第 23 卷第 6 期 半 导 体 学 报 Vol. 23, No. 6
2002 年 6 月 CHINESE JOURNAL OF SEMICONDUCTORS June, 2002

Cross Point Assignment Algorithm Under Consideration of Very Long Nets*

Zhang Yiqian, Xie Min, Hong Xianlong and Cai Yici

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

Abstract: A cross point assignment algorithm is proposed under consideration of very long nets (LCPA). It is to consider not only the cost of connection between cross points and pins and the exclusive cost among cross points on the boundary of a global routing cell, but also the cost of displacement among cross points of the same net. The experiment results show that the quality and speed in the following detailed routing are improved obviously, especially for very long nets.

Key words: cross point assignment; layout; VLSI
EEACC: 2570 **CCACC**: 7410D
CLC number: TN47 **Document code**: A **Article ID**: 0253-4177(2002)06-0582-07

1 Introduction

Due to the high complexity of VLSI layout design, nowadays a top-down strategy is generally adopted in the layout design.

Most systems divide the routing process into three phases: global routing, cross point assignment (CPA), and detailed routing. Global routing divides the chip area into a two-dimensional array of global routing cells(GRCs). This phase finds a Steiner tree on the GRC array for each net under certain constraints, without exact crossing locations of each wire on the GRC boundaries. After global routing is finished, CPA is required, followed by detailed routing.

CPA is to determine the exact crossing location on the GRC boundary for each net, while taking into consideration the effect of pins and obstacles in the GRC. The goal is to minimize the total wire length and the total number of vias in the detailed routing, and to improve the quality of final routing and the electronic performance in the circuit.

Up to now, the CPA problem has not been well addressed, and there are only a few algorithms published.

In Reference[1] a CPA algorithm was proposed based on the analysis of the category of the wiring patterns in a GRC after global routing. It reduces the CPA problem of the whole chip to a series of CPA problems in the single column or row. It has been proved that CPA problem in a single column or row is independent on each others[2]. The CPA algorithm tries to overcome the dependency of the CPA result on the order in which the net and the boundary of GRC are processed. The algorithm considers the effect of other cells' pins and obstacles in the GRC. The algorithm proposed in Ref. [3] tries to solve detoured wiring problem for long nets, and furthermore, to avoid over-congested cross points on a boundary. In this paper we

* Project supported by National Natural Science Foundation of China (Grant No. 60167016)
Zhang Yiqian female, was born in 1977, PhD candidate. Her research interests focus on VLSI layout design.
Received 24 October 2001, revised manuscript received 4 December 2001

propose a CPA algorithm under consideration of very long nets. The cost of displacement between two cross points next to each other was taken into consideration. The displacement has a major effect on the shape of the final routing. The wider the span is, the larger the displacement cost is. Our experiment results show that for long nets, introducing the displacement cost could greatly improve the routing shape by stretching the very long nets as straight as possible. It also reduces the number of vias and the total length of wire.

2 Model analysis

In this section, we take a vertical CPA as an example to demonstrate the mathematical model employed in our algorithm. The algorithm could be extended easily to the horizontal CPA.

We study a whole column every time. First, we give some definitions as follows:

$\mathrm{Col} = \{\mathrm{Grc}_1, \mathrm{Grc}_2, \cdots \mathrm{Grc}_{k+1}\}$ is a set of all GRCs in a column, with GRC_{i+1} following GRC_i.

$E = \{e_1, e_2, \cdots e_k\}$ is a set of all boundaries in a Col, where $e_i = \mathrm{GRC}_i \cap \mathrm{GRC}_{i+1}$

$N = \{n_1, n_2, \cdots, n_s\}$ is a set of all nets within a Col which has more spans than a GRC. That is, for those nets CPA has to be performed.

$N_i(e_i) = \{n \mid n \in N \wedge n \cap e_i \neq \varnothing\}$ is the set of all nets in N going across boundary e_i.

$\mathrm{CP}_i(e_i) = \{c \mid c \in e_i \wedge \neg \text{ occupied}(c)\}$ is the set of all available locations of cross points on boundary e_i.

$P_j(n_j) = \{c_{jk} \mid \mathrm{Edge}(c_{jk}) \in E_j(n_j) \wedge \mathrm{Owner}(c_{jk}) = n_j\}$ is the set of cross points for net n_j, where $\mathrm{Edge}(c_{jk})$ is the boundary of GRC which c_{jk} is on, and $\mathrm{Owner}(c_{jk})$ is the net which c_{jk} belongs to.

