标准单元模式下的一种快速增量式布局算法

姚 波[1],洪先龙[1],于 泓[1],蔡懿慈[1],顾 钧[2]
(1. 清华大学计算机科学与技术系,北京 100084;2. 香港科技大学计算机科学系,香港)

摘 要: 增量式布局是适应高性能设计要求的一种新的布局模式.它针对电路更改,局部地调整单元位置,重新获得合理的布局.本文提出了一种标准单元模式下的快速增量布局算法.算法采用单元行划分的方法处理布局约束,然后将布局调整归结为单元依次插入单元行的问题,并构造了一个数学规划求解最佳的插入方案.同时提出了复杂度为 $O(n)$ 的双对角线搜索法求解这个特殊的数学规划.实际电路测试表明算法高效而稳定,比简单的启发式算法快十倍,并使布局修改减少 20 %以上.

关键词: 增量式布局; 标准单元; 双对角线搜索
中图分类号: TP3 **文献标识码**: A **文章编号**: 0372-2112 (2001) 02-0211-04

A Fast Incremental Placement Algorithm for Standard-Cells

YAO Bo[1], HONG Xian-long[1], YU Hong[1], CAI Yi-ci[1], GU Jun[2]
(1. *Dept. of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*;
2. *Dept. of Computer Science, The Hong Kong University of Science and Technology, Hong Kong*)

Abstract: Incremental placement or ECO placement is a new field in VLSI layout to meet the demand of high performance design. It detects illegal cell positions and moves cells locally to find a feasible placement solution. This paper proposes an incremental placement algorithm for standard-cell design mode. It adopts row partitioning to cope with various placement constraints, and then inserts trouble cells one by one. An integer programming (IP) problem is constructed and solved to get the best solution of cell insertion. A dual diagonal searching (DDS) method is proposed to solve this special IP problem. We prove that DDS can always be the best solution and its time complexity is $O(n)$. Experiments on a group of industrial test cases show that our algorithm is efficient and robust. It runs 10 times faster than a simple heuristic method and reduces more than 20 percent placement modification in average.

Key words: incremental placement; standard-cell; dual diagonal searching

1 引言

现行的超大规模集成电路(VLSI)物理设计中,常常需要在完成布局后对电路进行局部修改,以改善电路的性能,满足时延、功耗和布线成功率方面的要求.比如,为改善时延性能而插入一些缓冲(buffer)单元,为保证布线成功而插入走线道(feed-through)单元,或为完成电源线/地线网络拓扑方案而移动一些单元的位置等.这些修改都需要知道确切的单元位置,必须在布局完成后进行.由此产生一个布局调整的问题,因为插入或替换单元都有可能与原有单元重叠,所以要调整单元的位置,重新得到合理的布局结果.作者把这种布局称为增量式布局或 ECO(Engineer Change Order)布局.

传统的布局方法[1~3]不能用于增量式布局,因为它们总是将所有单元作为布局对象,使设计周期大大延长.另一种办法是用采用交互式的方法手工修改版图.但超大规模集成电路设计中往往包含几万甚至几十万单元,手工修改是不合适的.因此,要寻找一种增量式布局的算法,能针对电路的修改,自动地以最小代价局部地调整布局,找到新的可行布局方案.增量式布局是超大规模集成电路布局研究中的一个新兴领域,应用范围很广.但由于增量布局问题本身的复杂性,很难找到快速高效的解决方法,故这方面发表的文献在国内外都不多见.文献[4]中提出了一种针对门阵列模式的增量式布局算法,这种方法先用模板枚举或值传播找到重叠单元到邻近空位的一条最佳路径,然后沿路径做一系列单元局部推移以完成增量布局的工作.但在标准单元设计模式下,单元宽度是不同的,而门阵列中每个单元大小都相同,这使文[4]中的方法很难应用于标准单元设计.文献[5]采用划分单元行的方法,解决了标准单元布局中的多种约束,使该模式下增量式布局得以实现,是标准单元增量式布局的较早尝试.其中采用了一种简单的启发式方法寻找单元插入一行的单元调整方法,使布局质量受到影响.在本文中,采用了文[5]中划分单元行的方法,将增量布局转化为一系列单元插入单元行的操作.然后将寻找最佳插入方案的问题转化为整数规划问题,并提出

收稿日期:2000-05-23;修回日期:2000-06-20
基金项目:国家自然科学基金(No. 69776027);973 国家重点科研项目(No. G1998030403)

了一种时间复杂度为 $O(n)$ 的快速求解方法——双对角线搜索法. 我们还给出了双对角线搜索法的分析,包括正确性证明和复杂度分析.

