第18卷第9期 半 导 体 学 报 Vol. 18, No. 9
1997年9月 CHINESE JOURNAL OF SEMICONDUCTORS Sep., 1997

VEAP：基于全局优化的有效VLSI布局算法*

孔天明 洪先龙 乔长阁

（清华大学计算机科学与技术系 北京 100084）

摘要 本文中，针对规则的VLSI设计模式（门阵列，标准单元等），我们提出一种新的非常简单有效的布局算法. 该算法基于严格的数学分析，可以证明能够找到全局最优解. 在实验中发现，对于很大规模的电路，我们的算法比现有的所有算法都快. 此外，我们的算法还能够同时适应于线长优化和时延优化模式.

EEACC：2570； **CCACC**：7410D

1 介绍

设计工具的质量和速度对人们接受规则的设计模式有着巨大影响. 在本文中，我们提出一个新的布局算法VEAP，它已经成功应用到各种规则设计模式的布图设计中，尤其是针对大规模的电路.

作为物理设计过程的第一步，布局阶段的任务是计算单元的位置. 由于布局结果决定了电路的最小可实现面积和线长，它对生产成品率和电路性能有着重要的影响. 因此，一个好的布局工具必须保证在最小或给定的面积内成功布通，而且必须能够处理极大规模的电路设计. 布局问题的难度随着单元数目的增加而增加. 因此，处理VLSI布局问题的经典方法是基于分而治之的策略，其典型代表是基于最小割的图划分算法（如文献[1～4]）. 但是，如Kernighan，Lin[5]和 Fiduccia，Mattheyses[6]等的最小割算法是依赖于初始划分的迭代改进启发式算法. Ng 等人[7]指出：要获得较好的解，有必要从许多随机产生的初始划分中选取一个. 他们提出了一个结群算法，构造一个收缩的网络，最小割算法在此网络上进行划分；通过这种方式，他们得到了较好的结果. Suaris 和 Kedem[4]将 Fiduccia-Mattheyses 的二等分算法扩展到四等分，并应用到标准单元布局，给出了更强的结果. 随后，许多算法将布局问题归结为线性或非线性的连续规划问题. 与基于最小割的算法不同，这种算法可以直接利用单元

* 国家攻关经费及高等学校博士学科点专项科研基金资助课题
孔天明 男，1971年生，博士生，从事专业：计算机辅助集成电路布图设计
洪先龙 男，1940年生，博士导师，从事专业：计算机辅助集成电路布图设计
乔长阁 男，1967年生，博士，从事专业：计算机辅助集成电路布图设计
1996年7月22日收到初稿，1996年11月18日收到修改稿

和芯片的几何大小以及引脚位置等几何信息；而且，一般地，这种算法不需要初始解，所有单元可以同时处理. 这些方法包括物理（力或电网络）模拟（文献[8～12]）以及特征根方法（文献[13～15]）等. 其中一些方法也采用了划分，从而递归产生一个一个的子问题；但他们将同时优化局限于第一步，也就是说，每个子问题都是独立求解的.

局限于局部最优点是基于划分方法的主要缺点. 已有一些工作来处理这个问题，尤其是针对广泛使用的最小割算法. 例如：Lauther[1]引进了端点传播（terminal propagation）；Dunlop 和 Kernighan[2]考虑连接不同区域单元的线网. 当连续优化应用到越来越小的子问题时，这种全局连接问题越来越突出.

现在有一些基于二次规划的算法，如：GORDIAN([16]) 和 Ritual/Tiger 就是两个著名的例子. 这类算法有一个非常好的特性：可以基于严格的数学分析证明它们的求解质量. 但现在这种算法的运行时间仍然较长. Ritual/Tiger 的开发者从来没有给出过极大规模的电路的测试结果. 在 GORDIAN 系统中，布局问题被归结为一个线性约束的二次规划问题：线性约束的存在是由于单元必须无重叠地均匀分布在芯片平面上. 在 GORDIAN 系统中，线性约束是直接求解的. 事实上，这意味着单元必须位于它被分配到的区域而不能移动到该区域以外；这一点过分限制了求解空间，因而严重影响了布局质量.

