第 25 卷第 11 期 半 导 体 学 报 Vol. 25, No. 11
2004 年 11 月 CHINESE JOURNAL OF SEMICONDUCTORS Nov., 2004

Arbitrary Rectilinear Block Packing Based on Less Flexibility First Principles*

Yang Zhong[1,2], Dong Sheqin[1], Hong Xianlong[1] and Wu Youliang[3]

(1 *Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)
(2 *Graduate School at Shenzhen, Tsinghua University, Shenzhen* 518057, *China*)
(3 *Department of Computer Science and Engineering, The Chinese University of Hong Kong, China*)

Abstract: The definition and formula of flexibility of arbitrary rectilinear blocks are introduced and LFF principles are extended to handle arbitrary rectilinear blocks and blocks with relative constraints. The experimental results demonstrate the efficiency and effectiveness of the proposed method.

Key words: LFF principles; rectilinear block packing; deterministic placement algorithm
EEACC: 2570 **CCACC**: 7410D
CLC number: TN405.97 **Document code**: A **Article ID**: 0253-4177(2004)11-1416-07

1 Introduction

Due to the layout complexity in deep submicron technology, integrated circuit components are not limited to rectangular blocks. Therefore new approaches are essential in VLSI design to handle arbitrary rectilinear shaped blocks effectively and efficiently.

Up to now, placement/floorplanning with rectilinear blocks has been extensively studied. Most previous works partitioned a rectilinear block into a set of rectangular sub-blocks with some constraints induced from the original rectilinear blocks. There is some literature on the rectilinear block packing problem based on well-known representations such as corner block list (CBL)[4,5], O-Tree[6], bounded sliceline grid (BSG)[7,8], sequence pair (SP)[9,10], transitive closure graph (TCG)[11].

Ma *et al.*[4] proposed a new method based on abutment constraint algorithm and transformed the L/T-shaped block into abutment constraint problem. Later they extended the stairway compact approach based on CBL representation to handle the placement with arbitrary shaped rectilinear blocks[5].

Pang *et al.*[6] decomposed rectilinear blocks into a set of sub-L-shaped-blocks and explored the packing for L-shaped blocks and derived the L-admissible O-tree which can always lead to a feasible solution.

Nakatake *et al.*[8] devised a method to handle pre-placed and rectilinear blocks. They partitioned a rectilinear block into a set of rectangle sub-blocks and placed these sub-blocks one by one until all sub-blocks were packed at right relative positions. Then, the placed sub-blocks were treated as pre-placed blocks.

Xu *et al.*[9] explored the conditions of "feasibility of sequence-pair" for L-shaped block. They

* Project supported by NSFC and Hongkong RGC Joint Project (No. 60218004), National Education Promotion Project (Tsinghua) (No. Jc2001025) and Key Project of National High Technology Research and Development Program of China (No. 2002AA1Z1460)

Received 15 November 2003, revised manuscript received 7 June 2004

partitioned a rectilinear block into L-shaped blocks. However, after packing, a post processing was performed to adjust misplaced sub-blocks. Fujiyoshi and Murata[10] also derived a necessary and sufficient condition for feasible sequence pair for rectilinear blocks by the properties of the horizontal and vertical constraint graphs. However, the constraint graphs are no longer acyclic after their augmentation, resulting in a longer running time for packing.

Lin *et al.*[11] partitioned a rectilinear block into a set of sub-blocks and then derived necessary and sufficient conditions of feasible TCG for the blocks. This algorithm treated a set of sub-blocks as a whole and could eliminate the need of the post processing on deformed blocks.

In this paper, we propose a deterministic algorithm using less flexibility first (LFF) principles to handle arbitrary rectilinear blocks. We partition rectilinear block into a set of sub-blocks. The left-lower sub-block is the main-block, the others are vice-blocks. Then we give definition of flexibility for rectilinear blocks. We place the main-block firstly and vice-blocks immediately with consideration of rotation. Unlike most of previous methods which process each sub-block individually and thus need post processing to fix deformed rectilinear blocks, we treat a set of sub-blocks as a whole and thus can guarantee the shape and size of rectilinear block without the post processing on deformed blocks. This method can be extended to handle the blocks with relative constraints that are not connected. Experiment results prove that the algorithm is effective.

