维普资讯 http://www.cqvip.com

求解 VLSI 布局问题的启发式算法*)

陈 矛 黄文奇
(华中科技大学计算机学院 武汉 430074)

摘 要 在人们现实布局实践经验的启发下,对 VLSI 布局问题提出了一个启发式算法。该算法由定序规则和定位规则组成,定序规则用来确定布局物体放入布局空间的先后顺序,定位规则规定每一布局物体都被当前最优的占角动作放入布局空间。对 5 个 MCNC 算例的测试结果表明,本文算法与基于 O-tree 表示的算法相比,速度提高 15～56 倍;对于其中 4 个算例,面积利用率提高 0.95%～5.31%。

关键词 启发式算法,布局,定序规则,定位规则

A Heuristic Algorithm for Solving VLSI Block Placement Problem

CHEN Mao HUANG Wen-Qi
(College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract Inspired by human's accumulated experience on solving similar problems in everyday life, a heuristic algorithm is proposed for solving the VLSI block placement problem. The algorithm consists of the ordering principle and the locating principle. The blocks are packed into the packing space one by one according to the ordering principle. According to the locating principle, each block is packed into the rectangular frame by the corner-occupying action with the highest benefit. The proposed algorithm is 15～56 times faster experimentally than the O-tree-based algorithm as well as achieving higher area usage with about 0.95%～5.31% improvements for the five MCNC benchmarks except xerox.

Keywords Heuristic algorithm, Placement, Ordering principle, Locating principle

1 引言

布局问题是指给定一个布局空间和若干待布物体,将待布物体按一定的约束条件合理地摆放在布局空间中,并要求达到某种最优指标。布局问题是 VLSI 设计流程中的重要问题,其完成的质量对布线工作的顺利完成乃至最终芯片的性能都有很大影响[1]。

VLSI 布局可以分为二划分(slicing)结构和不可二划分(non-slicing)结构两类[2]。二划分的布局结构简单,可以表示为一棵二叉树,但它只涵盖了部分布局结果空间。相对于二划分结构,不可二划分结构的布局更具有一般性,也更加灵活,因而具有更高的面积利用率。但不可二划分结构的布局更加复杂,实现也更困难。

不可二划分结构的布局问题是 NP 难度问题[3]。对于 NP 难度问题,在可行的时间界限内不可能找到问题的精确解[4～6],目前大多采用随机优化算法。使用随机优化算法求解 VLSI 布局问题需要建立在布局构形的有效表示形式的基础上。1999 年,Guo[7] 等人提出了一种称为有序树(O-tree)的编码表示来描述布局构形。对于模块数为 n 的布局问题,该表示形式只需要 $n(2+\lceil \lg n \rceil)$ 位编码来描述,冗余度较小。O-tree 表示的解空间为 $O(n!\ 2^{2n-2}/n^{1.5})$,要小于其它布局表示形式[8,9]。基于 O-tree 表示的算法的时间复杂度为 $O(n^3)$,但由于这一表示与模块尺寸有关,其实际计算复杂度比宣称的要大。

长期以来,人们在铺地、砌墙等现实的布局实践中,积累了丰富的经验。在人们实践经验的基础上,本文提出一个启发式算法来求解 VLSI 布局问题。通过对 5 个 MCNC 标准算例[5]进行测试,并与基于 O-tree 表示的算法进行比较,结果表明本文算法十分有效。

2 问题描述

把 VLSI 布局问题加以抽象化、形式化,可描述为:给定 N 个矩形块 $R_1, R_2, \cdots, R_N$(R_n 的长为 l_n,宽为 w_n,$n=1,2,\cdots,N$)和一个足够大的矩形框,把矩形块摆放到矩形框内,并要求满足合法条件:1)任一矩形块的两边分别与矩形框两边平行;2)任意两个矩形块不能重叠;3)任一矩形块不得超出矩形框边界。

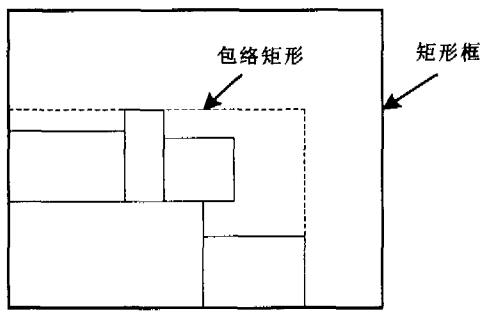

图 1 包络矩形示意图

*)国家自然科学基金资助项目(10471051)和国家 973 计划资助项目(2004CB318000)。**陈 矛** 博士生,主要研究方向为算法设计与分析;**黄文奇** 教授、博士生导师,主要研究领域为 NP 难问题现实求解和算法优化。

