第11卷 第2期　　半 导 体 学 报　　Vol. 11, No. 2
1990 年 2 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Feb., 1990

改进的"带宽最小化" BBL 布局算法

朱家璧　陈允康*
(清华大学电机工程系，北京)

1989 年 1 月 2 日收到

在本文中，作者把"带宽最小化"的布局算法改进为适于多尺寸模块模型．在这种模型中，可以为每个模块预先设计出多种不同长宽比的设计方案．文中给出了一个求多尺寸带宽的线性复杂度的算法．通过采用分级式布局，使得算法兼顾了几何设计与连线的优化．该算法按照自顶向下与自底向上相结合的顺序对布局进行优化，既考虑到了局部特点又使得总体规划做得很好，从而大大地提高了优化能力．

主题词　积木块模式布图，布局，带宽最小化布局算法，多尺寸模块模型，分级式

一、导　引

积木块模式布图（BBL）的自动布局以及 Floor-planning 属于困难问题．这些技术目前还尚未应用于工业实际中．在以前发表的文章中，一些人使用"硬模块模型"，在这种模型中模块的尺寸是固定的；而另一些人则使用"软模块模型"，在这种模型中，模块的尺寸可取连续可变的不同数值，但模块的面积是保持不变的．参见图 1(a), (b).

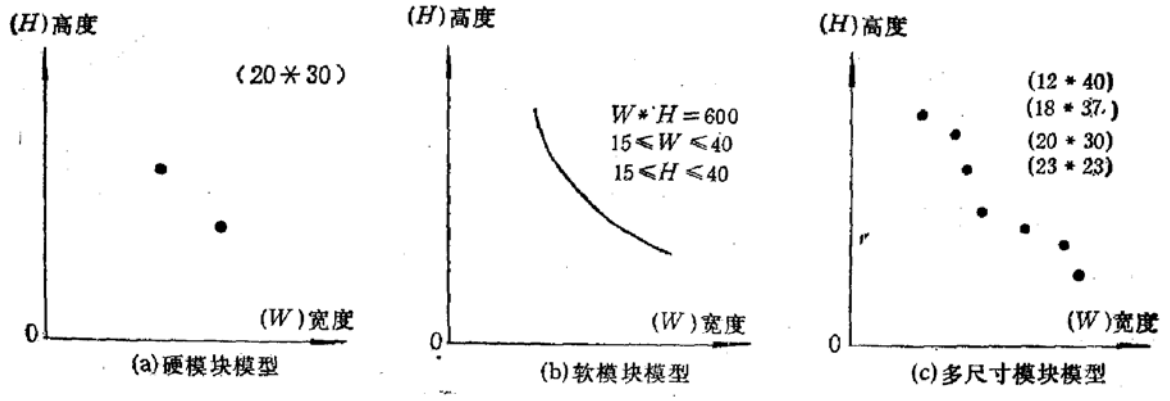


图 1　三种模块模型

许多人认为，在硬模块模型中，由于对模块的设计是在芯片设计之前进行的，所设计出的模块不能符合芯片优化的要求．因此，硬模块模型会导致芯片中产生许多空闲空间以及较长的连线．与此相反，在软模块模型中，在 Floor-planning 完成之后才进行模块

* IEEE 会员．

设计。当进行 Floor-planning 的时候，模块的尺寸被认为是在一定的区域内，沿着 $w \times h = a$ 曲线连续地变化(其中 a 是模块的面积，它是个常量)，参见图 1(b)。通过调整模块的长宽比，就可以得到一个空闲空间为零或者最小化的布图[1][2]。但是，这种模型仍然不符合实际的 IC 设计过程。因为当进行模块设计时，模块的尺寸一般来说并不能与所要求的尺寸相一致，因此有些模块将会相互重迭，同时还会出现空闲空间。

采用"多尺寸模块模型"是一种很好的选择。在这种模型中，可以为每个模块预先设计出多种不同长宽比的设计方案，参见图 1(c)。其中与各个不同尺寸相对应的模块面积可以不同。这种模型更加接近实际的 IC 设计过程。但是，对于这种模型的处理要比前面两种模型更为复杂。

在本文中，作者在两个方面对"带宽最小化布局算法"[3]进行了改进。首先，把它改进为适于多尺寸模块模型，并且能够以线性复杂度求出"多尺寸带宽"；其次，在分级结构上实现这个算法，并且通过采取特殊的优化策略改进它的效率。