本文的组织如下:第二节描述了增量式布局及其实现方法. 第三节介绍了寻找一个单元插入一行的最佳方案的方法,并证明了这种方法一定能找到最佳解. 第四节中给出了实验结果,并与已有的一个算法进行了比较. 最后给出结论.

2 增量式布局及其实现

2.1 问题的描述

标准单元模式下,增量式布局的输入为一个初始布局的结果,包括一些电路单元,它们已被安置在一定的位置. 还有单元放置的位置约束,包括布局区域和单元行的定义等. 输入的单元中含有部分位置不合法的单元,称之为非法单元. 算法要找出所有非法单元,并将它们安置在合理的位置,为此要进行必要的布局调整. 增量布局的输出为调整后的单元位置.

在调整布局时,希望对原有布局方案的改动尽可能小,以尽量保持原布局方案的优点,同时也减少布局调整带来的后续工作,如重新布线等. 用如下的指标来衡量一个布局调整方案的代价:首先是调整涉及的单元移动数目,它越少越好. 然后是单元在 X 和 Y 两个方向移动的最大距离和总距离,这些移动距离也应尽可能小. 这些指标反映了布局改动的大小,计算简单,现有的增量布局算法多采用这样的目标.

2.2 增量式布局的实现过程

增量布局分为以下几个步骤进行,流程如图 1 所示. 输入初始布局后,首先用单元行划分处理各种约束,然后寻找所有的非法单元,接着依次将非法单元插入合适位置,同时进行必要的布局调整,使非法单元插入后布局仍然合理. 安置完所有非法单元后,输出最终的布局结果.

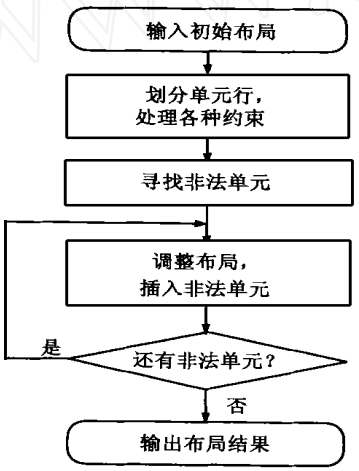

图 1 增量布局的总体流程

标准单元模式下的布局问题存在许多实际约束,比如固定单元约束,即某些单元只能放置在固定位置;禁入区域约束,即某些区域内不能安置单元;单元行重叠的约束,即布局平面中存在不同高度的单元行,它们之间有可能重叠. 我们采用单元行切割的方法来处理这些实际的布局约束,具体的约束和处理方法可参见文献[5]. 处理完约束后,重新得到一系列单元行,每行的所有位置都可以用来放置单元.

然后寻找所有的非法单元,作为增量式布局必须处理的对象. 非法单元可能有以下两种情况:一种是没有完全落在某个单元行内的单元,比如图 2(a) 中的单元 A. 它们每个都有一部分落在可布局区域之外. 另一种是虽然完全落在单元行内,但与其他单元重叠的单元,例如图 2(a) 中的单元 B. 找到所有非法单元后,我们先将它们从原来的布局中摘除. 这样,剩下的单元构成一个合法的布局.