此外，在过去还发表了很多基于模拟退火或遗传算法等随机优化方法的算法. 一般来说，这类算法可以得到非常好的布局质量，但必须运行相当长的时间以后才能保证布局质量，因而它们不大可能成为有实用价值的算法. 这类算法的最新进展来自 W. J. Sun 和 C. Sechen[17,18]，他们提出了一个基于模拟退火的算法，速度比以前的同类算法快得多. 但从本文中可以看到，无须借助于随机优化的技术，在甚至更短的时间内获得相同或更佳的布局结果依然是可能的.

本文中，我们提出了一个基于二次规划的算法，但与其它算法不同的是，1）我们并不直接求解单元分布约束（即：单元必须均匀无重叠地分布），也就是说，单元可以从分配到它的单元移动到其它的单元，从而避免了过分限制布局质量；2）我们的算法具有非常简单的迭代表示，而且不需要任何附加的存储量，因而具有极高的执行效率；3）我们的算法是内在可并行的，从而为进一步的提高速度提供了可能性.

本文中我们首先给出了布局过程的概述、算法讨论，并对算法复杂度进行了分析；然后，我们给出了最终布局的算法描述；实验结果中最大的测试电路达到了21000个单元以上的规模. 最后，我们给出结论并讨论进一步工作的方向.

2　布局过程概述

VEAP 的布局过程由全局优化和模糊划分交替组成，二者相互作用，相互影响.

VEAP 的输入包括网表，单元库，芯片的几何定义. 每个单元的大小（长度和宽度）以及引脚位置都可以从单元库中得到. 网表描述了单元之间的互连关系；从网表中，我们可以知道一条线网 n 连接哪些单元，单元 c 连接哪些线网. 我们用 C_n 表示线网 n 连接的单元集合，N_c 表示单元 c 连接的线网集合. 对于标准单元设计模式，芯片的几何定义包括了所用的单元行的数目；对于门阵列设计模式，则给出了单元行的大小尺寸. 此外，在布局过程中，我们假定所有的压焊引脚单元的位置是固定的；在实际的设计中，这些单元的位置往往是由外部

系统的需要决定的.

VEAP 的主循环是由全局优化和划分步骤的交替执行形成的. 优化的目标是使得总的互连线长最小化,同时使单元均匀分布在芯片平面而不引起单元之间的重叠.

在 VEAP 的第一个循环中,全局优化面对的是整个芯片平面,所有单元都应该均匀分布在该平面内. 这时,我们人为引进一个约束方程:所有单元的"质心"应该落在该区域的中心. 在此约束条件下,全局优化过程尽量减小总的互连线长度. 然后,划分操作进一步将每个区域均匀划分为子区域,每个子区域对应于下次优化过程中的约束方程. 这样,划分操作就形成了一棵 Slicing 树,其中的每个节点对应于一个区域.

以上循环重复执行,直到划分的次数达到了一个预先规定好的常数. 这样,我们就得到了每个单元的总体布局解.

3 全局优化

在每个全局优化阶段,我们将布局问题抽象为一个数学规划问题并解决之,从而得到一个总体布局.

3.1 问题定义

全局优化的目标函数是最小化线网总的二次线长. 对于线网 n,其长度可以由下列方程估计得出:

$$L_n = \sum_{i,j \in C_n} ((x_i + \xi_{i,n} - x_j - \xi_{j,n})^2 + (y_i + \eta_{i,n} - y_j - \eta_{j,n})^2) \tag{1}$$