二、多尺寸带宽

为了能更清楚地讨论"多尺寸带宽"，需要首先对于在文献[3]中所提出的"带宽"以及"带宽最小化布局算法"作一下简要的回顾。

定义 1 (带宽):

给定 n 个整数对 $(a_1, b_1), (a_2, b_2), \cdots\cdots (a_n, b_n)$。设区间 $[L, R]$ 至少包含了每个数对中的一个分量。把区间 $[L, R]$ 的最小长度定义为所给的整数对的带宽。

参见图 2。

这里的整数对表示单元 (cell) 的两个边长。一个单元可以是一个块 (block)、一组合并的块、或者一个被切开的部份。这里的块表示了带有走线区域的一个模块、或者已经在这个块中布局了的一组高连接度的模块——即"群"。图 2(a) 中示出了几个单元。把这些单元的尺寸点 (即整数对的分量) 排列在数轴上。如图 2(b)。其中，"□"符号表示 2(a) 中单元的尺寸点，"□"中的数字表示其所属单元的标志号。在图 2 的例子中，带宽为 6。这个带宽的意义为：把所有模块排成一列时，最高模块与最矮模块之间的最小高度差。参见图 2(c)。

"带宽最小化布局算法"的主要思想是，通过对块进行旋转、合并以及切割而达到带宽的最小化，然后通过一个贪心 (greedy) 算法，把这些单元划分成若干行，并按照所求得的方位 (pattern) 把它们一行行地放入芯片中。

这个算法对于一些规模较小的例子来说是一种比较有效的初始布局方法。它采取了硬模块模型。然而，这个算法只考虑了芯片的面积。因此，行结构使得它在改进诸如连线长度这类布图性质方面显得比较困难。另外，当芯片上的模块数量太多时，将导致较大的空闲空间。

本文所提出的算法成功地克服了以上的不足之处。图 3 给出了本算法的流程。

假设对于模块 M_i，有 K_i 个可选尺寸。一组整数对 $\{(a_i^1, b_i^1), (a_i^2, b_i^2), \cdots, (a_i^{K_i}, b_i^{K_i})\}$ 表示与模块 M_i 相对应的单元的尺寸。其中 $1 \leqslant i \leqslant n$，$K_i$ 是 M_i 的尺寸个数。多

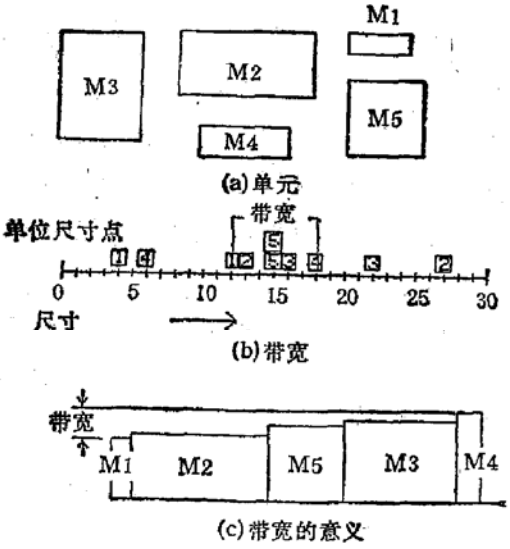


图 2 带宽

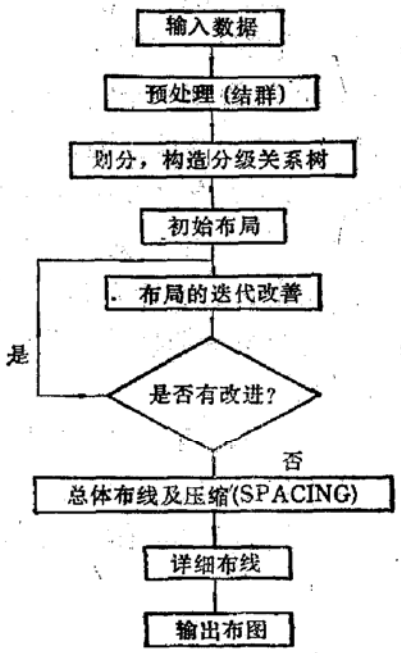


图 3 算法流程图

尺寸带宽推广如下:

定义 2(多尺寸带宽):

给定 n 组整数 $\{a_1^1, b_1^1, a_1^2, b_1^2, \cdots\cdots, a_1^{K_1}, b_1^{K_1}\}$, $\{a_2^1, b_2^1, a_2^2, b_2^2, \cdots, a_2^{K_2}, b_2^{K_2}\}$, $\cdots$, $\{a_n^1, b_n^1, a_n^2, b_n^2, \cdots\cdots, a_n^{K_n}, b_n^{K_n}\}$. 设区间 $[L, R]$ 至少包括了每组整数中的一个分量. 把区间 $[L, R]$ 的最小长度定义为所给整数组的多尺寸带宽.

求一组单元尺寸的多尺寸带宽只需 $O(m)$ 计算复杂度. 其中 m 为所有的单元尺寸数目之和,即:

$$m = \sum_{i=1}^{n} K_i$$

下面给出一个求多尺寸带宽计算复杂度为 $O(m)$ 的算法:

求多尺寸带宽的算法

(1) 初始化数组 cell_points_num[i] = 0. 其中, cell_points_num[i] 表示在当前区间 [left_pointer, right_pointer] 中第 i 个单元尺寸点的个数. $i = 1, 2, \cdots, n$.

(2) 把所有的尺寸点按照单调递增顺序排序.

(3) 使 left_pointer 指向最小尺寸点. 计数器 CIM 指示着那些至少有一个尺寸点在当前区间中的单元的个数. 把 CIM 置为 0. 设置 current_bandwidth = $+\infty$.

(4) **FOR** (right_point = 最小尺寸点 **TO** 最大尺寸点){

设 $i \Leftarrow$ 由 right_pointer 所指尺寸点的单元标号;

IF (cell_points_num[i] = 0) **THEN** CIM++;

cell_points_num[i]++;

WHILE (cell_points_num [由 left_pointer 所指的尺寸点的单元标号] > 1 **AND** left_pointer ≠ right_pointer) **DO**{

```
    cell_points_num [由 left_pointer 所指尺寸点的单元号]++;
    移动 left_pointer，使它指向下一个较大的尺寸点;
    }
  IF (CIM = 单元的个数 AND 当前带宽长度 > 当前区间长度) THEN {
    当前带宽⇐当前区间长度;
    记录下当前区间;
    }
  }#
```

在第(4)步之后，当前带宽即所求带宽，而与落入相应区间里的尺寸点所对应的单元尺寸即为候选尺寸．如果一个单元有多个候选尺寸，则应选择另一个边最短的尺寸．注意到多尺寸模型模块长宽关系是离散严格单调的，见图 1(c)，可知对算法进行上述推广是合理的．

与"带宽最小化布局算法"一样，通过对块进行旋转、合并以及切割，而使多尺寸带宽最小化．另外，当把单元(模块)一行一行地放进芯片中以后，还要对模块的尺寸和方位重新进行调整，以使得芯片面积最小化．

三、分级式多尺寸带宽最小化布局算法及其优化过程

我们通过分级方法，使得在芯片面积优化与连线总长优化之间得到折中．

考虑到有些高连接模块，例如总线线网，在程序开始要进行"结群"预处理．

在布局开始之前，按照自顶向下的顺序对模块进行划分，构造出拓扑布局关系树．在划分过程中，优化目标为(1)使属于不同部份的模块的连线最小化；(2) 使在同一级中各个部份的模块总面积、预估尺寸尽可能符合要求．划分数一般为 4，也可以为 6、8 或 9，这由一个启发式算法确定，使得划分结果尽量接近优化目标．当经过划分后，如果某一部份中的模块个数不少于 12，则继续进行下一级划分．图 4 示出了图 6 布局的分级关系树．