2 Placement using LFF principles

When a mason packs the polygon-shape stone plate in everyday work, they try to use up the more restricted resource from the corner and fulfill more requirements, tending to leave the more "general" items at the later stage of processing to have more chance to find a match. The schedule sequence of packing tasks can be named as "the less flexibility first(LFF)" principle.

Dong *et al.*[1] introduced three different flexibilities in a working area: (1) flexibility of empty space, (2) flexibility of a rectangle to be packed, (3) flexibility between two blocks and formulas for each of them.

In a working area, flexibility of empty space represents the directions at which a rectangle block can move. If we define one direction of move flexibility as one freedom, obviously, the corner/side packed rectangle block will has 3/5 freedoms, respectively, and hence a rectangle block at hollow space will has 8 freedoms. This is depicted in Fig. 1.

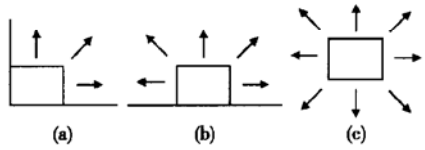


Fig. 1 Flexibility of empty space (a) Flexibility of corner packing; (b) Flexibility of side packing; (c) Flexibility of hollow packing

Flexibility of a rectangle block depends on its shape and size. A rectangle block with larger size or rectangle with longer side will have less flexibility.

Flexibility between two blocks describes the number of nets between two blocks. The more number of nets, the closer two blocks must be packed, and the less flexibility between two blocks.

Based on the three flexibilities described above, a deterministic placement algorithm has been proposed. The rectangle block with less flexibility would be packed earlier at corner. It will occupy at least one corner, at the same time, new corners generate. The rest blocks will be packed at the corner that makes their flexibility the least. Placement will be completed in this way from the original four corners step by step.

More details are shown in Ref. [1].

3 Placement of rectilinear block

3.1 Partition, definition, and representation

We partition a rectilinear block into a set of sub-blocks. The left-lower sub-block is the main-block, the others are the vice-blocks. According to the algorithm based on LFF principles[1], we know larger block has less flexibility than smaller one, so we should maximize the main-block of rectilinear block, which is placed earlier than vice-block to achieve a feasible placement easily. We can partition rectilinear block in the following step. Firstly, we rotate the rectilinear block to maximize the lower-left sub-block. Secondly, we divide the rest into sub-blocks as less as possible. This can improve the speed of placement. Figure 2 is a sample of rectilinear block partition, where a is main-block, both b and c are vice-blocks. Both the partitions in Figs. 2(a) and (b) are acceptable.

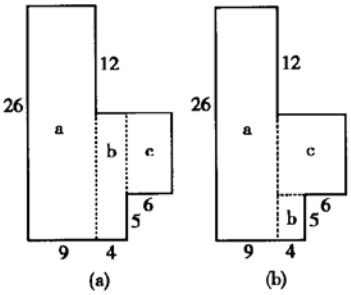


Fig. 2 Results of rectilinear block partition

As shown in Fig. 3, it is obvious that the flexibility of the rectilinear block is greater than the main-block and smaller than the circumscribed rectangle. We use the original definition of flexibility for rectangle block and add some compensation.