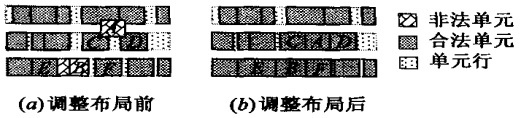

图 2 增量式布局示意图

接着再将所有的非法单元依次插入现有的布局中,并使每次插入后布局仍然合法. 插入单元时需要移动已布局的单元,以找到合适的插入位置. 例如图 2(a) 中,为了在第二行插入单元 A,将单元 C 和 D 分别向左右两边推,在中间让出的空隙中插入单元 A,布局调整后的结果如图 2(b) 所示. 我们用如下的方法寻找最佳的布局调整方案和单元插入位置. 对于一个非法单元,先找到离它现有位置最近的若干单元行作为插入的候选行,然后寻找将非法单元插入每个候选行的最佳方案. 寻找最佳插入方案的方法将在下节中介绍. 再计算插入这行的代价,它包括移动的单元数目和总移动距离. 最后取代价最小的那个候选行插入非法单元. 需要指出的是,虽然非法单元最终被插入一行,但最佳插入行的选择是在局部的二维区域内进行的,所以增量布局优化仍然是二维优化.

3 寻找单元插入一行的最佳方案

增量式布局的最基本操作是将一个单元插入某单元行,并调整行内原有的单元位置,使该行内所有单元相互不重叠. 希望找到一个最优的插入位置和单元移动方案,使调整带来的单元位置改变最少.

3.1 双对角线搜索算法

采用如下方法寻找移动单元数目最少的调整方案. 设将单元 C_i 插入行 R_k 中,C_i 的宽度为 W_i. 设 R_k 中原有单元从左往右依次为 $C_{k1}, C_{k2}, \cdots, C_{kn}$. 首先根据 C_i 的 x 坐标就近寻找它的插入位置,设它将被插入到单元 $C_{k(j-1)}$ 和 C_{kj} 之间,这时 $C_{k(j-1)}$ 和 C_{kj} 之间的空隙宽度为 G. 若 $G \geq W_i$,则直接将 C_i 插入该空隙;若 $G < W_i$,则将空隙扩大 $ME = W_i - G$ 的宽度后再插入 C_i. 空隙的扩大由单元推移实现,过程如下.

这时在插入单元的空隙左边有 $L = j$ 个单元,从右往左依次为 $C_{l1}, C_{l2}, \cdots, C_{lL}$,它们的 x 坐标和宽度分别为 $x_{l1}, x_{l2}, \cdots, x_{lL}$ 和 $w_{l1}, w_{l2}, \cdots, w_{lL}$. 试图把它们往左移. 显然 C_{lp} 就是前面所指的 $C_{k(j-p)}$. 空隙右边有 $R = n - j + 1$ 个单元,从左往右依次为 $C_{r1}, C_{r2}, \cdots, C_{rR}$,它们的 x 坐标和宽度分别为 $x_{r1}, x_{r2}, \cdots, x_{rR}$ 和 $w_{r1}, w_{r2}, \cdots, w_{rR}$. 它们将向右移. 移动时不改变单元的左右顺序,只是将遇到的每个空隙压紧,从而在插入单元的空隙处留出尽可能多的空间. 建立了一些辅助结构来寻找最佳推移方案. 设 G_{lp} 为单元 $C_{l(p-1)}$ 和 C_{lp} 之间的空隙宽度,而 G_{rp} 为单元 C_{rp} 和 $C_{r(p+1)}$ 之间的空隙宽度,即

$$G_{lp} = \begin{cases} 0, & p = 0; \\ x_{lp} - x_{l(p-1)} - w_{l(p-1)}, & l \leq p < L; \\ x_{lp} - LB, & p = L; \end{cases}$$

$$G_{rp}=\begin{cases}0, & p=0;\\ x_{l(p+1)}-x_{lp}-w_{lp}, & l\le p<R;\\ RB-x_{lp}-w_{lp}, & p=R;\end{cases}$$