$E_j(n_j) = \{e \mid e \in E \wedge n_j \in P_i(e_i)\}$ is the set of edges which n_j goes through.

pu_{jk} is the terminal in the upper GRC of the k-th cross point of the net n_j. If there is no terminal in that GRC, pu_{jk} is recursively defined as $\mathrm{pu}_{j,k+1}$. Similarly, pd_{jk} is the terminal in the lower GRC of the k-th cross point of net n_j. Since the global routing tree is generated by the connection relationship of net, there must be a physical terminal in the GRC related to top and bottom edges of $E_j(n_j)$.

For an available cross point c_{ir} on boundary e_i, we define $[c_{ir-l+1}, c_{ir+l-1}]$ as its exclusive interval.

The CPA problem is to determine matrix for every e_i and every N_i:

$$A_i = \begin{vmatrix} a_{i1,1} & a_{i1,2} & \cdots & a_{i1,|N_i(e_i)|} \\ a_{i2,1} & a_{i2,2} & \cdots & a_{i2,|N_i(e_i)|} \\ \cdots & \cdots & \cdots & \\ a_{i|\mathrm{CP}_i(e_i)|,1} & a_{i|\mathrm{CP}_i(e_i)|,2} & \cdots & a_{i|\mathrm{CP}_i(e_i)|,|N_i(e_i)|} \end{vmatrix}$$

where A_i is called the assignment matrix on edge e_i, in which each row vector expresses the allocation of each available cross point, and each column vector indicates which cross point is assigned for each crossing net on e_i. $a_{ip,q} = 1$ if and only if the pth cross point on e_i is assigned to the qth crossing net. The matrix should satisfy the following conditions:

$$\sum_{q=1}^{|\mathrm{CP}(e_i)|} a_{ip,q} = 1 \qquad 1 \leqslant p \leqslant |N_i(e_i)|$$

This means that a net has one and only one cross point.

$$\sum_{p=1}^{|N_i(e_i)|} a_{ip,q} \leqslant 1 \qquad 1 \leqslant q \leqslant |\mathrm{CP}_i(e_i)|$$

This means that a cross point can be allocated to one net at most.

$$a_{ip,q} = 0 \quad \text{or} \quad 1$$

The connection cost of net n_j is defined as

$$\mathrm{cost}(n_j) = \sum_{k=1}^{|E_j(n_j)|} \mathrm{conn}(c_{jk}) + \sum_{k=1}^{|E_j(n_j)|-1} \mathrm{dist}(c_{jk}, c_{j,k+1})$$

where $\mathrm{conn}(c_{jk})$ represents the cost of connection between the cross point and its upper and lower terminals, and $\mathrm{dist}(c_{jk}, c_{j,k+1})$ represents the cost of placement between two neighboring cross points of the same net. Let a_{ij} be the column vector in A_i corresponding to net n_j. The cost of CPA for a single net could be written as

$$\mathrm{cost}(n_j) = \sum_{i}^{|E_j(n_j)|} c_{ij} a_{ij} + \sum_{i=1}^{|E_j(n_j)|} a_{ij} d_{ij} a_{i+1,j}$$

where matrix c_{ij} is the matrix of connection cost for

net n_j on edge e_i, and d_{ij} is the matrix of displacement cost. Since c_{ij} and d_{ij} are independent on the result of CPA, they can be constructed in advance.

Now the connection cost of all the nets in the column Col is

$$\text{cost(col)} = \sum_{j=1}^{s} \text{cost}(n_j)$$

which could be rewritten as

$$\text{cost(col)} = \sum_{i=1}^{h} L_i A_i + \sum_{i=1}^{k-1} A_i^t D_{i,i+1} A_{i+1}$$

where L_i and $D_{i,i+1}$ are the connection matrix and displacement matrix, on boundary e_i of the GRC respectively.

For now, the analysis above does not include the correlation among cross points for different nets on the same GRC boundary. To estimate such correlation, we do the following operation for each A_i.

