation time between the representation and floorplan/placement should be minimal.
- **Flexibility**: The representation should be able to handle all types of modules such as hard, soft, preplaced, rectilinear, and symmetry modules and consider timing information. In deep sub-micron technology, in particular, the blocks are often not rectangular, and their shapes are not fixed. Most existing floorplanning/placement algorithms only deal with rectangles and cannot apply to arbitrary shaped rectilinear placement directly. New approaches that can handle arbitrary shaped blocks are essential to optimize area utilization.
- **Effectiveness**: The representation should lead to good area utilization and timing performance.

2. Related Research

Floorplans can be divided into two categories, the *slicing structure* [10, 14] and the *non-slicing structure* [1, 5, 8, 13]. A slicing structure can be represented by a binary tree whose leaves denote modules, and internal nodes specify horizontal or vertical cut lines. Wong and Liu proposed an algorithm for slicing floorplan design [14]. They presented a normalized Polish expression to represent a slicing structure, enabling the speed-up of the search procedure. However, this representation cannot handle non-slicing floorplans. Recently, researchers have proposed several representations for non-slicing floorplans, such as sequence pair [5], bounded slicing grid (BSG) [8], and O-tree [1].

Murata et al. in [5] proposed the sequence pair representation for rectangular module placement. The main idea is to use a sequence pair to represent the geometric relation of modules, place the modules on a grid structure, and construct corresponding constraint graphs to evaluate cost. This representation requires $2n\lceil \lg n \rceil$ space to encode a sequence pair, where n is the number of modules. Further, the transformation between a sequence pair and a placement takes $O(n \lg n)$ time. Nakatake el al. in [8] presented a grid based representation---BSG. The BSG structure utilizes a set of horizontal and vertical bounded-length lines to cut the plane into rooms and represents a placement by these lines and rooms. The complexity of BSG is similar to that of the sequence pair. The sequence pair and BSG were then extended to handle pre-placed modules and soft modules in [2, 6, 5, 9].

Guo et al. in [1] proposed a tree-based representation, called *O-trees*. The transformation between the representation and a floorplan takes only $O(n)$ time. However, the tree structure is irregular. Thus, this representation takes more time in tree operations, such as search and insertion. To reduce the operation complexity, the tree is encoded by a sequence of $2n$ bits and a

permutation of $n\lceil \lg n \rceil$ bits. (Note that O-trees are the fastest representation for non-slicing floorplans in the literature.)

For rectilinear block floorplanning/placement, Preas et al. in [12] proposed a graph model for the topological relationship among rectangular and arbitrarily shaped blocks. Wong and Liu in [15] extended the Polish expression to represent slicing floorplans with rectangular and L-shaped blocks. Lee in [4] extended the zone refinement technique to rectilinear blocks. A bounded 2D contour searching algorithm is proposed to find the best position for the block. Kang and Dai in [2] proposed a BSG-based method to solve the packing of rectangular, L-shaped, T-shaped, and soft blocks. The algorithm combines simulated annealing and a genetic algorithm for general non-slicing floorplans. Xu, Guo, and Cheng in [17] presented an approach extending the sequence-pair approach for rectangular block placement to arbitrarily sized and shaped rectilinear blocks. The properties of L-shaped blocks are examined first, and then arbitrarily shaped rectilinear blocks are decomposed into a set of L-shaped blocks. Kang and Dai in [3] proposed a method based on the sequence-pair structure for the rectilinear block placement. Three necessary and sufficient conditions for a sequence pair to be feasible are derived. A stochastic search is applied on the optimization of convex block floorplanning.

3. Objective

In this project, we intend to develop efficient, flexible, and effective representations for floorplan designs and explore their applications to the placement of hard, soft, preplaced, rectilinear, and symmetry modules. In deep submicron technology, communication between different blocks has significantly increased, and interconnect delay has become the dominant factor in total circuit delay. Therefore, in this project, we also combine interconnect planning with floorplanning.

To handle general (slicing as well as non-slicing) floorplans, we proposed at 2000 Design Automation Conference (DAC-2K) a new ordered binary-tree based representation, called *B*-trees*. Given an *admissible placement* [1], we can represent it by a unique *horizontal* and a unique *vertical* B*-trees. (See Figure 1(b) for the horizontal B*-tree for the placement shown in Figure 1(a).) The admissible placement here means that it is compacted and can neither move down nor move left (see Fig. 1(a) for an admissible placement).