其中 $(\xi_{i,n}, \eta_{i,n})$是单元与线网 n 连接时的引脚相对于单元中心坐标(x_i, y_i)的坐标;这些坐标值在布局阶段都是常数,可以从单元库中提取得到. 为了简单起见,我们不妨忽略这些坐标值,即令它们为0. 这种简化对算法的一般性不产生影响. 由于每条线网连接单元的数目不同,我们对每条线网 n 分配一个权重 w_n. 因此我们的目标函数就是所有线网的二次线长函数的加权和:

$$\varphi = \sum_{n \in N} L_n w_n \tag{2}$$

在最初的优化层次($l=0$),每个单元都限定在芯片平面上. 在第 l 层,布局平面被均匀划分为4^l 个区域,每个区域含有一个单元子集. 令 R^l 表示区域的集合. 假定 r 是一个区域,我们用 τ_r 表示该区域中所包含的单元集合,用(μ_r, ν_r)表示该区域的中心坐标. 由于任一单元只能存在于某一个区域中,我们用 R_i 表示单元 i 所在的区域. 于是对于任一区域 r,我们对全局优化问题施加如下两个约束方程:

$$\sum_{i \in \tau_r} x_i / |\tau_r| = \mu_r \tag{3}$$

$$\sum_{i \in \tau_r} y_i / |\tau_r| = \nu_r \tag{4}$$

在本文的后面,我们将称呼这类约束为分布约束. 结合目标函数和约束方程,我们就得到了一个线性约束的二次规划问题(LQP).

3.2 求解方法

我们用"拉格朗日松弛法"求解 LQP. 根据拉格朗日松弛法,每个分布约束对应有一个相应的拉格朗日乘子系数 α_r(x 方向)或 β_r(y 方向). 这样,我们得到一个无约束的优化问题

如下：

$$\max_{\alpha_r,\beta_r\geqslant 0}\min_{x,y}\varphi(x,y,\alpha_r,\beta_r)$$

$$\varphi=\sum_{n\in N}L_n w_n+\sum_{r\in R^l}\Big(\sum_{i\in\tau_r}x_i/|\tau_r|-\mu_r\Big)\alpha_r+\sum_{r\in R^l}\Big(\sum_{i\in\tau_r}y_i/|\tau_r|-\nu_i\Big)\beta_r \tag{5}$$

将(1)代入(5)，得到：

$$\varphi=\sum_{n\in N}w_n+\sum_{i,j\in C_n}((x_i-x_j)^2+(y_i-y_j)^2)+\sum_{r\in R^l}\Big(\sum_{i\in\tau_r}x_i/|\tau_r|-\mu_r\Big)\alpha_r+$$
$$\sum_{r\in R^l}\Big(\sum_{i\in\tau_r}y_i/|\tau_r|-\nu_i\Big)\beta_r$$

显然，这是一个凸函数，因而任意局部最优解就是全局最优解. 令$\frac{\partial\varphi}{\partial x_i}=0$，$\frac{\partial\varphi}{\partial y_i}=0$，我们有：

$$\frac{\partial\varphi}{\partial x_i}=\sum_{n\in N_i}w_n\sum_{j\in C_n}2(x_i-x_j)+\alpha_{R_i}/|\tau_{R_i}|=0$$

$$\frac{\partial\varphi}{\partial y_i}=\sum_{n\in N_i}w_n\sum_{j\in C_n}2(y_i-y_j)+\beta_{R_i}/|\tau_{R_i}|=0$$

从而，

$$x_i=\Big(\sum_{n\in N_i}w_n\sum_{j\in C_n}x_j-1/2\times\alpha_{R_i}/|\tau_{R_i}|\Big)/\sum_{n\in N_i}w_n \tag{6}$$

$$y_i=\Big(\sum_{n\in N_i}w_n\sum_{j\in C_n}y_j-1/2\times\beta_{R_i}/|\tau_{R_i}|\Big)/\sum_{n\in N_i}w_n \tag{7}$$

根据以上方程，我们得到图1中的全局优化算法.

1. 设置初始数据：$\alpha_r=0$，$\beta_r=0$
2. count 从1到 K
3. i 从1到单元数
4. 根据(6)(7)或(8)(9)更新 x_i，y_i
5. 更新拉格朗日系数：
6. $\alpha_r+=\sum_{i\in\tau_r}x_i/|\tau_r|-\mu_r$，
   $\beta_r+=\sum_{i\in\tau_r}y_i/|\tau_r|-\nu_r$
7. 如果达到要求的精度，返回；否则转2；

图 1　全局优化算法

4　算法分析

4.1　算法收敛性

算法的收敛性有两重含义：全局算法(拉格朗日松弛法)的收敛性，以及求解拉格朗日子问题的迭代算法的收敛性.

