第 19 卷第 2 期　　半 导 体 学 报　　Vol. 19, No. 2
1998 年 2 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Feb., 1998

SERR: 基于模拟进化技术的性能驱动总体布线算法*

杨昌玲　　严晓浪

(杭州电子工业学院 CAD 所　杭州　310007)

摘要　本文结合 BBL 布图模式, 以提高整个芯片的时间性能为目标, 提出了关于总体布线的基于模拟进化(SE)技术的拆线与重布线算法(简称为 SERR 算法). SERR 算法对传统的顺序布线算法进行了改进, 运用概率准则选择线网进行拆除与重布, 具有绕开局部优化点、得到全局优化解的能力. 实验证明, SERR 算法能够较好地达到优化整个芯片的连线延时性能的目标

EEACC: 7410D, 5120

1　引言

总体布线是 IC 布图中继布局之后的又一重要阶段, 它是基于全局的考虑, 把所有的线网合理地分配到各个布线通道中去, 完成所有线网的拓扑布线

目前, 总体布线算法从其布线顺序来分, 可分为顺序布线与并行布线两大类. 其中并行布线算法是对所有线网同时布线, 但并行算法很难实现, 甚至对于 2-端点的线网也没有一个有效的多项式时间算法[1]; 而许多基于整数规划的同时布线算法虽然能够从整体上对所有线网进行同时布线, 但是这些算法也比较复杂, 而且对于单个的线网来讲, 其结果也许并不优化[9~11]. 顺序布线方法是对所有线网依次进行布线, 该方法对于布线顺序十分敏感, 因为先布线的线网很可能会影响其他线网的布线; 但是, 顺序布线适用于各种布图模式与工艺条件下布线, 所以, 其应用最为广泛[1].

顺序布线方法的基本思想是: 一般地, 根据线网的平面位置关系, 或边界盒的大小, 或端点个数排序, 或线网的延时纷束, 赋予线网一定的权值, 根据线网依权值大小的顺序对线网

* 电科院预研项目
杨昌玲　女, 1969 年出生, 博士研究生, 研究方向为集成电路计算机辅助设计
严晓浪　男, 1946 年出生, 教授, 主要从事集成电路设计自动化方面的科研和教学工作
1996-12-17 收到, 1997-04-03 定稿

www.cnki.net

依次进行布线[2,5]. 当进一步布线已不可行时, 通常采用一个改善的方法, 这就是拆线与重布线技术(Rip-up and Reroute), 也就是将妨碍其他线网布线的导线折开并重新布线, 以保证其他线网的布线顺利进[2,5].

上述总体布线方法的特点是: 第一, 布线、拆线与重布线的过程都是按照线网的关键值(或权值)的高低次序进行的, 虽然能使关键值(或权值)高的线网得到很好的布线结果, 但可能会导致其他线网的布线结果很坏, 例如会使线长、延时等增加, 甚至会变成新的关键线网; 第二, 只选择布线结果很"坏"的线网进行拆除与重布线, 却不考虑拆除布线结果好的线网[1], 这也许会导致布线结果很"坏"的线网由于得不到合适的通道, 即便重新布线也得不到好的布线结果, 同时那些布线结果较好的线网也一直得不到进一步改善, 因而仍不能克服顺序布线方法的弊端

综上所述, 为了改进顺序总体布线算法, 本文结合BBL 布图模式, 以提高整个芯片的延时性能为目标, 采用无约束布线(Unconstrained Global Router)形式[1], 提出了基于模拟进化(SE)技术的拆线与重布线(Rip-up and Reroute)算法(简称为 SERR 算法). SERR 算法不仅选择布线结果很"坏"的线网进行拆除与重布, 而且还以一定的小概率选择少数布线结果较好的线网进行拆除与重布; 其目的是使得这些线网得以进一步优化, 或使其他性能差的线网得到合适的布线通道, 获得较好的布线结果; 从而达到优化整个芯片的连线延时性能的目标 同时, 由于采用无约束布线形式, 因而布线完成率为 100%[1].