If we define vector $s = \{1, 1, \cdots\cdots 1_{|N(e_i)|}\}^t$, $A_i s$ indicates which cross points on boundary e_i have been assigned. Now we introduce matrix G_i:

$$G_i = \begin{vmatrix} g_{i1,1} & g_{i1,2} & & & & \\ g_{i2,1} & g_{i2,2} & \cdots & & & \\ \cdots & \cdots & \cdots & g_{ir-1+1,r} & & \\ g_{i1,1} & \cdots & \cdots & \cdots & \cdots & \\ & g_{i1+1,2} & \cdots & g_{ir,r} & \cdots & \\ & & \cdots & \cdots & \cdots & g_{i|\text{CP}(e_i)|-1+1,|\text{CP}(e_i)|} \\ & & & g_{ir+1-1,r} & \cdots & \cdots \\ & & & & & g_{i|\text{CP}(e_i)|,|\text{CP}(e_i)|} \end{vmatrix}$$

G_i is called the exclusiveness matrix on boundary e_i, where l is the exclusive interval. Obviously the diagonal elements $g_{ir,r} = 0$, for $1 \leqslant r \leqslant |\text{CP}(e_i)|$.

$(A_i s)^t G_i (A_i s)$ could represent the sum of the exclusiveness costs of any two available cross points. By setting the non-zero elements in matrix G_i, we can build the exclusiveness cost of the GRC boundaries or some parts of boundaries.

So far, the cost of CPA for the column Col could be written as

$$\text{cost(col)} = \sum_{i=1}^{h} L_i A_i + \sum_{i=1}^{k=1} A_i^t D_{i,i+1} A_{i+1} + \sum_{i=1}^{k} (A_i s)^t G_i (A_i s) \quad (1)$$

where the first term is the connection cost, the second is the displacement cost, and the last is the exclusiveness cost.

The minimum cost CPA is a quadratic programming problem. Its variables could only take a value of 0 or 1, so it is an integer quadratic programming problem.

3 LCPA algorithm

Integer quadratic programming is an NP hard problem. We simplify the CPA problem to several linear assignment problems. Then solve them one by one on the GRC boundary.

The algorithm works heuristically, by assigning cross points on each GRC boundary from the bottom-most one upward. Once a GRC boundary has been processed, the resulting locations of cross points for the nets are used to update the displacement cost for the next GRC boundary. We rewrite cost(col) as

$$\begin{aligned} \text{cost(col)} &= \sum_{i=1}^{h} L_i A_i + \sum_{i=1}^{k-1} A_i^t D_{i,i+1} A_{i+1} + \sum_{i=1}^{k} (A_i s)^t G_i (A_i s) \\ &= L_1 A_1 + \sum_{i=2}^{k} (L_i + A_{i-1}^t D_{i-1,i}) A_i + \sum_{i=1}^{k} (A_i s)^t G_i (A_i s) \end{aligned} \quad (2)$$

in which $L_i' = L_i + A_{i-1}^t D_{i-1,i}$ could be viewed as the correcting connection cost.

As to the exclusiveness cost, our algorithm utilizes the method of priority queue in Reference [3]. It only assigns cross points for a certain subset of the nets in Col with some priority. After the current queue is processed, the result is substituted into the exclusiveness cost to get the linear exclusiveness cost of other available cross points. The cost could be represented by a linear function of A_i, and then be combined into the connection cost. Finally, the cost for the whole column cost(Col) could be written as:

$$\text{cost(col)} = \sum_{h=1}^{m}\left(L_{h1}A_{p1} + \sum_{i=2}^{k}\left(L_{hi} + A_{hi}^{t}D_{hi-1,i}\right)A_{hi} + \sum_{i=1}^{k}\left(A_{h-1,is}\right)^{t}G_i\left(A_{his}\right)\right) \quad (3)$$

We decompose the cost in such a way that for each priority level, the objective function for each GRC boundary is approximated by a linear function.

3.1 Preparation of our algorithm

Before running the algorithm, we have to construct the priority queue. Based on each global routing tree, we can estimate roughly the total wire length of each net, sort them in the decreasing order of the length, and divide them into m priority levels. In our algorithm, let m be 3. Larger m might provide a better estimation of the exclusiveness cost, but the complexity may also be increased. After the queue is constructed, the algorithm is performed on all the nets with the same priority once at a time.