全局算法的收敛性是由原问题的性质决定的. 我们所有的约束都是线性约束，因此，约束形成的解空间是一个凸集. 此外，我们的目标函数也是一个可微的凸函数. 所以，我们的 LQP 问题是一个凸规划问题. 拉格朗日松弛法求解这类问题是可以收敛的.

为清楚迭代算法的收敛性，我们将式(5)中的目标函数用矩阵形式改写如下：

$$\varphi(\boldsymbol{x},\boldsymbol{y})=1/2\boldsymbol{x}^TQ\boldsymbol{x}-b_x^T\boldsymbol{x}+1/2\boldsymbol{y}^TQ\boldsymbol{y}-b_y^T\boldsymbol{y}$$

其中　向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 表示所有可移动单元的坐标. 矩阵 Q 对应于二次线长函数的二次型. 因此，矩阵 Q 是正定的当且仅当所有的可移动单元都直接或间接与某些固定单元相连接. 由于所有的压焊块引脚都是固定的，这个条件对于所有有用的线网都是成立的；因为所有的单元都应该可以从外部世界见到.

于是，我们可以得到全局最优解于：

$$\boldsymbol{x}=Q^{-1}b_x,\boldsymbol{y}=Q^{-1}b_y$$

不幸的是，在绝大多数的情况下，逆矩阵不是稀疏的；而且矩阵求逆是一件极为费时的

运算.因此我们必须求助于迭代型的算法.可以看出:图1中的算法实际上是高斯-塞德尔迭代法.因为矩阵是对称正定的,因而该迭代算法是收敛的.

一般地,超松弛迭代法要快于高斯-塞德尔迭代法.因此,我们给出了根据超松弛迭代法更新 x_i, y_i 的方法,即:

$$x_i^{(k+1)} = (\sum_{n\in N_i} w_n(\sum_{j\in C_n, j<i} x_j^{(k+1)} + \sum_{j\in C_n, j>i} x_j^{(k)}) - 1/2 \times \alpha_{R_i}/|\tau_{R_i}|)/ \sum_{n\in N_i} w_n \times \omega + (1-w) \times x_i^{(k)} \tag{8}$$

$$y_i^{(k+1)} = (\sum_{n\in N_i} w_n(\sum_{j\in C_n, j<i} y_j^{(k+1)} + \sum_{j\in C_n, j>i} y_j^{(k)}) - 1/2 \times \alpha_{R_i}/|\tau_{R_i}|)/ \sum_{n\in N_i} w_n \times \omega + (1-w) \times y_i^{(k)} \tag{9}$$

式中 ω 是松弛因子.

4.2 算法的复杂度

在极大规模的电路布局布线中,算法的复杂度是一个很值得关注的问题.随着问题规模的增加,许多很好的算法都由于过长的运算时间而不再适用了.

4.2.1 时间复杂度

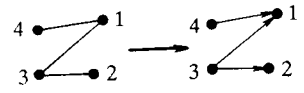
求解拉格朗日子问题的每个迭代步要花费的时间正比于$(m+n+p)$,其中 n, m, p 分别表示可移动单元数,线网数,引脚数.在大规模的电路中,p 和 m 是与 n 成线性增长的.因此,每个迭代步花费的时间为 $O(n)$.注意到我们并不需要找到原问题的精确解,而且,矩阵 Q 是对称正定的,我们的迭代法收敛速度很快.一般只需迭代与 n 无关的常量次数即可.因此,求解每个拉格朗日子问题的时间复杂度为 $O(n)$.求解原问题需要解一系列的拉格朗日子问题.拉格朗日子问题的数目依赖于给定的精度要求.一般来说,我们可以为该数目设置一个上界 $n^{0.5}$.此外,在我们根据单元位置划分区域的时候,需要对单元进行排序,相应的时间复杂度为 $O(n\times\log n)$因此,全局优化算法总的时间复杂度为 $O(n^{1.5}+n\log n)$, 即 $O(n^{1.5})$.