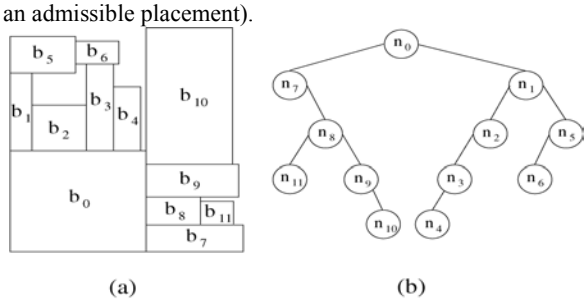

Figure 1: (a) An admissible placement. (b) The B*-tree representing the placement.

四、研究方法

We shall discuss the formulation, underlying techniques and approaches, procedures, and potential challenges and solutions for handling the floorplanning problems.

1. Formulation

Let $B=\{b_1, b_2,..., b_n\}$ be a set of n rectangular modules, and w_i, h_i, and a_i the width, height, and area of b_i, $1 \le i \le n$. The aspect ratio of b_i is given by h_i / w_i. We denote r_{min} and r_{max} as the minimum and maximum aspect ratios, i.e., $h_i / w_i \in [r_{min}, r_{max}]$. A placement $P=\{(x_i,y_i) | 1 \le i \le n\}$ is an assignment of the rectangular modules b_i's with the coordinates of their bottom-left corner. Being assigned to (x_i, y_i) so that no two modules overlap. We consider in this paper three kinds of modules: *hard modules*, *pre-placed modules*, *soft modules* and *rectilinear modules*. A hard module is not flexible in its shape but free to rotate. A pre-placed module is inflexible in both its shape and coordinate. It has to be located at a fixed position. A soft module is free to move and change its shape within the range $[r_{min}, r_{max}]$.

A rectilinear block can be represented by four profiles, called *the top profile sequence*, *the bottom profile sequence*, *the left profile sequence*, and *the right profile sequence*, specifying the profiles viewed from the top side, the bottom side, the left side, and the right side of the block, respectively. The top (bottom) profile sequence of a rectilinear block uses the leftmost horizontal segment on the top (bottom) boundary of the block as a *base* and records the length of the succeeding horizontal segments on the top (bottom) boundary and the relative height. Specifically, the top profile sequence consists of the length of the base followed by a sequence of two-tuples composed of the lengths of the succeeding horizontal segments and their relative heights to the base (could be negative). For example, Figure 2 shows a rectilinear block with the top profile sequence (4, [5, 7], [7, 4], [6, -1], [8, 4]). The base of the sequence is segment *a* which has the length of 4 units. The second horizontal segment is *c* which has the length of 5 units and is 7 units higher than the base *a*. The other three profile sequences are similarly defined.

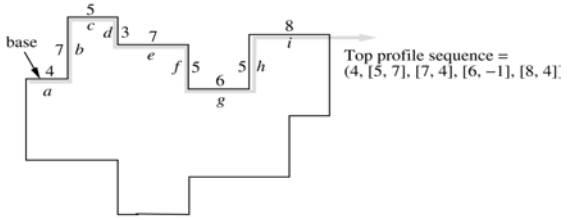

Figure 2: The top profile sequence consists of the length of the base followed by a sequence of two-tuples composed of the lengths of the succeeding horizontal segments and their relative heights to the base (could be negative).

Definition 1 *A rectilinear block placement is* feasible *if and only if no two blocks overlap with each other and all profile sequences remain unchanged after placement (i.e., all blocks are in their original shapes).*

2. Approaches

2.1 The B*-tree Representation

We propose a new representation, B*-trees, to represent an *admissible* placement. A B*-tree is an ordered binary tree whose root is the left-bottom corner module. We define two kinds of B*-tree: *horizontal B*-tree* and *vertical B*-tree*. A placement can be represented by a horizontal B*-tree and a vertical B*-tree at the same time. We denote a B*-tree, a horizontal B*-tree, and a vertical B*-tree by T, T_h, and T_v

respectively. In the following, we detail the correspondence between a placement P and a horizontal B*-tree T_h, and that between P and a vertical B*-tree T_v. Fig. 3(a) and (b) shows the T_h and T_v which represent the placement P in Fig. 1(a).