2 问题描述

总体布线问题通常作为图模型来研究, 对某一BBL 布图(如图 1 所示), 可定义其通道

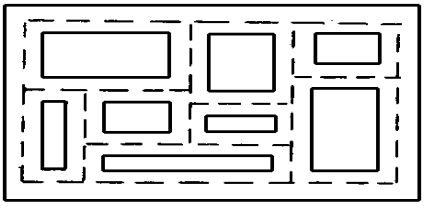


图 1 BBL 模式总体规划图

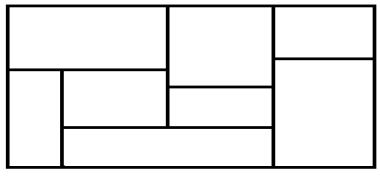


图 2 通道相交(布线)图 $G=(V,E)$

相交图即布线图 $G=(V,E)$, 其中顶点 $\forall v\in V$ 代表一个通道交叉点, 边 $\forall e\in E$ 代表一个通道(如图 2 所示), 对 $\forall e\in E$, 都有相应的容量 $c(e)$ 和长度 $l(e)$. 总体布线就是要求给网表 $N=(N_1,N_2,\cdots,N_n)$ 中的每个线网 N_i, 在图 G 上找出其满足特定目标及约束条件的拓扑布线结构, 即 Steiner 树 T_i; 网表 N 对应的布线树集合为 T.

2.1 性能驱动总体布线目标函数

设线网 $N_i=\{S_0,S_1,S_2\cdots,S_k\}$, i= 1, 2, $\cdots n$, S_0 为源点, S_j $(1\leq j\leq k)$ 为汇点, 其布线树为 T_i, 设 $d(T_i)$ 为其最大延时, 延时等于 $d(T_i)$ 的网点称为关键点 S_c. 令 $M_D(T)=\max\{d(T_i),\ i=1,2,\cdots,n\}$, 则性能驱动的总体布线就是使得芯片的最大延时 $M_D(T)$ 最小化, 即

$$f_1(T)=\min M_D(T) \tag{1}$$

设 $u(e)$ 为通过图 G 中边 $\forall e\in E$ 所代表的通道区的线网数目, 即边 e 的交通量 如果 $u(e)>c(e)$, 则边 e 对应的通道发生了交通量溢出 在BBL 布图模式中, 由于图 G 中 $\forall e\in E$

的容量 $c(e)$ 是通过对布线面积的估算和统计得出的, 这种估算并不很精确, 因而在布线过程中可以允许发生在一定范围内的少量交通量溢出. 所以, 本算法将采用无约束总体布线形式[1]. 令 $z(e)=u(e)/c(e)$, 称 $z(e)$ 为边 e 的载荷. 令 $Z(T)=\max\limits_{e\in E} z(e)$, 则布线结果应使得最大载荷最小化, 即

$$f_2(T)=\min Z(T) \tag{2}$$

SERR 算法还同时考虑了连线总长最短问题, 即使得

$$f_3(T)=\min\left\{\sum_{e\in E} l(e)u(e)\right\} \tag{3}$$

综上所述, 本文以最大延时最小化为第一目标, 最大溢出量最小化为第二目标, 连线总长最短为第三目标, 所以, 总的布线目标函数为

$$f(T)=\eta f_1(T)+\lambda f_2(T)+\gamma f_3(T) \tag{4}$$

其中　系数 η、λ 和 γ 分别视各目标的相对重要性而定.

2.2　连接矩阵

总体布线的状态可通过矩阵 X 表示, 即

$X=(x_{ij})_{n\times m}$, 其中 n 为线网数, m 为通道数.

如果线网 N_i 的布线树占有用了第 j 个通道, 则 $X_{ij}=1$; 否则, $X_{ij}=0$.

第 i 个行向量表示线网 N_i 的布线状态; 第 j 个列向量表示 j 个通道的布线状态.

第 j 个通道的交通量为: $u(e_j)=\sum\limits_{j=1}^{m} X_{ij}$. 如果 $u(e_j)>c(e_j)$, 则第 j 个通道发生了通道量溢出; 设 j 个通道的溢出量为 $v(e_j)$, 设线网 N_i 的总溢出量、连线总长分别为 $v(N_i)$、$l(N_i)$, 则

$$v(e_j)=u(e_j)-c(e_j) \tag{5}$$

$$v(N_i)=\sum_{\forall e_j\in T_i} v(e_j) \tag{6}$$

$$l(N_i)=\sum_{e_j\in T_i} l(e_j) \tag{7}$$

拆除线网 N_i 就相当于从矩阵 X 中减去第 i 行向量的值; 当线网 N_i 布线完毕时, 将相应的行向量值加入到矩阵 X 中去.