3.2 Linear assignment algorithm

As stated before, this algorithm reduces the CPA problem in a column to multiple sub-problems of sequential CPA on a GRC boundary. The sub-problem on each GRC boundary is to determine

$$A_{i_h} = \begin{vmatrix} a_{i_h1,1} & a_{i_h1,2} & \cdots & a_{i_h1,|N_{i_h}(e_i)|} \\ a_{i_h2,1} & a_{i_h2,2} & \cdots & a_{i_h2,|N_{i_h}(e_i)|} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i_h|\mathrm{CP}_{i_h}(e_i)|,1} & a_{i_h|\mathrm{CP}_{i_h}(e_i)|,2} & \cdots & a_{i_h|\mathrm{CP}_{i_h}(e_i)|,|N_{i_h}(e_i)|} \end{vmatrix}$$

$$\min L''A_{hi}$$

$$\text{st.} \quad \sum_{q=1}^{|\mathrm{CP}_{i_h}(e_i)|} a_{i_hp,q} = 1 \quad 1 \leqslant p \leqslant |N_{i_h}(e_i)|$$

$$\sum_{p=1}^{|N_{i_h}(e_i)|} a_{i_hp,q} \leqslant 1 \quad 1 \leqslant q \leqslant |\mathrm{CP}_{i_h}(e_i)|$$

$$a_{i_hp,q} = 0 \quad \text{or} \quad 1$$

This is a linear programming problem. Due to the special characteristics of the constrains, it is much better to solve using linear assignment or min-cost matching in bipartite graphs.

3.3 Computing each cost

3.3.1 Connection cost

Our algorithm adopts the method in Ref. [2] to classify the nets into four categories, as illustrated in Fig. 1.

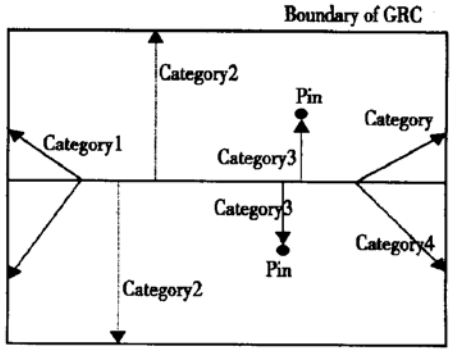

Fig. 1　Four categories of cross points

(1) category 1: The net has a cross point on the left boundary of the GRC.

(2) category 2: The net has a cross point on the upper boundary of the GRC.

(3) category 3: The net has a pin inside the GRC.

(4) category 4: The net has a cross point on the right boundary of the GRC.

The connection cost of the net is written as

$$\text{conn}(c_{jk}) = |c_{jk} - \text{pd}_{jk}|_x + |c_{jk} - \text{pu}_{jk}|_x \quad (4)$$

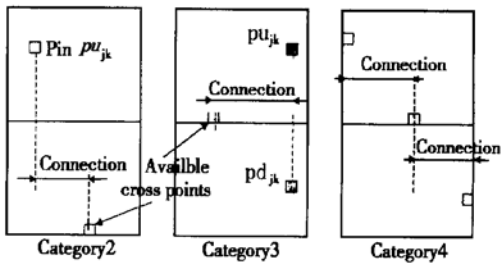

Fig. 2　Pin and caculation of connection cost

which represents the horizontal distance between the cross point and the upper and lower terminals. Different types of cross points have different pd_{jk} and pu_{jk}.

3.3.2 Computing and updating displacement cost

Let $c_{j,k+1}$ be the cross point for the net on the next GRC boundary. The displacement cost could

be written as

$$\mathrm{dist}(c_{jk}, c_{j,k+1}) = |c_{jk} - c_{j,k+1}|_x \qquad (5)$$

Before doing the linear assignment for each GRC boundary, we search in the previous GRC boundary for the nets of category 2, compute the displacement cost for each cross point available, and combine the result into the corresponding connection cost. At the same time we project the obstacles onto the top and bottom boundary, and introduce a penalty term for them in the displacement cost. In our algorithm, a constant value is added according to the position of the projection. The displacement cost and the obstacles are illustrated in Fig. 3.