For each module b_i in P, we introduce a node n_i in T_h and T_v. We first describe the construction of T_h. Let R_i denote the set of modules located on the right-hand side and adjacent to module the b_i, and U_i denote the set of modules above and adjacent to the module b_i. Then, the left child of node n_i is the corresponding node of the lowest module in R_i, and the right child of node n_i is the module in U_i whose x-coordinate is equal to that of . bi. Similarly, for the node n_i in T_v, its left child is the corresponding node of the lowest module in U_i, and the right child is the module in R_i whose y-coordinate is equal to that of bi.

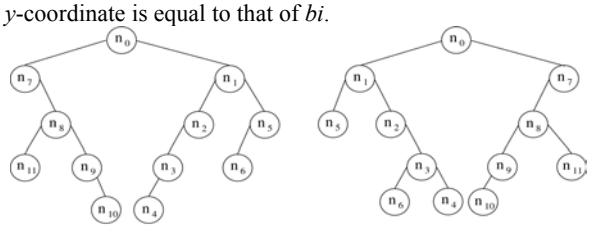

Figure 3: (a) The horizontal B*-tree T_h of the placement shown in Figure 1(a); (b) The vertical B*-tree T_v.

2.2 Coping with Rotated, Pre-placed, soft, and Rectilinear Modules

In this section, we discuss the potential solutions to handling rotated, pre-placed, soft, rectilinear, and symmetry modules placement problem and coping with interconnect planning.

2.2.1 Rotated Modules

The rotation process can be executed in the inserting step. When inserting a deleted node into a B*-tree, we can perform the operation twice at each position to find a better solution, one is for the original shape, and the other is for the rotated shape.

2.2.2 Pre-placed Modules

We propose the approach: if there exists a pre-placed module that cannot be located on its fixed positions during compaction, we exchange the pre-placed module with another so that the pre-placed module could be located on its fixed positions. There are two subproblems to be solved: (1) how to choose the module that swaps with the pre-placed module, and (2) how to locate the pre-placed one on its fixed position.

We define that a module is ahead (behind) another if its left-bottom x-coordination is smaller (larger) than that of another. Similarly, a module is below (above) another if its left-bottom y-coordination is smaller (larger) than that of another. Let b_i be a pre-placed module, and (f_i^x , f_i^y) denote its fixed coordinate. The modules that are ahead and below b_i and their coordinations are smaller than f_i^x and f_i^y are denoted by S_{ex}^i. If there are many modules ahead or below b_i so that b_i cannot be located at (f_i^x , f_i^y), we would exchange b_i with the module in S_{ex}^i that is most close to (f_i^x , f_i^y).. After deciding the elements in S_{ex}^i, we shall choose the one that is most close to (f_i^x , f_i^y) as the exchanged module.

2.2.3 Soft Modules

A soft module is flexible in its shape. With the fixed area, it is free to change the width and height in the range of its aspect ratio. We propose a heuristic algorithm to handle the placement problem with soft modules. The heuristic consists of two stages: the first stage picks a soft module for processing (deleting and inserting a node associated with the module), and the second stage adjusts the shapes of all other modules except the processed one.

2.2.3 L-shaped Modules

Let b_L denote an L-shaped block. b_L can be partitioned into two rectangular *sub-blocks* by slicing b_L along its middle vertical boundary. After partitioning and placement, the rectilinear block b_L might not conform to its top profile sequence, as illustrated in Figure 4. Figure 4(a) shows a B*-tree and its corresponding placement. We can pull sub-block b_2 up to align with the sub-block b_1, so that the block b_L can maintain its top profile sequence without changing the overall topology of the blocks. However, there might not be enough room to do so; see Figure 4(b) for such an example. It is obvious that a feasible placement can be generated from the B*-tree shown in Figure 4(a) with a local adjustment, but it is impossible for the case shown in Figure 4(b). Therefore, if we represent an L-shaped block by two sub-blocks, we must guarantee that the two sub-blocks abut. To ensure that the left sub-block b_1 and the right sub-block b_2 of an L-shaped block b_L abut, we impose the following *location constraint* (*LC* for short) for b_1 and b_2:

LC: Keep b_2 as b_1's left child in the B*-tree.