维普资讯 http://www.cqvip.com

根据上述3个合法条件将所有矩形块摆放到矩形框后就得到一种布局，称为合法布局。这样，在矩形框内就形成了一个把所有矩形块恰好容纳进去的包络矩形，如图1所示。VLSI布局问题就是要寻找一种合法的布局，并使所得包络矩形面积最小。

3 启发式算法

3.1 算法思想

人们在劳动过程中，经常会碰到现实的布局问题，如用石块、砖头等多边形物体铺地、砌墙等。在实际布局中，人们都是逐块将物体放入到布局空间。为了使块与块之间紧凑致密，人们在实践过程中形成了一些布局经验，概括起来可以用定序和定位两方面来描述。

定序是用来确定布局物体放入布局空间的先后顺序，它对布局结果有很大影响。不同的块，被重视程度是不同的，在实际布局中，通常那些体积大或长度长的块被优先考虑。

定位是用来确定当前布局块在布局空间中的位置。在人们的布局实践中，布局物体通常被摆放到布局空间的某一个角上，以尽量靠近已摆放的块。这样的布局过程就是一个占角摆放的过程。直观上看，这样的布局结果很紧凑，布局空间的利用率较高。

本文算法是在对这些实践经验做进一步完整化和精确化处理的基础上得到的。

3.2 基本定义

定义1 格局。假设矩形框里已经放入 $i(i=0,1,2,\cdots,N)$ 个矩形块在各自确定的位置上，这些已放入的矩形块的位置的集合称为一个格局。

定义2 实角和半虚角。如果组成角的两边都是实边（构成实边的可以是矩形块的边或矩形框的边），就称之为实角，如图2中 $\angle AEI$ 、$\angle ADC$ 。反之，称之为虚角，如图2中 $\angle HIG$ 、$\angle HIE$ 。对于虚角，如果组成角的两边中有一条边是实边，称之为半虚角，如图2中 $\angle HIE$、$\angle GIF$ 。

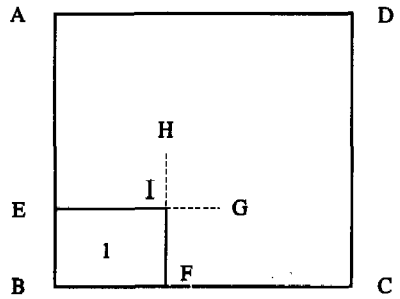

图2 角示意图

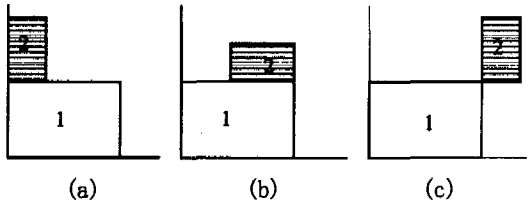

图3 占角动作示意图

定义3 占角动作 α。在任一格局下，如果某个动作把当前待布块摆放到矩形框中的某个角上，并且满足摆放过程的3个合法条件，我们称该动作为占角动作。若占角动作占据的是个实角，就称为占实角动作，如图3(a)。若占角动作占据的是个虚角，就称为占虚角动作，如图3中(b)和(c)。在占虚角动作中，如果矩形块占据的是一个半虚角，则称该占角动作为占半虚角动作，如图3(b)所示。本文算法只考虑占实角动作和占半虚角动作。

定义4 占角动作的接触度 P。占角动作把当前待布块 R_n 摆放到矩形框内某一角时，在已放入的 $n-1$ 个矩形块和矩形框的四个边框中，与 R_n 相接触的个数就称为该占角动作的接触度。图4中，R_4 与 R_2、R_3 以及矩形框的右边框接触，所以摆放 R_4 的占角动作的接触度 P 等于3。若占角动作具有较高的接触度，就能把矩形块摆放得很紧凑，使得布局空间的利用率较高。

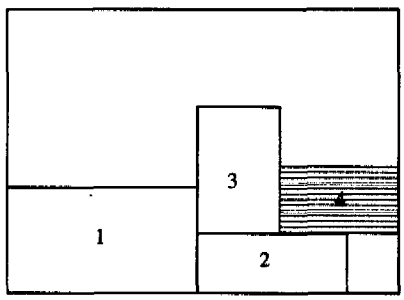

图4 接触度示意图

3.3 定序规则和定位规则

在实际布局中，那些体积大或长度长的布局物体被优先放入布局空间中。为了综合考虑面积和长边这两个因素，本文给出价值度 V_n 作为矩形块 R_n 的评价函数：

$$V_n=\lambda_1 * l_n * w_n+\lambda_2 * \max(l_n,w_n)$$

其中 λ_1、λ_2 是权重因子，$\lambda_1+\lambda_2=1$，l_n 和 w_n 分别是 R_n 的长和宽。对矩形块按照价值度大小降序排列，就是矩形块被放入矩形框的顺序。