4.2.2 空间复杂度

基本的电路连接关系需要存储空间 $O(m+n+p)$,即 $O(n)$.此外,存储分布约束需要两个长度为 n 的数组.值得注意的是,我们的算法没有任何额外的存储要求了.这样,总的空间复杂度为 $O(n)$.

4.3 算法的高度可并行性

除了超松弛迭代法的加速外,进一步的加速是可能的.我们的算法是内在高度可并行的.从(8)(9),我们可以知道,在一个单元 i 的坐标得到更新之前,所有直接与该单元相连且下标小于 i 的单元的坐标已经更新.我们可以将这种约束用图的形式来表示,从而可以得到计算优先顺序约束图 $G(V,E)$.图中顶点 i 与单元 i 一一对应;边$(i,j)\in E$ 表示:单元 j 直接与该单元 i 相连且下标小于 i,也就是说,单元 j 的更新计算必须在 i 之前进行.例如,图2中电路的计算优先顺序约束图列出在图2右边.由此,基于计算优先顺序约束图,我们可以得到一个非常简单的并行计算方案.例如,我们将电路连接关系图用 p 种颜色着色,从而将单元集合可以划分成 p 个子集.由于在同一子集中的单元没有



电路连接关系图 计算优先顺序约束图

图 2 计算优先顺序约束图的构造

连接关系，它们之间不存在计算优先顺序约束，因而可以独立计算. 由图论知道，p 小于或等于电路连接关系图中顶点的最大度数加一，即一个单元能够连接的最大单元数目加1. 在实际电路，尤其是极大规模的电路中，由于单个单元的大小和引脚数目有限，p 是有限的. 因此，在极大规模的电路中，并行带来的加速将是非常可观的.

5　最终布局

全局优化和划分过程交替执行的结果是一个总体布局. 在这个结果中，存在着许多重叠的单元，因此需要一个最终布局过程. 在这个过程中，我们必须移动单元使得不存在重叠，同时，单元必须依要求的设计模式按行排列.

在标准单元设计中，单元是等高的，但宽度可能差别很大. 芯片面积取决于单元行之间的通道宽度和单元行的长度. 最终布局的目标是在保证均匀分布的线网密度和单元行尽可能等宽的前提下，获得尽可能窄的单元宽度.

我们采用了一种有效的最终布局算法，尽可能地实现线长最小化和单元行等宽. 我们递归地划分芯片平面，直到每个区域只含数十个单元为止，我们根据总体布局的结果确定每个单元应该归属的区域；然后，在每个区域中，将单元分配到区域中特定的位置上，我们称这一过程为“slot 分配”.

我们采用的是一种简化的线性分配算法. 在每个区域，“slot 分配”问题是：给定一些 slot 位置和一个单元集合，寻找一个单元到 slot 位置的映射，使得得到的总线长最小. 假定 slot 数目为 m，待分配的单元数目为 n，将单元 i 分配到 slot j 的费用为 C_{ij}. 一般地，我们有 $n \leqslant m$. 从而，“slot 分配”问题可以写为如下的线性分配问题：

$$\text{minimize} \quad \sum_{ij} C_{ij} Z_{ij}$$

$$\text{subject to}:$$

$$\sum_{j=1}^{m} Z_{ij} = 1, \quad i = 1, \cdots n$$

$$\sum_{i=1}^{n} Z_{ij} \leqslant 1, \quad j = 1, \cdots m$$

$$Z_{ji} \in \{0,1\}, \ i = 1, \cdots, n, \ j = 1, \cdots, m$$

其中 $x_i = \sum_{j=1}^{m} Z_{ij} X_j$，$y_i = \sum_{j=1}^{m} Z_{ij} Y_j$，$(i = 1, \cdots, n)$分别表示单元 i 的位置坐标. 这个问题的求解是非常简单的.