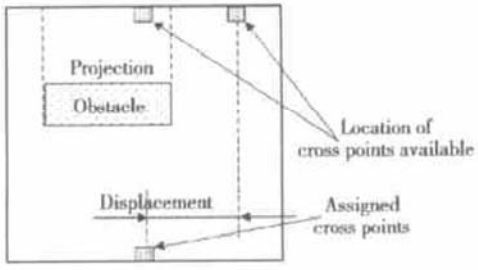

Fig. 3 Displacement cost

3.3.3 Computing and updating exclusiveness cost

Priority queue is used to deal with the congestion of the cross points. A higher priority is assigned to a longer net. CPA is performed first for those nets with higher priority. Those cross points that have not been assigned are passed on to the next stage. When doing CPA for higher priority level, the exclusiveness cost could not be computed because it is impossible to predict the impact of cross points for lower priority nets on the current priority level. To estimate that, a heuristic method was adopted in our algorithm, as shown in Fig. 4.

First, we find in the GRC the pins of those nets with a lower priority level, and project them onto the boundaries. The projections are called reserved cross points for other nets. Later, the algorithm uses these reserved cross points to compute the exclusiveness cost at the current priority level, based on the current exclusiveness matrix. Different from the actual cross points, the diagonal element in the exclusiveness matrix G_{hi} corresponding to the reserved cross points is non-zero, which means the impact on the nets with lower priority if it is occupied. The reserved cross points may still be assigned in the CPA. As a penalty, the non-zero element on the diagonal of G_{hi} tries to prevent the algorithm from assigning the reserved cross points. The above method gives a rough estimation of the cost.

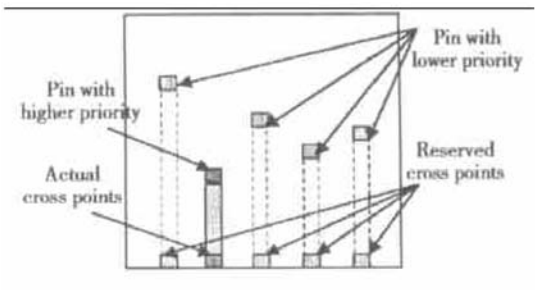

Fig. 4 Reserved and actual cross points

For each following priority queue, G_{hi} is computed before CPA is performed, based on the assignment result of the previous stage. G_{hi} reflects the congestion of the cross points on the current GRC's boundary, which is dependent only on the assigned cross points on the boundary, not on the nets. Then the exclusiveness cost is combined into the cost matrix of all crossing nets on the current GRC boundary. In our algorithm, the cost increased at each cross point for each net is the same.

4 Complexity analysis

Let m be the number of the priority queues generated by the algorithm, k be the number of GRC boundaries in a col, n be the number of available cross points on a boundary, p be the average number of crossing nets, l be the exclusive interval. We have:

The complexity of performing linear assignment on each GRC boundary is $O(n^3)$;

The complexity of updating the displacement cost on each GRC boundary is $O(np)$;

The complexity of CPA for each priority queue is $O(kn^3)$, updating the congestion cost be-

tween different priority queues is $O(knl)$;

So, the complexity of the algorithm is $O(mkn^3)$. If we could implement it in parallel, the algorithm could run even faster.

5 Experimental results

The algorithm is implemented in C language on SUN Enterprise E450. Below is a comparison of the results derived from LCPA in this paper and the CPA algorithms in Ref. [3]. The testing data is provided by MCNC, whose parameters are listed in Table 1.

Table 1 Parameters of the testing circuits

Bench mark	Number of nets	Number of GRCs
C2	745	9×11
C5	1764	16×18
C7	2356	16×18
Avq	21851	65×68

We input the results of LCPA and CPA to our gridless router, and the routing results are listed in Table 2 and Table 3.

Table 2 Gridless routing result after CPA

Bench mark	Completion rate/%	t/s	Number of vias	Total wire length
C2	99	88	8709	464325
C5	100	235	23085	1265757
C7	99	571	30087	1981404
Avq	99	2837	265100	9434387

Table 3 Gridless routing result after LCPA

Bench mark	Completion rate/%	t/s	Number of vias	Total wire length
C2	100	43	7846	457012
C5	100	136	20290	1247697
C7	100	161	25819	1953556
Avq	99	1612	224783	9254748

From the gridless routing results, we can see that the completion rate was improved, and the total wire length was reduced. The number of vias dropped significantly by 10% to 15%, and the detailed routing time was shortened by approximately 50%.

Table 4 gives the comparison in the running speed of CPA and LCPA.