2.3　代价矩阵

本文用代价矩阵表示重布线过程中各个线网穿越有关通道的代价, 其形式如下:

$W=(w_{ij})_{n\times m}$, 其中　W_{ij} 表示线网 N_i 穿越第 j 个通道的代价.

$$W=A+B+C+D \tag{8}$$

A —如果线网 N_i 的最大延时 $d(T_i)$ 越大, 则重布时穿越第 j 个通道的代价 c_{ij} 越小, 令

$$c_{ij}=M_D(T)/d(T_i) \tag{9}$$

B —如果第 j 通道的载荷 $z(e_j)$ 越大, 则线网 N_i 重布时穿越该通道的代价 b_{ij} 越小, 令

$$b_{ij}=10\times z(e_j) \tag{10}$$

C —如果 j 通道含有线网 N_i 的网点, 则 $c_{ij}=0$; 否则, c_{ij} 等于第 j 个通道的长度 $l(e_j)$.

D —如果第 j 个通道在线网 N_i 的边界盒内, 则 $d_{ij}=0$; 否则 d_{ij} 为第 j 个通道距线网 N_i 边界盒的最近边的通道序数之值.

线网 N_i 进行重布线的结果是得到代价最小、且关键点 S_c 延时 $d(S_c)\leq M_D(T)$ 的

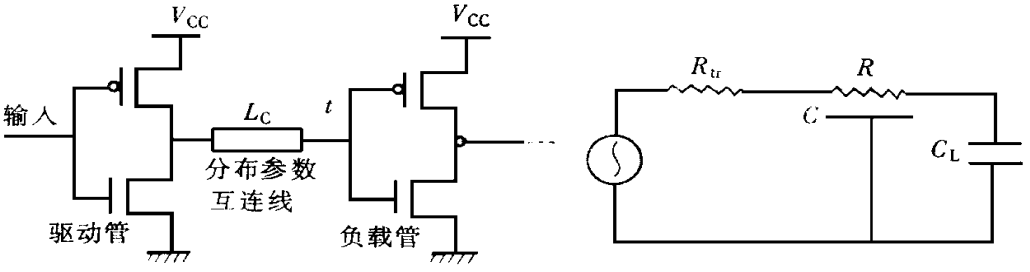
Steiner 树 T_i, 设 T_i 的代价为 $W(T_i)=\sum_{j=1}^{m} x_{ij} W_{ij}$, 则重布线的目标函数为

$$\min W(T_i)$$
$$\text{s.t.}\quad d(S_c) \leq M_D(T) \tag{11}$$

显然, 式(11)与目标函数式(4)所要达到的目标是一致的.

3 连线延时计算模型

在性能驱动的总体布线过程中, 需要不断地计算互连线的延时. 本文采用目前应用较为广泛的互连线延时计算模型[7], 如图 3 所示, 为一个 2 端点的线网, 设 C_L 为负载电容, R_{tr} 为驱动电阻, R、C 分布的 RC 互连线总电阻和总电容, $d(t)$ 是汇点引脚 t 的延迟时间, $d(t)$ 可以



(a) 互连线的参数模型 (b) 互连线的等效电路

图 3 互连线延时计算模型

由下式近似计算[7]:

$$d(t) = \beta R_{tr}(C + C_L) + \alpha R C + \beta R C_L \tag{12}$$

其中 $\beta = 2.21$; $\alpha = 1.02$.

4 SERR 算法

SERR 算法主要有三个部分组成: 即初始布线、拆线与重布线.

4.1 初始布线

初始布线不考虑通道溢出问题, 对所有线网依次进行以最小延时为目标的总体布线. 所采用的算法是文献[12]中的迭代的 k-Steiner 点算法, 不同点是: 每次迭代需要计算的是布线树的延时而不是代价.

在初始布线阶段, 各线网之间是独立布线的, 也就是说, 没有布线顺序可言, 不管线网的布线顺序如何, 其布线结果是相同的: 每个线网的布线结果都是最小延时的 Steiner 树.

4.2 线网的拆除

为了解决初始布线阶段产生的某些通道溢出量过大, 布线不均匀以及连线过长等问题, SERR 提出了一个迭代的拆线与重布线方法. 其基本思想是: 依据一定的概率准则, 拆除一些线网; 然后对拆除的线网依次重新布线. 每次迭代能解决一部分"坏"的线网, 但也许会引起其他线网所分布的通道溢出量增大, 或延时增大, 或连线总长增加. 这时, 需要重复执行上