在当前格局下，矩形框内可能有很多角。若当前待布块被放到某个角后能满足布局过程的合法条件，该角就称为候选角。一个候选角对应一个候选占角动作。定位规则就是确定如何从众多候选占角动作中挑选一个最优的来摆放当前矩形块。在人们直观经验的基础上，我们给出如下定位规则：

在有 $n-1$ 个矩形块已放入的格局下，假设当前待布块 R_n 有 m 个候选占角动作。对候选占角动作 α_j $(j=1,2,\cdots,m)$，试探性地做 α_j 后将得到一种新的格局，计算新格局所对应包络矩形的面积 S_j 和 α_j 的接触度 P_j，以及 R_n 的左下角坐标 (x_j,y_j)，这样就得到一个有序四元组 $\langle S_j,P_j,x_j,y_j\rangle$。这 m 个候选的占角动作就对应 m 个有序四元组。

对有序实数四元组 $\langle S_u,P_u,x_u,y_u\rangle$ 和 $\langle S_v,P_v,x_v,y_v\rangle$ $(1\leqslant u,v\leqslant m)$，若满足下列条件之一：(1) $S_u<S_v$；(2) $S_u=S_v$ 且 $P_u>P_v$；(3) $S_u=S_v$ 且 $P_u=P_v$ 且 $x_u<x_v$；(4) $S_u=S_v$ 且 $P_u=P_v$ 且 $x_u=x_v$ 且 $y_u<y_v$，则规定四元组 $\langle S_u,P_u,x_u,y_u\rangle$ 比四元组 $\langle S_v,P_v,x_v,y_v\rangle$ 具有高的优先级，即占角动作 α_u 的优先级比占角动作 α_v 的优先级高。

利用该优先准则，从这 m 个有序四元组中选取优先级最高的一个，以该有序四元组所对应的候选占角动作摆放当前待布块 R_n。

3.4 算法实现

按照定位规则，在试探性地摆放一矩形块后，从试探性布局得到的结果中选择最优的一个，以该最优结果对应的占角动作来放置当前块。

为了提高优度，本文算法“把眼光看得更长远一些”。在试探性地放入当前块后，分别从新的格局出发，按照定位原则

维普资讯 http://www.cqvip.com

把所有剩下待布块都试探性地放入到矩形框后，才从试探性布局得到的结果中选择最优的一个，以该最优布局结果对应的候选占角动作来确定性地摆放当前块。算法描述如下：

Step1　输入矩形框和 N 个矩形块的长和宽，并计算各个矩形块的价值度。$n \leftarrow 1$。

Step2　从待布矩形块中选取价值度最大的一块作为当前待布块 R_n，若 $n=N$，转 Step4。

Step3　在当前已有 $n-1$ 块放入的格局中，假设当前待布块 R_n 有 m 个候选占角动作，对于每个候选占角动作，试探性地做该占角动作后将得到一种新的格局，那么 m 个候选占角动作就对应 m 个新的格局。

分别从这 m 个新格局出发，按照定位规则将所有剩下待布块试探性地布局到矩形框，就得到 m 个合法布局及其相应包络矩形面积 $S_1, S_2, \cdots, S_m$。这样，初始的 m 个候选占角动作就对应 m 个合法布局，从这 m 个合法布局中选取面积最小的合法布局 $S_k^* = \min(S_k)$，其中 $k=1,2,\cdots,m$。以该合法布局对应的候选占角动作确定性地摆放 R_n。$n \leftarrow n+1$，转 Step2。

Step4　按照定位原则摆放 R_N。

Step5　输出布局面积和各矩形块位置，算法结束。

3.5 算法分析与比较

对一个有 n 个模块的布局问题，由于矩形框内角的数量与已布局块数存在线性关系，因此定位原则的时间复杂度为 $O(n)$。从一个候选占角动作出发，试探性地把剩下待布块都放入矩形框的时间复杂度为 $O(n^2)$。把当前候选占角动作都试探完后，确定性地摆放一块的时间复杂度是 $O(n^3)$。这样，本文算法的时间复杂度就是 $O(n^4)$。

基于 O-tree 表示的算法的时间复杂度是 $O(n^3)$，要低于本文算法的时间复杂度。但是在实际布局中，表示形式与布局之间需要转换。在每次转换时，基于 O-tree 表示的算法都要对模块进行向左和向下的压缩操作，这样它的实际运算时间将会延长。

与随机优化算法不同的是，本文算法是一个确定性算法。随机优化算法需要建立在布局表示形式的基础上。为了保证解空间包含最优解，每一个布局都要有一个表示与之相对应，这样会使得解空间很大，搜索时间长。并且表示形式都具有一定的冗余性，即多个表示对应同一个布局，从而影响了算法的效率。作为确定性算法，本文算法不会出现上述缺点。