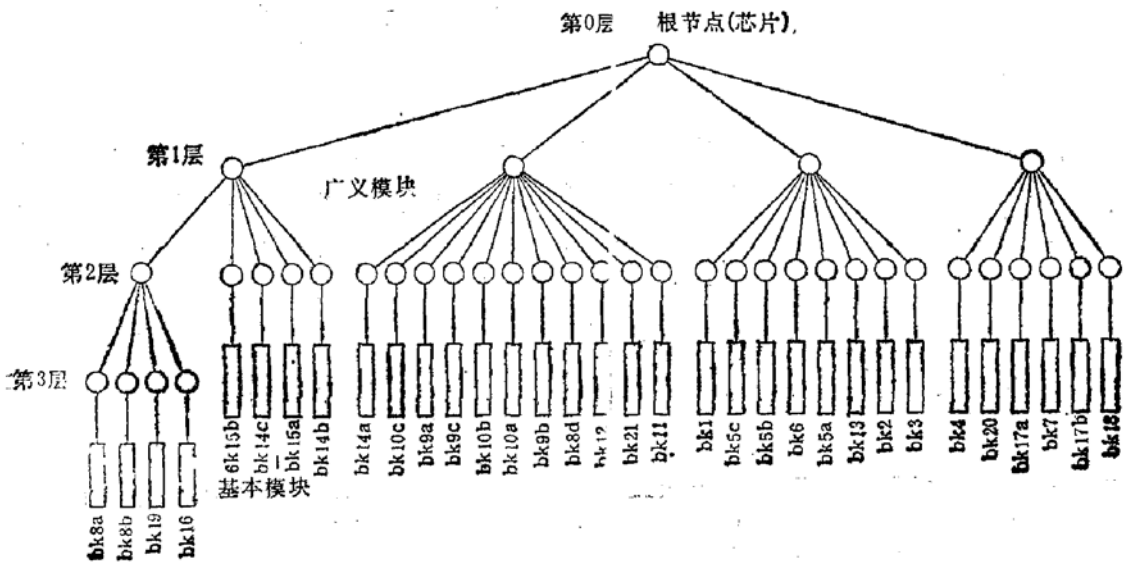


图 4　模块的分级关系树

其中，广义模块为(基本)模块或广义模块的父模块．广义模块下面亦简称为模块．

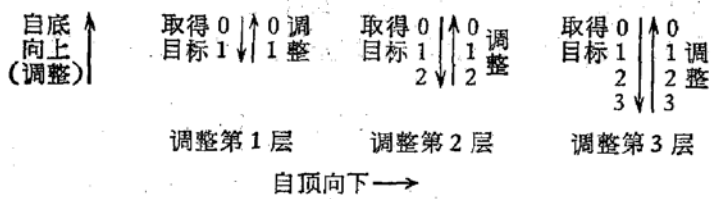
初始布局确定了模块在其父模块中的位置．父模块的尺寸应尽可能符合预定的尺寸．初始布局使用了自底向上的优化顺序．

在布局的迭代改善与优化阶段中，调整模块在其父模块中的相对位置，使所有加权连线总长最小化，并且调整各级模块的尺寸，使芯片面积最小化．这里采取了一种特殊的优化顺序，下面将作解释．

所有的分级式布局算法可以根据其优化顺序分为两大类：自顶向下以及自底向上．按自顶向下的顺序进行优化，可以把总体规划做得很周到，但各部份的细节就很难考虑．与此相反，按自底向上的顺序进行优化，则可以照顾到细节，但很难做好总体规划．局部利益往往同总体优化目标相悖．因此，只按一种顺序进行的优化是十分有限的，而同时按照自顶向下和自底向上两种顺序进行的优化则可以取二者之长．

在本文的算法中就采取了这种策略，但如何在一个算法中结合这两种优化顺序这个方面上与以前发表的文章有所不同．

对布局所进行的改善调整包括旋转与/或翻转模块、对模块在其父模块中的位置进行重新布局、以及重新选择模块尺寸，使得芯片面积以及连线总长最小化．在对某一级模块的布局进行调整之前，改进目标（如模块的长宽尺寸）是从由关系树的根（root）到与所要调整的模块相对应的节点这条路径上所得到的．根据这些目标，对这级布局进行调整．然后，在被调整过的那一级模块的祖先模块的布局也需要进行调整，以便消除布线区域拥挤以及缩小芯片面积．因此，调整就象冒气泡一样是自底向上的．为了避免被与整体优化目标相矛盾的局部最优所限制，应该自顶向下一级一级地调整布局．也就是说，对较高级上布局的调整要先于较低级．对于在图 4 出示的例子，布局调整是按照这样的分级顺序：取得目标 0 → 1，调整 1 → 0；取得目标 0 → 1 → 2，调整 2 → 1 → 0；取得目标 0 → 1 → 2 → 3，调整 3 → 2 → 1 → 0．这个顺序可写为：