述拆线与重布线过程

本文选择拆除线网的方法是根据模拟进化(SE)技术进行的. SE 的基本思想是[4, 13]: 对当前这一代的生物群质量进行估值, 通过给较低值的种群一个较高的生存概率, 决定每一种群存活率(即估值越低, 存活率越高); 然后, 对不能存活下去的那些种群进行变异, 使其能够生存, 或以新种群代替它们生存下去

在 SERR 算法中, 线网就是种群. 由 SE 的思想知道, 给每个线网的布线树进行正确的估值是十分重要的, 这是因为正确地选择淘汰者将有利于变异向好的方向进行, 从而加快进化过程

SERR 算法将根据各线网的最大延时、总溢出量和连线总长等三个因素对各个线网的布线树进行估值. 设当前线网为N_i, 其最大延时为$d(T_i)$, 总溢出量为$v(N_i)$, 连线总长为$l(N_i)$, 令 score(i)的为其估值, 则其估值计算公式如下:

$$\text{score}(i) = a_1 g_1(d(T_i)) + a_2 g_2(v(N_i)) + a_3 g_3(l(N_i)) \tag{13}$$

其中 a_1、a_2、a_3 为重要性系数; 函数 g_1、g_2、g_3 的构造如下:

(1) 最大延时 $d(T_i)$ 越小的线网, 被拆除的可能性就越大; 对于 $d(T_i) = M_D(T)$ 的线网, 拆除的可能性则极小, 一般不予拆除. 因此, 令 $g_1(d(T_i)) = M_D(T)/D(T_i)$.

(2) 线网 N_i 的总溢出量 $v(N_i)$ 越大, 则被拆除的可能性就越大. 如果 $v(N_i) \geq 0$, 令 $g_2(N_i) = v(N_i)$; 否则, 令 $g_2(N_i) = 0$.

(3) 连线总长 $l(N_i)$ 越长, 线网 N_i 被拆除的可能性越大, $g_3(l(N_i)) = l(N_i)$.

由于在公式(4)中, 第一、二项非常重要, 而对于连线总长的要求并不十分严格, 因此, 本文中对重要性系数 a_1、a_2、a_3 的取值情况如下:

a_1 取值区间为[1.0, 1.5], a_2 取值区间为[10, 12], a_3 的取值区间为[0.01, 0.03].

对所有线网的布线树估值之后, 接着要选择待拆除的线网

选择拆除线网的传统或直觉做法, 是选择布线结果"坏"的线网(即不满足目标、违反约束等)为待拆除的线网, 这是一种贪婪的做法, 会导致迭代过程陷入局部最优点, 即使增加迭代次数, 也无法"跳"出局部最优点

所以, 本文用 SE 方法, 即概率规则来处理选择拆线问题

首先, 将所有线网的估值进行规格化处理, 使之成为 0～1 之间的数值, 即

$$\text{Nscore}(i) = (\text{score}(i) - \text{misco})/(\text{masco} - \text{misco}) \tag{14}$$

其中 $\text{misco} = \min\{\text{score}(i), i = 1, 2, \cdots n\}$, $\text{masco} = \max\{\text{score}(i), i = 1, 2 \cdots n\}$.

然后, 利用随机数发生器产生 n 个 0～1 之间均匀分布的随机数; 将第 i 个线网的规格化估值 Nscore(i) 与第 i 个随机数 Ran(i) 进行比较: 如果 Nscore(i) > Ran(i), 且最大延时 $d(T_i) < M_D(T)$, 则线网 N_i 将被拆除并重布

如果线网 N_i 重布的结果是 $d(T_i) > M_D(T)$, 则必须再重新布线

由于迭代过程中, 最大延时的趋势是增大的, 而连线总长最短化是一个重要性相对较小的目标, 所以, 当迭代至最大载荷 $Z(T)$ 减小到指定的范围(或者难以进一步减小), 则迭代过程终止. 本文规定: 当连续迭代三次的结果之差在±0.01 范围内时, 迭代终止

用 SE 技术选择拆除线网的方法的优点在于: 使估值越高的线网, 被拆除的机会越大; 即便是估值很小的线网, 也能够以极小的概率被选择拆除, 为其他线网提供有利的通道, 从

www.cnki.net

而得到更合理的布线结果, 最终得到全局优化解.

当上述比较过程结束后, 将有"拆除"标志的线网号按估值的高低顺序送入重布队列, 以备进行重布线.

4.3 重新布线

对所有拆除的线网进行重新布线时, 与初始布线的不同点是将以函数式(11)为重布线目标. 本文采用文献[8]中的CS-Steiner算法, 即: 对当前线网N_i, 其关键汇点为S_c, 先在汇点集合$\{N_i \setminus S_c\}$上构造出最小代价的Steiner树T_i; 然后将具有最大延时的关键汇点S_c插入到T_i中并使得: 源点S_0到S_c的延时最小.