一般的“slot 分配”问题是一个二次分配问题，是很难求解的. 因此，我们对问题进行了一定的简化：在构造费用函数时，简单地忽略同一区域内单元之间的互连，由此单元分配的费用计算可以独立地进行，从而我们可以得到上面讨论的线性分配问题. 这种简化肯定会带来一些误差. 为了消除这些误差，同时不要过分约束单元，我们允许单元移动到指定的区域以外. 这一点是通过在划分区域时，有意地对相邻的区域有半个区域大小的重叠. 而且，我们反复运行“slot 分配”算法多次，以消除费用计算可能带来的误差. 很清楚，“slot 分配”算法的时间复杂度为 $O(n)$，n 为单元数目.

6 实验结果

我们在前面提到:我们的算法非常适合于大规模的电路布局.我们可以设想,对于较小的电路,如果矩阵 Q 是稀疏的,直接矩阵求逆法计算速度将快于我们的算法.幸运的是,实际电路中对应的矩阵都是稀疏的,因此直接求逆法能够很好地适用于中小规模的电路设计.对于超大规模的电路,该矩阵仍然是稀疏的;但由于规模过大,直接求逆法不能适用.这一点在我们的实验中得到了证实.

我们共测试了10个实际电路.这些电路的特征数据列在表1中.

表 1 电路特征

电路	引脚数	单元数	线网数	单元行数
C2	373	590	963	9
sioo	62	602	664	12
balu	100	701	801	10
C5	301	1586	1887	16
C7	315	2150	2465	16
s13207	1490	4267	5757	24
s2	1608	9906	11514	28
c213	1489	11030	12519	20
s3	1726	15545	17271	24
avq	64	21854	22183	65

在表2中,我们将我们的布局结果和 Ritual/Tiger 系统获得的布局结果进行了比较.比较基于三种指标:"slot 分配"以后的芯片宽度(由于高度是一样的,因而代表了芯片面积),总线长,布局所用的 CPU 时间(基于 SUN Sparc-20/50工作站,运行 SUNOS4.1.4).

表 2 实验比较结果

电路	Ritual			VEAP		
	总线长	芯片宽度	CPU 时间/s	总线长	芯片宽度	CPU 时间/s
C2	4.15×10^5	2664	47.46	4.09×10^5	2648	57.0
C5	1.09×10^6	3752	194.04	1.03×10^6	3716	170.39
C7	1.81×10^6	4316	199.74	1.64×10^6	4308	220.26
s13207	6.51×10^6	6896	577.21	6.37×10^6	6356	658.05
balu	2.74×10^7	105672	296.89	2.67×10^7	104784	320
s2	1.55×10^7	13876	2617.76	1.46×10^7	12364	2165
sioo	2.90×10^7	125800	262.13	2.76×10^7	124616	105
c213	1.60×10^7	23240	3915.56	1.25×10^7	20904	2863
s3	2.57×10^7	22540	6740.73	2.33×10^7	22336	4716
avq	na	na	na	1.27×10^7	11768	7268.73

在所有的电路中,我们算法得到的芯片宽度比 Ritual/Tiger 得到的要好,总线长结果也比较好(最好的结果高达20%).对于中小规模的电路,VEAP 所需的时间要多于 Ritual/Tiger.但对于非常大规模的电路(如 s2, c213, s3, avq 等),我们的算法速度快于 Ritual/Tiger.对于电路 avq,我们在7268.73秒内得到了较好的结果.而由于资源限制,Ritual/Tiger 不能给出结果.对于这个电路,目前已知的最快算法是[18],在 DEC station 5000/200工作站

上运行13018秒后得到结果.

我们在图3中画出了全局优化算法所用的 CPU 时间随着电路规模增长而增长的函数关系. 从该图中我们可以看出，全局优化算法所用的 CPU 时间随着电路规模的增加近似线性地增长. 从上述实验数据可以看出，我们的算法能够获得相当或更好的布局质量. 虽然在较小规模的电路上，我们的算法不如 Ritual/Tiger 速度快，但在大规模的电路例子中，我们的算法将 Ritual/Tiger 远远抛在身后. 在当前的集成电路工业中，这一点是非常重要的.