其中 LB 为行 R_k 的左边界,RB 为行 R_k 的右边界.再设 $ML_p=\sum_{i=0}^{p}G_{li}$ 和 $MR_p=\sum_{i=0}^{p}G_{ri}$ 分别为推移到左边第 p 个单元和推移到右边第 p 个单元时可在插入空隙处增加的宽度.这样,移动单元数最少的移动方案可由如下的整数规划问题求得:

$$\begin{aligned}&\text{Min} && l+r\\ &\text{Subject to} && 0\le l\le L,\ l\text{为整数};\\ &&& 0\le r\le R,\ r\text{为整数};\\ &&& ML_l+MR_r\ge ME\end{aligned}\qquad(1)$$

对这个特殊的整数规划问题,采用下面的方法求解.这种方法总是试图沿两个对角线方向寻找最优解,称之为双对角线搜索法(Dual Diagonal Searching,DDS).

最优解的搜索在一个 $(L+1)\times(R+1)$ 的矩阵 M 中进行.矩阵元素 $m_{lr}=ML_l+MR_r$.首先沿主对角线寻找第一个可行解.对于 $L>R$ 的情况,搜索的序列为 $m_{00},m_{11},\cdots,m_{RR},m_{(R+1)R},\cdots,m_{LR}$.类似地可以得到 $L\le R$ 时的搜索序列.序列中第一个大于或等于 ME 的值 m_{pq} 对应第一个可行解 (p,q).若在序列中不存在大于等于 ME 的值,则规划问题无解,即不能将单元插入该行.

找到第一个可行解 (p,q) 后,我们得到了将非法单元插入这行所需移动单元数目的一个上界 $N_u=p+q$,然后分两支寻找最优解.左支:设当前找到的解为 (l,r),则搜索的下一个解 (l',r') 由如下方法确定,先定 l',再定 r':

$$l'=\begin{cases}l, & m_{lr}\ge ME\\ l+1, & m_{lr}<ME\end{cases},$$

$$r'=\begin{cases}r-1, & m_{lr}\ge ME\text{ 或 }m_{lr}<ME\text{ 且 }l'+r>N_u\\ r, & m_{lr}<ME\text{ 且 }l'+r\le N_u\end{cases}$$

直到 $r<0$ 或 $l>R$ 为止.当找到一个可行解后,实时更新 N_u,使它始终为以后要寻找的可行解中单元移动数目的上界.

同样对于右支,设当前找到的解为 (l,r),则搜索的下一个解 (l',r') 由如下方法确定,先定 r',再定 l':

$$r'=\begin{cases}r, & m_{lr}\ge ME\\ r+1, & m_{lr}<ME\end{cases}$$

$$l'=\begin{cases}l-1, & m_{lr}\ge ME\text{ 或 }m_{lr}<ME\text{ 且 }r'+1>N_u\\ l, & m_{lr}<ME\text{ 且 }r'+1\le N_u\end{cases}$$

直到 $r>R$ 或 $l<0$ 为止.左右两支都从解 (p,q) 开始,搜索时找到的 $l+r$ 最小的可行解即为最优解.

找到最优解 (P,Q) 后,将单元 $C_{l1},C_{l2},\cdots,C_{lP}$ 向左推移,并使它们之间没有空隙,同样将单元 $C_{r1},C_{r2},\cdots,C_{rQ}$ 向右推移,使它们之间也没有空隙.最后将单元 C_i 插入单元 $C_{k(j-1)}$ 和 C_{kj} 之间.

3.2　双对角线搜索算法的分析

首先证明上述双对角线搜索法一定能求得规划(1)的最优解(如果最优解存在).为此,给出如下引理、定理及证明:

引理　若 (M,N) 不是规划(1)的可行解,则对于一切 $0\le l\le M$ 和 $0\le r\le N$,(l,r) 也不是可行解.若 (M,N) 是规划(1)的可行解,则对于一切 $M\le l\le L$ 和 $N\le r\le R$,(l,r) 也是可行解.