The *LC* relation ensures that the x-coordinate of the left boundary of b_2 is equal to that of the right boundary of b_1.

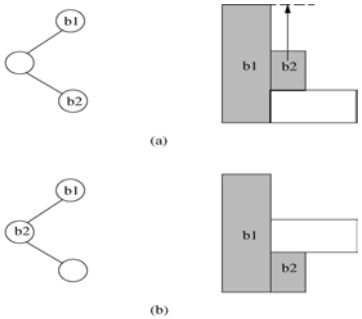

Figure 4: Placing the L-shaped block shown in Figure 8(a) by two sub blocks: (a) a feasible placement; (b) an infeasible placement.

2.2.3 Multilevel Floorplanning

Design complexities are growing at a breathtaking speed with the continuous improvement of the nanometer IC technologies. On one hand, designs with tens of million gates are already in production, IP modules are widely reused, and a large number of buffer blocks are used for delay optimization in very deep-submicron interconnect-driven floorplanning/placement, which all drive the need of a tool to handle very large-scale modules. On the other hand, the highly competitive IC market requires faster design convergence, faster design turnaround, and better silicon area utilization. Efficient and effective hierarchical design methodology and tools capable of

placing and optimizing very large-scale mixed modules and cells are essential for such large designs.

We propose to adopt a two-stage technique, clustering followed by unclustering. (1). Clustering: The clustering stage iteratively groups a set of (primitive or cluster) modules (say, two modules) based on a cost metric defined by area utilization, wirelength, and connectivity among modules, and at the same time establishes the geometric relations among the newly clustered modules by constructing a corresponding B*-subtree. The clustering procedure repeats until a single cluster containing all modules is formed, denoted by a one-node B*-tree that bookkeeps the entire clustering scheme. (2). Unclustering: The unclustering stage iteratively ungroups a set of previously clustered modules (i.e., expanding a node into a subtree according to the B*-tree topology constructed at the clustering stage) and then refines the placement/floorplanning solution based on a simulated annealing scheme. The refinement shall lead to a "better" B*-tree structure that guides the unclustering at the next level. It is important to note that we always keep only one B*-tree for processing at each iteration, and the multilevel B*-tree based placer/floorlanner preserves the geometric relations among modules during unclustering (i.e., the tree expansion), which makes the B*-tree an ideal data structure for the multilevel placement/floorplanning framework.

2.2.4 Interconnect Planning

As the process technology advances into the deep submicron era, interconnect plays a dominant role in determining circuit performance and signal integrity. Crosstalk-induced noise has been attracting increasing attention when technology improves, spacing diminishes and coupling capacitance/inductance increases. Buffer insertion/sizing is one of the most effective and popular techniques to reduce interconnect delay and decouple coupling effects. It is traditionally applied to post-layout optimization. However, It is obviously infeasible to insert/size hundreds of thousands buffers during the post-layout stage when most routing regions are occupied. Therefore, it is desirable to incorporate buffer planning into floorplanning to ensure timing closure and design convergence. In this project, we first derive formulae of buffer insertion for timing and noise optimization, and then apply the formulae to compute the feasible regions for inserting buffers to meet both timing and noise constraints.

五、成果 (Publications)

1. G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, ``Rectilinear block placement using B*-trees," in *Proc. of IEEE International Conference on Computer Design (ICCD-00)*, pp. 351-356, Austin, TX, Oct. 2000.
2. J.-M. Lin and Y.-W. Chang, ``TCG: A transitive closure graph based representation for non-slicing floorplans," in Proc. of ACM/IEEE Design Automation Conference (DAC-2001), pp. 764--769, Las Vegas, NV, June 2001.
3. C.-Y. Chang, H.-R. Jiang, and Y.-W. Chang, "Formulate for performance optimization and their applications to interconnect-driven floorplanning," to appear in 2002 IEEE International Symposium on Quality of Electronic Design (ISQED 2002), San Jose, CA, March 2002.
4. G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, ``Rectilinear block placement using B*-trees," submitted to ACM Trans. on Design Automation of El*ectronic Systems*
5. S.-C. Lee, J.-M. Hsu, and Y.-W. Chang, "Multilevel large-scale module floorplanning," to be submitted to ACM Int'l Symp. On Physical Design..