通过这种办法，把自顶向下与自底向上的优化相互结合起来．这部份算法主要由 ITERAT_IMPROVE( )、GLOBAL_ADJUST( )、WFS_GLOBAL_ADJUST( )、WFS_LOCAL_ADJUST() 四个子程序实现．它们的基本思想描述如下：

```
ITERAT_IMPROVE()/*布局的迭代改善主程序 */
{FOR level = 1 TO 关系树的分级数
CALL GLOBAL_ADJUST(level);
}
GLOBAL_ADJUST(level) /*对 level 级模块进行总体调整并且对上层进行局部调整*/
{CALL WFS_GLOBAL_ADJUST(root, level);
  FOR i = level TO 0, STEP = (-1)
```

```
    CALL WFS_LOCAL_ADJUST (root, i);
}

WFS_GLOBAL_ADJUST(node, level)  /*对 level 级模块进行总体调整.*/
{IF (node 节点为第 level 级) THEN{
    IF (node 为基本模块) THEN
      重新调整该模块的尺寸,使它适应所给定的目标;
    ELSE
      对 node 的儿子模块进行总体调整，对行结构的排列顺序以及模块在行内的排
      列顺序进行调整,使连线总长最短;
  试翻、旋转 node 对应的模块,使连线总长最短;
  }
ELSE IF (node 的分级号 < level AND node 不是基本模块) THEN
    用启发式算法，对 node 模块内处于关键路径上的有希望对长宽比进行调整的几
    个儿子模块节点 {node_son}，给出调整目标，调用 WFS_GLOBAL_ADJUST
    (nodel, level)，其中 nodel∈{node_son};

WFS_LOCAL_ADJUST (node,level)  /*对 level 级模块进行局部调整. */
{IF (node 节点的分级号 < level AND node 不对应基本模块) THEN
  对 node 的所有儿子节点集 {node_son} 中的元素 nodel∈{node_son} 调用 WFS_
  LOCAL_ADJUST (nodel, level);
ELSE IF (node 节点的分级号 = level AND node 不对应基本模块) THEN
  对 node 的儿子模块进行局部调整,保持布局拓扑结构不变,使模块间的通道空间适
  于预估布线所需的空间;
```

四、运行结果

本算法是在 Micro VAX-II 计算机的 Ultrix-32m 操作系统环境下用 C 语言开发的.为了能清楚地比较最终芯片布图结果,我们使用了总体布线和压缩程序以及 BBL2[4][5] 的详细布线程序. 总体布线和压缩部份的作用是对布局进行局部调整，使之与总体布线匹配. 有关这一部份的内容将另文发表. 这里所给出的两个例子是在没有任何人为干预的条件下运行的结果. 实验结果是令人满意的.