4 实验结果

在主频为 2.4GHz 的 P4 微机(512M 内存)上对 MCNC5 个标准算例进行了测试，并与基于 O-tree 表示的算法[10]进行了比较(表 1)。基于 O-tree 表示的算法是在 SUN Spark Ultra-I 工作站上进行测试的。这两种计算机的运行速度大致相当。

表 1　本文算法与基于 O-tree 表示的算法的比较

算例	O-tree			本文算法		
	面积 (mm^2)	面积利用率(%)	时间 (s)	面积 (mm^2)	面积利用率(%)	时间 (s)
apte	47.76	97.47	0.63	47.31	98.42	0.02
xerox	20.18	95.89	1.68	20.25	95.56	0.03
hp	9.49	93.05	1.09	9.17	96.07	0.03
ami33	1.25	92.52	25.40	1.20	96.37	2.23
ami49	38.60	91.83	154.7	36.49	97.14	9.89

从表 1 可以看出，在计算时间上，对不同算例，本文算法比基于 O-tree 表示的算法要快 15～56 倍。

在面积利用率上，除算例 xerox 外，本文算法得到的结果比基于 O-tree 表示的算法得到的结果分别提高 0.95%，3.02%，3.85%和 5.31%，说明本文算法具有较高的效率。特别是对 ami33 和 ami49 这两个模块数多的算例(模块数分别为 33 和 49)，在面积利用率上的提高更有现实意义。

对算例 xerox，本文算法得到的面积利用率比基于 O-tree 表示的算法得到的结果小 0.3%。算例 xerox 有 10 个模块，其中 9 个模块的大小和形状比较接近。在基于 O-tree 表示的算法中，压缩操作与模块尺寸有关，对于大小形状差异较小的模块，压缩操作的效果更好。所以对算例 xerox，基于 O-tree 的算法得到的面积较优一些。

图 5 和图 6 分别给出 ami33 和 ami49 的布局图。图中黑色部分是没有利用的区域。

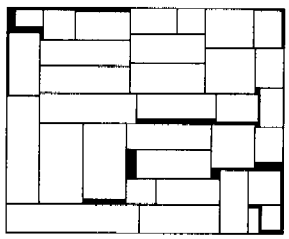

图 5　ami33 的布局结果

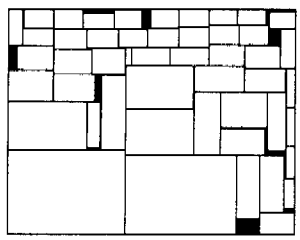

图 6　ami49 的布局结果

结束语　通过向人类生活和劳动中的经验和智慧学习，可以启发我们设计出各种策略以制定出高效实用的近似算法，来解决 VLSI 布局问题和其它一些 NP 难度问题。沿着启发式的思路，下一步我们将求解带有预置模块的布局问题和带有软模块的布局问题，并把模块之间的互连线总长考虑进来，使之与布局面积共同作为算法的优化目标。

参 考 文 献

1　王金敏，简其和. 矩形布局问题中的群组策略. 计算机辅助设计与图形学学报，2004,16(4):572～575

2　董社勤，洪先龙. 基于矩形宏模块的片上系统布图规划算法. 清华大学学报(自然科学版)，2003,43(4):484～486

3　Murata H, Fujiyoshi K, Nakatake S, et al. Rectangular packing-based module placement. In: Proc. IEEE Int Conf on Computer-Aided Design, Los Alamitos, 1995. 472～479

4　黄文奇，朱虹，许向阳，等. 求解方格 packing 问题的启发式算法. 计算机学报，1993,16(11):829～836

5　Huang Wenqi, Xu Ruchu. Two personification strategies for solving circles packing problems. Science in China(Series E), 1999, 42(6):595～602

6　陈传波，何大华，黄文奇. 求解单位等边三角形 Packing 问题的近似算法. 计算机学报，2003,26(2):212～220

7　Guo Peining, Cheng Chungkuan, Yoshimura T. An O-Tree representation of non-Slicing floorplan and its applications. In: Proc. 36th ACM/IEEE Design Automation Conf, New Orleans, 1999. 268～273

8　Nakatake S, Fujiyoshi K, Murata H, et al. Module placement on BSG structure and IC layout applications. In: Proc. ICCAD, Los Alamitos, 1996. 484～491

9　Tang Xiaoping, Wong D F. FAST-SP: a fast algorithm for block placement based on sequence pair. In: Proceedings of the ASP-DAC 2001, 2001. 521～526

10　Chang Yunchih, Chang Yaowen, et al. B*-Trees: A New Representation for Non-Slicing Floorplans. In: 37th ACM/IEEE Design Automation Conference(DAC), Los Angeles, 2000. 458～463