5 实验结果

本算法已在HP9000/375工作站上用C语言编制实现, 对几个例子进行了实测, 并与文献[5]中的算法结果进行了对比, 结果如表1所示, $f_1(T)$、$f_2(T)$和$f_3(T)$分别为最大延时、最大载荷和连线总长(见2.1中式(1)、(2)、(3)). 实验表明, SERR算法的确比文献[5]的算法大大提高了芯片的延时性能, 降低了最大载荷, 同时也兼顾了连线总长最短化的要求, 说明SERR算法具有很强的寻找全局最优的能力.

表1 用SERR算法与文献[5]的算法结果对比(延时单位: ns, 线长单位: mm)

例子	单元数	线网数	SERR算法结果						文献[5]的算法结果		
			初始布线结果			最后布线结果					
			$f_1(T)$	$f_2(T)$	$f_3(T)$	$f_1(T)$	$f_2(T)$	$f_3(T)$	$f_1(T)$	$f_2(T)$	$f_3(T)$
1	7	18	1.329	1.23	69	1.329	0.91	61	2.030	0.93	52
2	10	32	0.545	1.47	98	0.545	1.05	82	1.343	1.07	68
3	20	50	1.078	1.29	134	1.078	0.96	116	3.208	1.03	98
4	47	128	5.088	2.13	884	5.088	1.04	803	8.258	1.09	679

6 结论

本文提出了关于总体布线的基于模拟进化(SE)技术的拆线与重布线算法(简称为SERR算法), 对传统的顺序布线算法进行了改进. SERR算法运用概率准则选择线网进行拆除与重布, 具有绕开局部最优点、得到全局优化解的能力. 实验证明, SERR算法能够较好地达到优化整个芯片的连线延时性能的目标.

SERR算法不仅可以应用于BBL布图模式, 而且能够应用于标准单元、门阵列等其他的布图模式.

参 考 文 献

[1] T. Lenguer, Combinatorial algorithms for integrated circuit layout, Kluwer Academic publishers, 1993, Chapter8.

www.cnki.net

[2] M. Sarafzadeh *et al*., IEEE Trans on CAD, 1994, **13**(1): 38～47.
[3] 庄文君, 李玉兴, 集成电路布图设计自动化, 上海交通大学出版社(1986 年), 第七章第二节.
[4] S. Prasitjutrakul, William J. Kubitz, A timing-driven global router for custom chip design, Proc 27th DAC, 1990, 48～51.
[5] R. Nair, IEEE Trans on CAD, 1985, **7**(5): 165～172.
[6] Youn-Long Lin *et al*., IEEE Trans on CAD, 1989, **8**(10): 1108～1114.
[7] T. Sakurai, IEEE Jounal of Solide-State Circuits, 1983, **18**(4): 418～416.
[8] K. D. Boese *et al*., High-perfomance routing trees with identified critical sinks, Proc 30th DAC, 1993, 182～187.
[9] S. Mehrotra *et al*., Perfomance driven global routing and wiring rule generation for high speed PCBs and MCMs, Proc 32nd DAC, 1995, 381～387.
[10] Jin Huang, Xian-Long Hong *et al*., An efficient timing-driven global routing algorithm, Proc 30th DAC, 1993, 56～62.
[11] R. C. Carden Ⅳ *et al*., IEEE Trans on CAD, 1996, **15**(2): 208～216.
[12] J. Griffith *et al*., IEEE Trans on CAD, 1994, **13**(11): 1351～1365.
[13] 刘勇, 等著, 非数值并行算法—遗传算法, 北京: 科学出版社 (1995 年), 第五章.

SERR: A Simulated Evolution Perfomance-Driven Global Router

Yang Changling and Yan Xiaolang

(*Hangzhou Institute of Electronic Engineering, Hangzhou* 210007)

Received 17 December 1996, revised manuscript received 3 April 1997

Abstract We propose a simulated evolution rip-up and reroute perfomance-driven global routing algorithm called SERR algoritnm for building block layout. SERR algorithm modifies traditional sequential global router using probability criterion to choose nets to rip-up and reroute, and it can avoid local optimal solutions and find the global optimal one. The experiment results show the effectiveness of SERR algorithm for perfomance-driven global routing.

EEACC: 7410D, 5120

www.cnki.net