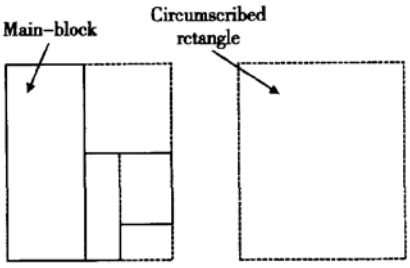


Fig. 3 Rectilinear block and its circumscribed rectangle

We define the flexibility of rectilinear block as below:

$$M_i^f = R_{im}^f + \omega(R_{io}^f - R_{im}^f) \quad (1)$$

where M_i^f is the flexibility of rectilinear block, R_{im}^f is the flexibility of main-block (rectangle flexibility definition shown in Ref. [1]), R_{io}^f is the flexibility of circumscribed rectangle. Parameter ω can vary from 0 to 1. If $\omega = 0$, the flexibility of rectilinear block is the main-block's flexibility, and if $\omega = 1$, the flexibility of rectilinear block is the circumscribed rectangular flexibility.

The following describes how to represent rectilinear block. The representation of rectilinear block should satisfy:

(1) Recording the size of sub-block;

(2) Maintaining the relation between sub-blocks.

So we can use a set of four-tuple (width, height, rx, ry) to represent a rectilinear block. The first one is main-block, the others are vice-blocks. The width and height are values of the horizontal and vertical sides of sub-block. (rx, ry) is left-lower corner coordinate of sub-block related to main-block. $rx = -1$, $ry = -1$ for the main-block. The rectilinear block in Fig. 2(a) can be represented as follow:

(9, 26, − 1, − 1)
(4, 14, 9, 0)
(6, 9, 13, 5)

3.2 Rotation of rectilinear block

In the placement procedure, the rotation of rectilinear block must be considered. For rectilinear block, it has eight possible rotation positions. Because we decompose rectilinear block into rectangles, we only need to know the rotation rule of rectangle in order to maintain the constraint between sub-blocks. If we know the coordinates of left-lower and right-upper corners, we can construct the rectangle. Thus we transform the rotation of rectangle to the rotation of two points.

A point, $A(x, y)$, has eight cases of rotation in coordinate system shown in Table 1.

Table 1 Rotation rule of point

Clockwise				Reflection and clockwise			
0°	90°	180°	270°	0°	90°	180°	270°
(x, y)	$(y, -x)$	$(-x, -y)$	$(-y, x)$	$(-x, y)$	(y, x)	$(x, -y)$	$(-y, -x)$

With the rotation rule of point, we can handle the rotation of rectilinear block.

3.3 Packing

In a given rectangle chip area, it is reasonable that packed rectangle block must occupy one empty corner of the current chip packing configuration. When an unpacked block is packed in the selected corner, a new current chip packing configuration is formed. There are many candidate corners for a packing step and one corner has two packing orientations for a rectangle. For example, in Fig. 4, there are total 7 corners in the current chip packing configuration, and a block B can be packed at any one of them with two possible orientations. As a result, there are 14 packing choices for block B. The real line block form the current chip packing configuration.

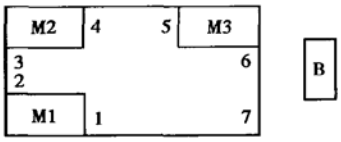


Fig. 4 Candidate packing corners for block B

In terms of the flexibility definitions discussed above, we define one choice of this kind as a candidate corner packing step (CCPS). A CCPS can be represented by a six-tuple as ⟨block_id, length, width, orientation, x, y⟩. The block_id is block name, the length and width are values of the longer and shorter sides of the packed block, orientation indicates that the rectangle is placed with its longer side laid horizontally or vertically. (x, y) is the left-lower corner coordinate of the suggested location.

Under the current chip packing configuration, for an unpacked block i we have a list of candidate CCPSs. For example, in the current chip packing configuration in Fig. 4, unpacked block B has 14 CCPSs. We sort all CCPSs of all unpacked blocks for the current chip packing configuration in a list according to the definition of the flexibility of a rectangle. In the list, an earlier CCPS possesses a higher priority. Then the unpacked blocks are pseudo-packed with a greedy strategy following the list order until no rectangle can be packed in the present chip packing configuration.

For a CCPS, a fitness value F_i is calculated as follows:

$$F_i = P^f + M_i^f \tag{2}$$

where P^f is the total area of blocks that can be packed in this CCPS.