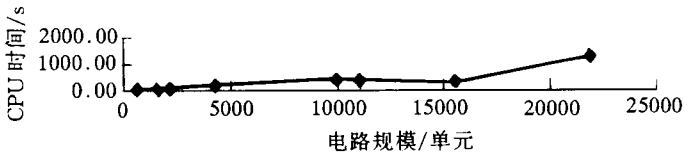


图 3　CPU 时间随电路规模增长的关系

7　结论和进一步的工作

我们的算法非常适合于超大规模的电路布局；实验表明，该算法比现有的所有算法执行效率要高. 这主要是由于该算法的简单迭代方式造成的. 值得注意的是：由于该算法内在的高度并行性，执行效率的进一步提高仍然是可能的. 我们认为，对于极大规模的电路，迭代算法是比较有希望的算法.

我们的算法不仅仅适合于线长优化模式，其基本思想可以稍做修改就可以在总体布局过程中引进时延约束. 我们正在将我们的算法扩展到时延优化模式. 基于集成电路工业发展的现状，仅仅适合于线长优化模式的算法是没有什么吸引力的.

参　考　文　献

[1] U. Lauther, ACM/IEEE Proc. 16th DAC, 1979, 1～10.
[2] A. E. Dunlop and B. W. Kernighan, IEEE Trans. CAD, 1985, **4**(1): 92～98.
[3] D. P. LaPotin and S. W. Director, IEEE Trans. CAD, 1986, **5**(10): 477～489.
[4] P. R. Suaris and G. Kedem, IEEE Trans. Circuits Syst., 1988, **35**(3): 294～303.
[5] B. W. Kernighan and S. Lin, Bell Syst. Tech. J., 1970, 291～307.
[6] C. M. Fiduccia and R. M. Mattheyses, ACM/IEEE Proc. 19th DAC, 1982, 175～181.
[7] T. K. Ng, J. Oldfield and V. Pitchumani, Proc. IEEE Int. Conf. CAD, ICCAD-87, 1987, 470～473.
[8] K. J. Antreich, F. M. Johannes and F. H. Kirsch, IEEE Int. Symp. Circuits and Systems, Proc. ISCAS, 1982, 481～486.
[9] G. J. Wipfler, M. Wiesel and D. A. Mlynski, ACM/IEEE Proc. 19th DAC, 1982, 671～676.
[10] C. K. Cheng and E. S. Kuh, IEEE Trans. CAD, 1984, **3**(7): 218～225.
[11] K. M. Just, J. M. Kleinhans and F. M. Johannes, ACM/IEEE Proc. 23rd DAC, 1986, 308～313.
[12] R. Tsay, E. S. Kuh and C. P. Hsu, ACM/IEEE Proc. 25th DAC, 1988, 318～323.
[13] R. H. J. M. Otten, IEEE Int. Symp. on Circuits and Systems, Proc. ISCAS, 1982, 1017～1020.
[14] J. P. Blanks, ACM/IEEE Proc. 22nd DAC, 1985, 609～615.
[15] J. Frankle and R. M. Karp, IEEE Int. Conf. on CAD, ICCAD-86, 1986, 414～417.
[16] J. M. Kleinhans, G. Sigl, F. M. Johannes *et al.*, IEEE Trans. CAD, 1991, **10**(3): 356～365.
[17] W. Swartz and C. Sechen, Proc. Int. Conf. on CAD, 1990, 336～339.
[18] W. J. Sun and C. Sechen, Proc. Int. Conf. on CAD, 1993, 170～177.

VEAP: Efficient VLSI Placement Algorithm Based on Global Optimization

Kong Tianming, Hong Xianlong and Qiao Changge

(*Dept. of Computer Sci. & Tech., Tsinghua University, Beijing* 100084)

Received 22 July 1996, revised manuscript received 18 November 1996

Abstract For regular ICs, a novel VLSI placement algorithm is presented. The algorithm is based on strict mathematical analysis, can provably find the global optima. And the algorithm's requirements for system resource are rather low. Experimental results are very promising. For a test circuit avq with scale large up to 21000 cells, our algorithm is faster than any existing algorithms. Another point is that our algorithm is suitable for timing driven placement.

EEACC: 2570; **CCACC**: 7410D