六、參考文獻

1. Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," *Proc.IEEE/ACM Design Automation Conference*, 2000.
2. P.-N. Guo, C.-K. Cheng, and Takeshi Yoshimura, ``An O-Tree Representation of Non-Slicing Floorplan and Its Applications," *Proc.IEEE/ACM Design Automation Conference*, 1999.
3. M. Z. Kang and W. Dai., ``General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 265-270, 1997.
4. M. Z. Kang and W. Dai., ``Arbitrary Rectilinear Block Packing Based on Sequence Pair," *Proc. International Conference on Computer-Aided-Design*, pp. 259--266, 1998.
5. T. C. Lee, ``An Bounded 2D Contour Searching Algorithm for Floorplan Design with Arbitrarily Shaped Rectilinear and Soft Modules," *Proc.IEEE/ACM Design Automation Conference*, pp. 525--530, 1993.
6. H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, ``Rectangle-Packing Based Module Placement," *Proc. International Conference on Computer-Aided-Design*, pp. 472-479, 1995.
7. H. Murata, K. Fujiyoshi, and M. Kaneko, ``VLSI/PCB Placement with Obstacles Based Sequence Pair," *Proc. Internal Symposium on Physical Design*, pp. 26-31, 1997.
8. H. Murata, Ernest S. Kuh, ``Sequence Pair Based Placement Method for Hard/Soft/Pre-placed Modules," *Proc. Internal Symposium on Physical Design*, pp. 167-172, 1998.
9. S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, ``Module Placement on BSG-Structure and IC Layout Applications," *Proc. International Conference on Computer-Aided-Design*, pp. 484-491, 1996.
10. S. Nakatake, M. Furuya, and Y. Kajitani, ``Module Placement on BSG-Structure with Pre-Placed Modules and Rectilnear Modules," *Proc. Asia and South Pacific Physical Design Automation Conf.*, pp. 571-576, 1998.
11. R. H. J. M. Otten, Automatic Floorplan Design," *Proc.IEEE/ACM Design Automation Conference*, pp.261-267,1992.
12. Christos H. Papadimitriou, and Kenneth Steiglitz, *Combinatorial Optimization*, prentice Hall, 1982.
13. B. T. Preas, and W. M. vanCleemput, ``Placement Algorithms for Arbitrarily Shaped Blocks," *Proc. IEEE/ACM Design Automation Conference*, pp.474-480, 1979.
14. T. C. Wang, and D. F. Wong, ``An Optimal Algorithm for Floorplan and Area Optimization," *Proc. IEEE/ACM Design Automation Conference*, pp.180-186, 1990.
15. D. F. Wong, and C. L. Liu, ``A New Algorithm for Floorplan Design," *Proc. IEEE/ACM Design Automation Conference*, pp. 101--107, 1986.
16. D. F. Wong, and C. L. Liu, ``Floorplan Design for Rectangular and L-shaped Modules," *Proc. IEEE International Conference on Computer-Aided-Design*, pp. 520--523, 1987.
17. J. Xu, P.-N. Guo, and C. K. Cheng, ``Cluster refinement for block placement," *Proc. IEEE/ACM Design Automation Conference*, pp. 762--765, 1997.
18. Jin Xu, Pei-Ning Guo, and Chung-Kuan Cheng, ``Rectilinear Block Placement Using Sequence-Pair," *Proc. Internal Symposium on Physical Design*, pp. 173--178, 1998.
19. F. Balasa and K. Lampaert, " Module placement for analog layout using the sequence-pair representation," *Proc. of ACM/IEEE Design Automation Conference*, pp. 274-279, June 1999.
20. J. Cohn, D. Garrod, R. Rutenbar, and L. Carley, Analog Device-Level Automation, Kluwer Academic Publishers, 1994..
21. H.-M. Chen, et al, "Integrated floorplanning and interconnect planning," Proc. of IEEE/ACM International Conference on Computer-A*ided Design*, pp. 354-357, Nov. 1999.