Table 4 Comparison in the running time

Bench mark	t/s	
	CPA	LCPA
C2	1.800	1.890
C5	4.680	4.780
C7	6.840	6.950
Avq	37.800	38.750

From Table 4, we can see that though there is an update on the displacement cost for neighboring GRC boundary in each iteration in the algorithm of LCPA, the running time of LCPA is almost equal to that of CPA.

6 Conclusions

In this paper, we propose a cross point assignment algorithm under consideration of very long nets (LCPA). Our experiment results show that LCPA can obviously reduce the difficulty of the following detailed routing and improve the routing quality with little cost of running time compared with previous CPA algorithm. In the future work, we plan to consider crosstalk during cross point assignment.

References

[1] Li Jiang, Hong Xianlong, Qiao Changge, et al. Cross point assignment algorithm based on the analyse of net type. Chinese Journal of Semiconductors, 1997, 18: 609(in Chinese) [李江, 洪先龙, 乔长阁, 等. 基于线网类型分析的过点分配算法. 半导体学报, 1997, 18: 609]

[2] Hong X L, Huang J, Cheng C K, et al. FARM: an efficient feed-through pin assignment algorithm. Proc 29th ACM/IEEE Design Automation Conference, 1992: 53

[3] Huang Songjue, Hong Xianlong, Cai Yici, et al. Parallel cross point assignment algorithm with nets priority. Microelectronics, 2000, 30: 28(in Chinese) [黄松珏, 洪先龙, 蔡懿慈, 等. 带线网优先级分类的并行过点分配算法. 微电子学, 2000, 30: 28]

[4] Kong Tianming, Hong Xianlong, Qiao Changge. VEAP: efficient VLSI placement algori-thm based on global optimization. Chinese Journal of Semiconductors, 1997, 18: 692 (in Chinese) [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: 692]

[5] Kong Tianming, Hong Xianlong, Qiao Changge. POTIF: efficient power and timing Driven placement algorithm. Chinese Journal of Semiconductors, 1998, 19: 54(in Chinese) [孔天明, 洪先龙, 乔长阁. 功耗和时延双重驱动的 VLSI 布局算法. 半导体学报, 1998, 19: 54]

[6] Zhou Feng, Tong Jiarong, Tang Pushan. Routing algorithm for FPGA with time cons-traints. Chinese Journal of Semiconductors, 1999, 20: 831(in Chinese) [周峰, 童家榕, 唐璞山. 带时延驱动的 FPGA 布线算法. 半导体学报, 1999, 20: 831]

[7] Yao Bo, Hou Wenting, Hong Xianlong, et al. Fame: a fast detailed placement algorithm for standard cell layout based on mixed Mincut and enumeration. Chinese Journal of Semiconductors, 2000, 21: 744

[8] Yu Hong, Hong Xianlong, Yao Bo, et al. A new timing-driven placement algorithm based on table-lookup delay model. Chinese Journal of Semiconductors, 2001, 21: 1129

[9] Yu Hong, Yao Bo, Hong Xianlong, et al. ECOP: a row-partition based incremental placement algorithm for standard cell layout design. Chinese Journal of Semiconductors, 2001, 22: 96(in Chinese) [于泓, 姚波, 洪先龙, 等. ECOP: 一种基于单元行划分的标准单元模式增量布局算法. 半导体学报, 2001, 22: 96]

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

长线网预处理的过点分配算法*

张轶谦 谢 民 洪先龙 蔡懿慈

(清华大学计算机科学与技术系, 北京 100084)

摘要: 提出了一个长线网预处理的过点分配算法. 该算法不仅考虑了过点和物理连接端的连接费用、总体布线单元边界上不同过点之间的互斥费用, 而且考虑了同一线网不同过点之间的错位费用. 实验结果表明, 该算法极大地提高了详细布线阶段的布线质量和速度, 特别是对于长线网而言, 效果更为显著.

关键词: 过点分配; 布图; 超大规模集成电路

EEACC: 2570 **CCACC**: 7410D

中图分类号: TN47 **文献标识码**: A **文章编号**: 0253-4177(2002)06-0582-07

* 国家自然科学基金资助项目(批准号: 60167016)

张轶谦 女, 1977 年出生, 博士研究生, 主要从事超大规模集成电路布图算法的研究.

2001-10-24 收到, 2001-12-04 定稿