证明　当行内原有单元之间没有重叠时,G_{lp} 和 G_{rp} 非负,所以 ML_p 和 MR_p 随 p 的增加而单调增加.这时:若 $m_{MN}<ME$,则对于所有 $0\le l\le M,0\le r\le N$,有 $m_{lr}\le m_{MN}<ME$.这表示若 (M,N) 不是可行解,则对于所有 $0\le l\le i,0\le r\le j$,(l,r) 也不是可行解.而当 $m_{MN}\ge ME$,则对于所有 $M\le l\le L,N\le r\le R$,有 $m_{lr}\ge m_{MN}\ge ME$.这表示若 (M,N) 是可行解,则对于所有 $M\le l\le L,N\le r\le R$,(l,r) 也是可行解.引理得证.由于是依次插入非法单元,可以保证插入之前行内单元没有重叠.

定理　如果规划(1)存在可行解,双对角线搜索法一定能求得最优解.

证明　如果规划(1)存在可行解,根据引理,(L,R) 必为可行解,因而寻找第一个可行解的序列中必存在可行解,设 (L_1,R_1) 为序列中的第一个可行解,这时 (L_1-1,R_1-1) 不是可行解.根据引理,最优解只可能存在于两个集合 S_{L0} 和 S_{R0} 中,$S_{L0}=\{(l,r)\mid L_1\le l\le L,r\ge 0,l+r\le L_1+R_1\}$,对应于左支查找;$S_{R0}=\{(l,r)\mid R_1\le r\le R,l\ge 0,l+r\le L_1+R_1\}$,对应于右支查找.下面以左支搜索为例说明不论 S_{L0} 中哪一个解是最优解,左支查找都能找到.设这个最优解为 $(L_L,R_R)\in S_{L0}$,则根据引理,集合 $S_{LL}=\{(l,r)\mid L_1\le l\le L_L,r=R_R\}$ 中每个解均不是可行解,而集合 $S_{LU}=\{(l,r)\mid l=L_L,R_R<r\le R_1\}$ 中每个解都是可行解.这样若搜索到 S_{LL} 中的任何一个解 (l,R_R) 后,根据左支搜索的方法,都会沿 $(l,R_R),(l+1,R_R),\cdots,(L_L,R_R)$ 的顺序搜索到最优解.同样,若搜索到 S_{LU} 中的任何一个解 (L_L,r) 后,也会沿 $(L_L,r),(L_L,r+1),\cdots,(L_L,R_R)$ 的顺序搜索到最优解.而在左支搜索中,每查找一个新解,或者是 l 加 1,或是 r 减 1.这样从 (L_1,R_1) 开始若干步后,必定搜索到 S_{LL} 或 S_{LU} 中的某一解而最终搜索到 (L_L,R_R).于是左支搜索能找到落在 S_{L0} 中的任何一个最优解.同理,若最优解落在 S_{R0} 中,右支查找也一定能找到.综上所述,只要规划(1)存在可行解,双对角线搜索法就能求得最优解,定理得证.

然后分析双对角线搜索算法的时间复杂度.设单元插入的候选行含 n 个单元,在规划(1)无解时,查找步数最多为 n.而当规划(1)有解时,最多的查找步数为 $(n+1/4n)$,对应于最优解在矩阵 M 的左上角或右下角得到的情况.所以算法的时间复杂度为 $o(n)$.

4　实验结果

在 Sun 工作站上用 C 语言实现了本文提出的算法,称之为 F-ECOP.实验测试了一组来自工业界的算例.这些算例的特征数据列于表 1 中,其中包含的单元数目从几百至几万不等,位置非法的单元数目也从几十至几千.将 F-ECOP 的运行结果列于表 2 中,并与另一个增量布局算法 ECOP[5] 的结果进行了

表 1　测试电路特征数据

测试用例	单元数	非法单元数
cnt100	656	32
cnt1000	7146	357
gfsm300	3536	176
gfsm50	616	30
m16_6	2984	149
m32_my	7018	350
obs1_d	197	20
po300	2005	100
son_1	24584	1229
tsb_2	16056	1062

比较. ECOP 采用了简单的启发策略将单元放入行中:若插入位置的空隙宽度不足,则首先将左边的单元向左推移,若还得不到足够宽度,再考虑将右边的单元向右推移. 表 2 中的距离单位为某个约定值.