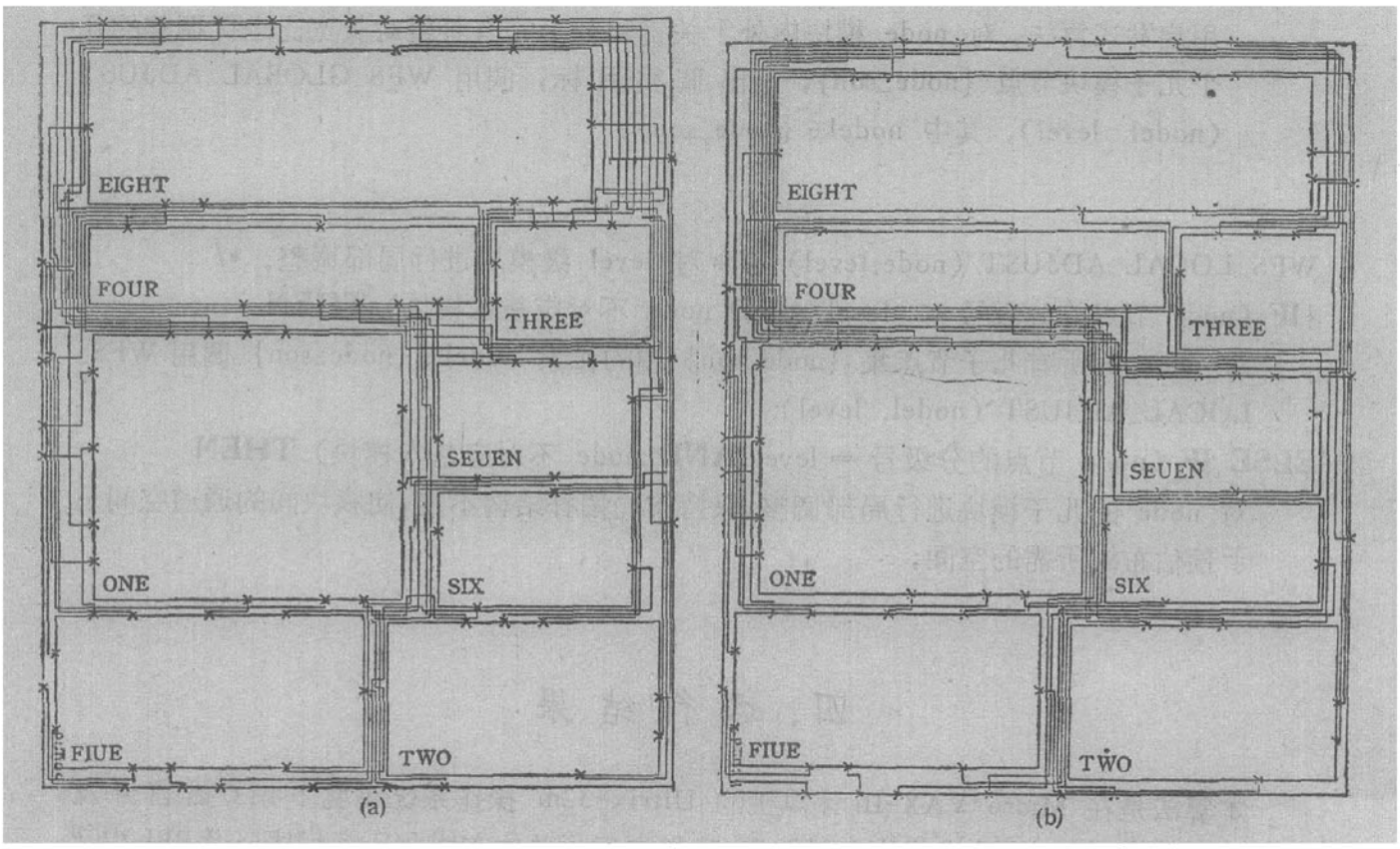
图 5 所给出的第一个例子中有 8 个模块及 38 个线网. 表 1 中列出了模块的尺寸,一个模块可以有多种可选尺寸. 图 5(a)显示了硬模块模型的情况,布图尺寸为 163 × 201. 图 5(b) 显示的是多尺寸模块模型的情况,布图尺寸为 164 × 196,比硬模块模型小 2%.

第二个例子是选自[3],其中所有模块的尺寸都是固定的. 这是多尺寸模块模型的一个特例. 在这个例子中,有 33 个模块和 132 个线网. 我们的布图结果如图 6 所示. 它的

表 1 例一中模块的尺寸

模块名	(a) 硬模块模型的模块尺寸	(b) 多尺寸模块模型的模块尺寸			
		1	2	3	4
ONE	(80*70)	(85*65)ⓐ	(80*60)		
TWO	(70*40)	(90*35)	(70*40)ⓐ	(62*52)	
THREE	(40*30)	(40*30)ⓐ			
FOUR	(100*20)	(100*20)ⓐ			
FIVE	(80*40)	(80*40)ⓐ	(60*54)		
SIX	(50*30)	(55*40)	(50*30)	(56*26)ⓐ	
EIGHT	(130*40)	(140*35)ⓐ	(130*40)	(115*47)	(100*60)

说明：标号ⓐ表示该尺寸被选用.



(a) 硬模块模型的结果 (b) 多尺寸模块模型的结果

图 5 例一运行结果

面积是 226 × 198，比[3]的结果小了 6.5%，这是由于我们算法的分级式布局结构及其特殊的优化过程. 这两个例子布局所用的 CPU 时间分别为 3 秒和 51 秒（包括数据输入/输出时间）.

作者在完成本文的过程中，曾多次与清华大学计算机系洪先龙教授以及电机系肖达川教授讨论，得到了他们的热情帮助. 对此，我们表示衷心的感谢.