Under the current chip packing configuration, block whose CCPS has the highest F_i will be selected as the next packing step. After this step, a new current chip packing configuration is formed and the rest unpacked blocks again have a new set of CCPSs. This placement process iterative to have all modules placed in the chip area.

When we meet a rectilinear block, we treat the main-block of rectilinear block as same as the other rectangle blocks. When the main-block is settled, we place the related vice-block immediately and try the possible rotations. For example, in Fig. 5, M1, M2, M3 are packed blocks. PA is a rectilinear block to be packed. Block P is the main-block and block

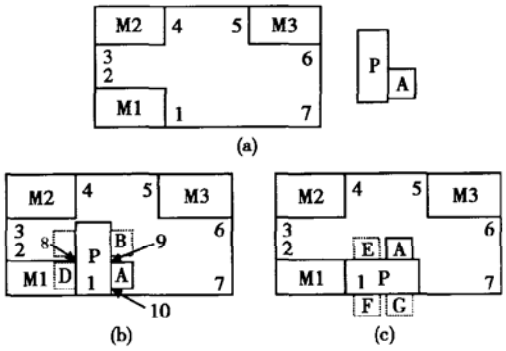


Fig. 5 Rectilinear block packing

A is the vice-block, when the main-block P is settled in corner 1. The possible rotation cases of the

rectilinear block are eight. B, C, D, E, F, G are the possible positions of vice-block A after rotation. The overlapped case, such as D, and exceeding bound case, such as F, G, will be eliminated. Supposing that the vice-block is placed at position A, three new corners (8, 9, 10) come out. They can be used by the rest blocks.

3.4 Placement with relative constraint

The flexibility partition, definition, representation and rotation rule of rectilinear block described above do not require the rectilinear block to be continued. So it also function in blocks with relative constraint, which are not connected. We can name the left-lower block of these blocks with related constraint main-block, the others are vice-blocks. But the related position between blocks must be fixed. For example, in Fig. 6, block A and B are two blocks with related constraint which are not connected, x_1, x_2, y_1, y_2 are constant. The room between blocks can also be made use of.

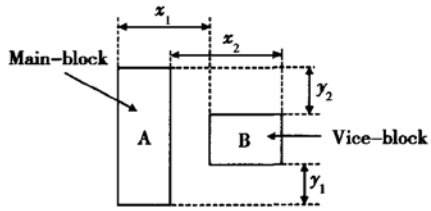


Fig. 6 Blocks with relative constraint which are not connected

4 Algorithm

We implemented the algorithm R_LFF. The algorithm can be summarized as follows. In fact, we treat a rectangle block as a rectilinear block with one main-block and no sub-block. So we can treat rectangular block and rectilinear block as the same.

Assume that:
B_{BS}: Block set to be packed
U_{BS}: Unpacked block set
P_{BS}: Packed block set
P_L: Current placement configuration
F_i: Flexibility of block i
w_1, w_2: weight for the two flexibility

Deterministic algorithm: R_LFF
Input: array of blocks with width, height
Output: blocks with position and orientation
Flow chart of Algorithm:

(1) $U_{BS} = B_{BS}$
(2) Updated the Placement with an estimated blank rectangle with required ratio of length and width;
(3) while UBS is not empty and there is space for packing in P_L
{
(4) for each block i in UBS, under the current , calculate
(5) $F_i = P^f + M_i^f$
(6) Update P_L with the block of the smallest F_i packed in it
(7) Add the main-block and related vice-block of the smallest F_i to P_{BS}
(8) Remove the main-block and related vice-block of the smallest F_i from U_{BS}
}

5 Experiment results

We implemented the deterministic algorithm using C programming language and run it on a Linux workstation with P4 1.7GHz CPU and 128M memory. There is not standard circuit benchmark for rectilinear block packing. Most of researchers

use the modified MCNC benchmarks that are constructed by themselves to test their method. We apply our algorithm on the modified MCNC benchmark circuits. We add some additional rectilinear blocks to Microelectronics Center of North Carolina (MCNC) benchmark circuit, including L-shaped, T-shaped and some more complicated rectilinear blocks and blocks with related constraint which are not connected.