从测试结果可以看出,F-ECOP 在运行速度上较 ECOP 明显加快,在测试的十个算例上平均快了 10 倍. 这主要是因为我们限制了单元插入时候选行的范围,而不是尝试布局区域中的每一行. 另外,双对角线法同时加快了有解和无解两种情况下的搜索. 在布局效果上,F-ECOP 同样优于 ECOP. 在考察的四个指标上,F-ECOP 都比 ECOP 有较大幅度的减少. 其中移动单元数目平均减少了 19.4 %,总移动距离平均减少了 38.7 %. 这是由于双对角线法能找到最优解,而 ECOP 中的简单启发式方法只是找到一个可行解.

表 2 测试结果

	F-ECOP					ECOP				
	移动单元数	X 方向最大移动距离	Y 方向最大移动距离	总移动距离	运行时间(s)	移动单元数	X 方向最大移动距离	Y 方向最大移动距离	总移动距离	运行时间(s)
cnt100	33	3944	3938	219447	0.02	51	4500	22138	316487	0.06
cnt1000	383	27000	10355	3996306	0.38	439	16163	171645	6530346	9.00
gfsm300	176	8452	5453	1398948	0.15	253	11048	60053	2297246	1.75
gfsm50	30	2032	2021	92078	0.02	46	3000	20221	161338	0.05
m16_6	152	8813	4699	984558	0.10	179	10313	49901	1482532	1.21
m16_6_block	150	6308	5687	1185227	0.14	194	6000	30713	1525427	1.30
m32_my	378	9743	24953	3384512	0.34	435	11243	47847	4728072	8.30
obsl_d	21	7500	36400	360424	0.02	21	4500	54600	478348	0.02
po300	100	5197	3698	559590	0.08	141	8197	87302	931022	0.54
son_1	1864	40500	55673	17435698	9.69	1846	38347	180927	32709454	105.05
tsb_2	1560	16800	18964	7250648	7.16	1591	15436	80564	12125504	54.20

5 结论

本文研究了标准单元模式下的增量式布局问题,我们将布局调整问题归结为单元插入单元行的问题,然后构造一个整数规划求调整的最佳方案. 对于这个特殊的整数规划问题,我们提出了双对角线搜索方法求解,其时间复杂度为 $O(n)$. 实例测试证明,我们的算法可以快速高质量地解决标准单元模式下的增量布局问题.

参考文献:

[1] R. Tsay, E. S. Kuh, and C. P. Hsu. PROUD: A sea-of-gates placement algorithm [J]. IEEE Design Test Comput., Dec. 1988, 5: 44 - 56.

[2] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antereich. GORDIAN: VLSI placement by quadratic programming and slicing optimization [J]. IEEE Trans. Computer-Aided Design, Mar. 1991, 10: 356 - 365.

[3] A. Srinivasan, K. Chaudhart and E. S. Kuh. RITUAL: A performance driven placement algorithm [J]. IEEE Trans. Circuits Syst. II, Nov. 1992, 39: 825 - 840.

[4] C. S. Choy, T. S. Cheung, and K. K. Wong. Incremental layout placement modification algorithms [J]. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Apr. 1996, 15: 437 - 445.

[5] 于泓,姚波,洪先龙,蔡懿慈. ECOP:一种基于单元行划分的标准单元模式增量布局算法 [J]. 半导体学报.

作者简介:

姚 波 1997 年在浙江大学获学士学位,现为清华大学计算机科学与技术系硕士研究生,从事电子设计自动化(EDA)系统中布图软件的研究和开发工作.

洪先龙 清华大学毕业,1964 年进入清华大学工作. 1988 年起,任清华大学计算机科学与技术系教授. 曾于 1991 年 4 月至 1992 年 6 月,1993 年 6 月至 9 月到美国加州大学伯克利分校做访问学者,在由 E. S. Kuh 教授领导的研究小组工作. 他的研究领域包括 VLSI 布图算法与布图系统的研究,他是国家第二、三代超大规模集成电路计算机辅助设计系统(熊猫系统)的主要设计者.