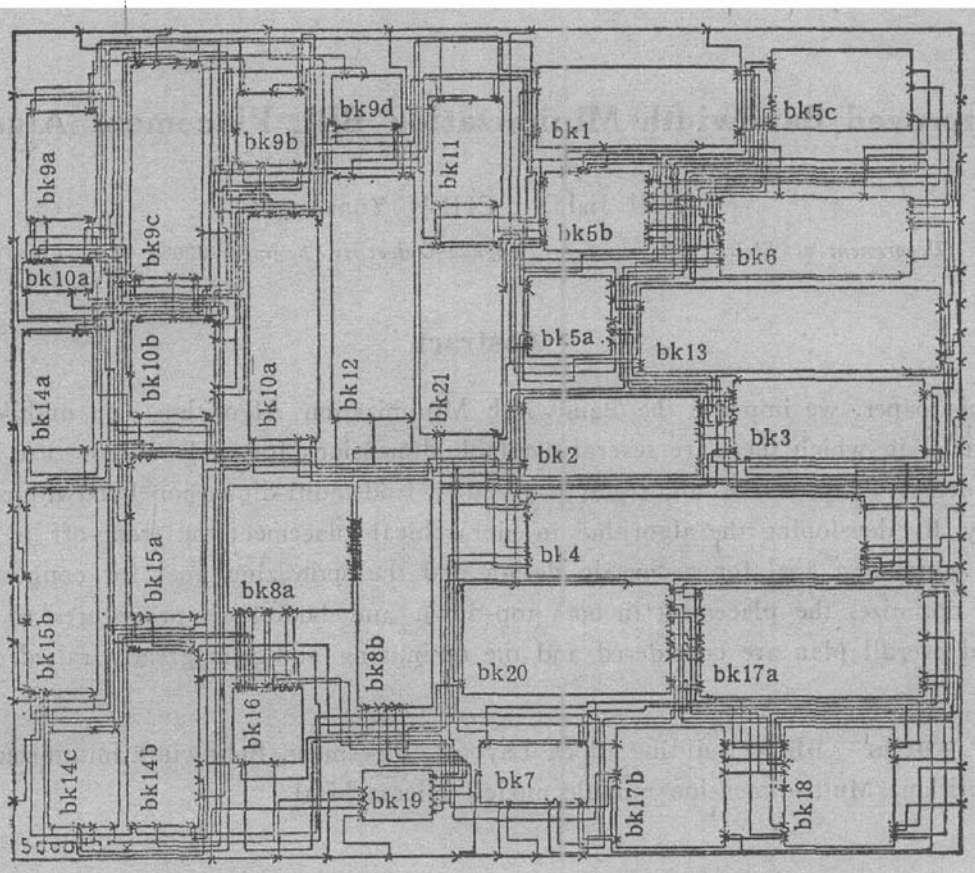


图 6　例二运行结果

参　考　文　献

[1] Shmuel Wimer, *et al.*, *IEEE Trans. on Circuits and System.*, 35, 267(1988).

[2] Ralph H. J. M. Otten, *et al.*, Proc. of ICCAD'82, 499(1983).

[3] E. S. Kuh, C. C. Chen, Proc. of ICCAD'84, 90(1984)

[4] N. P. Chen, *et al.*, BBL. 2 User's Manual, Electronics Research Lab., College of Engineering, University of California, Berkeley (1984).

[5] N. P. Chen, *et al.*, Proc. of Int. Conf. on VLSI'83, 37(1983).

An Improved Bandwidth Minimization BBL Placement Algorithm

ZHU Jiabi, CHEN Yunkang

(Department of Electric Engineering, Tsinghua University, Beijing, 100084 *P. R. China)*

Abstract

In this paper, we improve the Bandwidth Minimization Algorithm on multi-dimension module model in which there are several available dimensions for each module that is predesigned in various aspects. The algorithm is given to find multi-dimension bandwidths in linear complexity. By developing the algorithm in hierarchical placement the trade-off is made between the optimizing goal for geometric design and the optimizing goal for connections. The algorithm optimizes the placement in both top-down and bottom-up procedures so that both details and overall plan are considered and the optimizing ability is greatly raised.

Key words BBL (Building Block Layout), Placement, Bandwidth minimization placement algorithm, Multi-dimension module model, Hierarchical

* Member, IEEE