In Eq. (1) parameter ω is a very important parameter. We use the modified MCNC benchmark ami33 and ami49 to study the rule of it. We try different ω with area usage varied in the range of 90% ~ 99%, aspect ratio of the chip is set to 1 : 1.5, 1 : 1.2, 1 : 1, 1.2 : 1, 1.5 : 1. Figure 7 shows the rule between the ratio of success and ω. We can see $\omega = 1$ is better. It is also reasonable that giving the less flexibility for rectilinear block is easier for its packing.

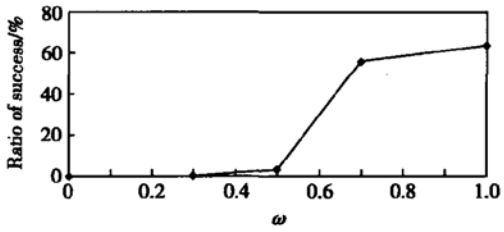
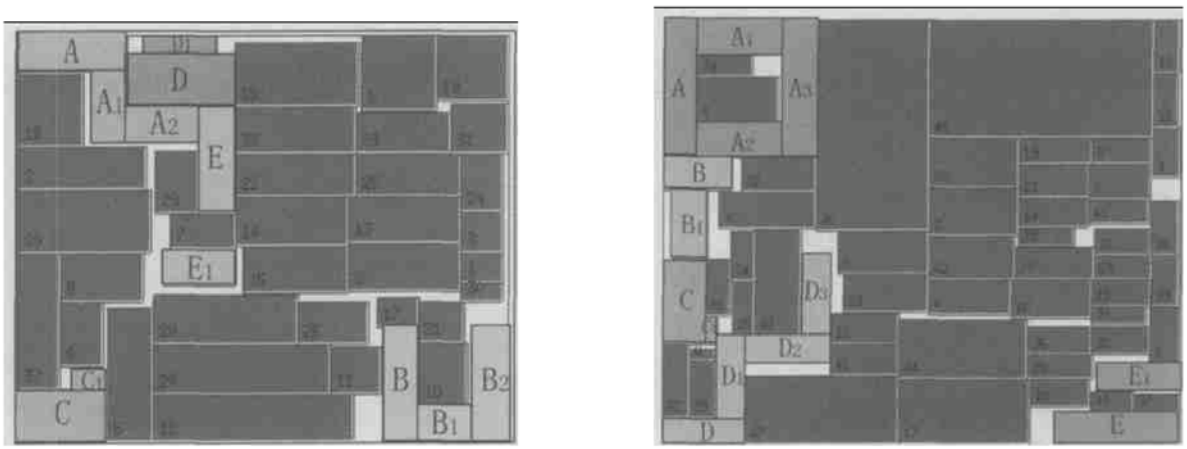


Fig. 7 Rule between the ratio of success and ω

Our experiment results are shown in Table 2 and Fig. 8. We can see that both concave and convex blocks can be handled and the performance is quite good. We can not directly compare our method with others, such as CBL[5], O-tree[6], SP[9], TCG[11], but we can designate the length and width of working area and all the blocks' shape is fixed, not flexible. Our rectilinear blocks are more complicated. We obtain satisfactory area usage (more than 93%) and other methods can hardly achieve it.

Table 2 Results of testing rectilinear blocks

	# Total blocks	# L-shaped	# T-shaped	# Other rectilinear blocks	Net area /mm²	area /mm²	Dead space /%	Time /s
Ami33_R	39	1	1	4	1.455	1.562(1.369×1.141)	6.83	38.82
Ami49_R	55	1	1	4	43.684	46.236(7.449×6.207)	5.52	246.38



(a) (b)

Fig. 8 Experiment results (a) Modified MCNC Benchmark ami33, including 4 rectilinear blocks and two blocks with related constraint (blocks E and E_1), area usage 93.17%; (b) Modified MCNC Benchmark ami49, including 4 rectilinear blocks and tow blocks with related constraint (blocks E and E_1), area usage 94.48%

References

[1] Dong Sheqin, Hong Xianlong, Wu Youliang, et al. VLSI block placement using less flexibility first principles. Proc IEEE ASPDAC, 2001: 601

[2] Wu Yuliang, Huang Wenqi, Lau Siuchung, et al. An effective quasi-human based heuristic for solving rectangle packing

problem. Proc IEEE APCCAS: Microelectronic and Integration System, 1998: 137

[3] Dong S, Hong X, Wu Y L, et al. VLSI placement with preplaced modules based on less flexibility first principles. Proc IEEE International Conference on ASIC (ASICON), 2001: 106

[4] Ma Y, Hong X, Dong S, et al. Floorplanning with abutment constraints and L-shaped/T-shaped blocks based on corner block list. Proc DAC, 2001: 770

[5] Ma Yuchun, Hong Xianlong, Dong Sheqing, et al. Stairway compaction using corner block list and its applications with rectilinear blocks. Proceedings of 7th IEEE/ACM Asia & South Pacific Design Automation Conference, 2002

[6] Pang Y, Cheng C K, Lampasert K, et al. Rectilinear block packing using O-tree representation. Proc ISPD, 2001: 156

[7] Kang M, Dai W. General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure. Proc ASP-DAC, 1997: 265

[8] Nakatake S, Furuya M, Kajitani Y. Module placement on BSG-structure with pre-placed modules and rectilinear modules. Proc ASP-DAC, 1998: 571

[9] Xu J, Guo P N, Cheng C K. Sequence-pair approach for rectilinear module placement. IEEE Trans on CAD, 1999: 484

[10] Fujiyoshi K, Murata H. Arbitrary convex and concave rectilinear block packing using sequence-pair. IEEE TCAD, 2000, 19(2): 224

[11] Lin J M, Lin H L, Chang Y W. Arbitrary convex and concave rectilinear module packing using TCG. IEEE Trans VLSI Syst, to be published

[12] Lee T C. A bounded 2-D contour searching algorithm for floorplan design with arbitrarily shaped rectilinear and soft modules. Proc 30th Design Automation Conf, 1993: 525

[13] Kong Tianming, Hong Xianlong, Qiao Changge. VEAP: efficient VLSI placement algorithm based on global optimization. Chinese Journal of Semiconductors, 1997, 18(9): 692(in China)[孔天明, 洪先龙, 乔长阁. VEAP: 基于全局优化的有效VLSI布局算法. 半导体学报, 1997, 18(9): 692]

[14] Hou Wenting, Yu Hong, Hong Xianlong, et al. A new congestion-driven placement algorithm based on cell inflation. Chinese Journal of Semiconductors, 2001, 22(3): 275

基于最小自由度优先原则的任意多边形模块布局算法*

杨 中[1,2] 董社勤[1] 洪先龙[1] 吴有亮[3]

(1 清华大学计算机科学与技术系, 北京 100084)

(2 清华大学深圳研究生院, 深圳 518057)

(3 香港中文大学计算科学与工程系, 香港)

摘要: 给出了直角多边形模块自由度的定义和公式,扩展了最小自由度优先原则,使算法能够处理任意直角多边形模块以及有相对约束的模块. 实验结果说明该方法在布局效果和效率上都有良好的表现.

关键词: 最小自由度优先原则; 直角多边形模块布局; 确定性布局算法

EEACC: 2570 **CCACC**: 7410D

中图分类号: TN405.97 **文献标识码**: A **文章编号**: 0253-4177(2004)11-1416-07

* 国家自然科学基金与香港研究资助局联合资助(编号: 60218004), 国家教育振兴计划(清华)(编号: Jc2001025)和国家高技术研究发展计划重点工程(编号: 2002AA1Z1460)资助项目

2003-11-15 收到, 2004-06-07 定稿