第16卷第3期　　半 导 体 学 报　　Vol. 16, No. 3
1995年3月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Mar., 1995

一个以时延优化为目标的力指向 Steiner 树算法*

洪先龙
（清华大学计算机系　北京　100084）

摘要　本文提出了一种用于总体布线的以时延优化为目标的力指向 Steiner 树算法. 它在构造 Steiner 树时同时考虑使线网总长和从源点到漏点的路径长度最小，以期得到时延最小的 Steiner 树. 文中首先给出多端线网连线延迟模型，并导出其上界. 基于这个时延模型，提出了力指向 Steiner 树算法. 算法已用 C 语言在 Sun 工作站上实现，并用于以性能优化为目标的总体布线中. 实验表明，算法十分有效.

CCACC：7410D，5120

1　引言

由于集成电路工艺的发展，芯片面积不断增大、线条尺寸不断减小，连线所造成的延迟已逐渐成为主要因素. 在亚微米工艺下，连线电容和电阻已不能忽略. 但传统的总体布线算法只考虑连线总长. 对多端线网，连线总长最小不一定能保证线网延迟最小[1]. 如果我们在布图时希望优化整个芯片性能，即使得最大延迟最小，就必须总体布线时考虑把性能优化作为目标.

Steiner 树算法是总体布线算法的基础. 绝大部分已发表的 Steiner 树算法都是把连线总长最小作为目标. 只有少数几篇文章考虑了时延约束. 文献[1]给出了一个多端线网延迟模型，并被用于在全定制芯片总体布线时进行时延分析. 文献[2]中提出了一种以半径为界的 Steiner 树算法，如设离源点最远的那个漏点到源点的距离为 R，则算法求得的 Steiner 树的总长不超过最佳的 Steiner 树总长的 $2(1+1/\varepsilon)$ 倍，它的从源点到漏点的最长路径为 $O(1+\varepsilon)R$. 可以调节参数 ε 控制源点到 漏点的路径长度. 但当 ε 足够小时，导致了连线总长太大. 而连线总长恰恰是影响线网延迟的主要因素. 此外，文献[2]中也未给出延迟模型的细节.

本文首先从多端线网的 Elmore 延迟模型出发[3]，导出它的上界. 这个上界是连线总长和源点到漏点的路径长度的函数. 基于这个模型，我们设计了一个构造性的力指向 Steiner 树算法. 在构造树的过程中，算法同时考虑使连线总长和源点到漏点的路径长度最小.

* 国家自然科学基金资助项目
洪先龙　男，1940年生，教授，主要从事集成电路计算机辅助设计方面的科研及教学工作
1993年10月25日收到初稿，1994年1月12日收到修改稿

2 时延模型和问题的表达

我们把如图 1 所示的电路看成是一个具有晶体管导通电阻 R_d 的电压源、分布电阻 R_j 和电容 C_j 的 RC 传输线以及负载电容 C_{Lj}. 设 S 为源点，t 为漏点. 我们用基于 Elmore 分布 RC 延迟模型[4,5]来计算从 S 到大的延迟 del (s,t). 树结构 T 的 Elmore 延迟定义为

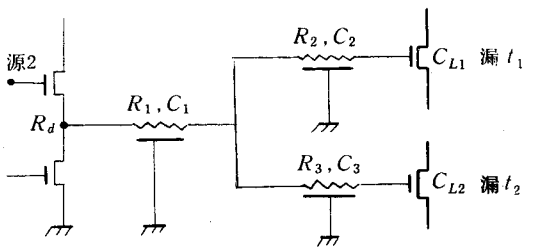

图 1　分布 RC 延迟模型

$$\mathrm{del}(s,t) = \sum_{k \in T} R_{kt} C_k \qquad (1)$$

其中 R_{kt}是与漏 t 共享的公共路径上的电阻，C_k 是节点 k 上的电容，k 是 T 中的一个节点. 在布线过程中精确计算 del(s,t)不仅计算量太大，而且在算法结束前我们还不能知道 s 到 t 的具体路径. 为此，我们导出它的一个上界 Del(s,t)来代替 del(s,t)用于布线过程中的时延分析. 设 R(s,t)是从 s 到 t 路径上的总电阻，由于对所有 T 中的 k，$R_{kt} \leqslant R(s,t)$，因此有

$$\mathrm{del}(s,t) \leqslant \mathrm{Del}(s,t) = R(s,t) \sum_{K \in T} C_K = R(s,t) C_{\mathrm{total}} \qquad (2)$$

其中 C_{total}为 T 的总电容(包括连线和负载电容).

如记 $L(s,t)$为 S 到 t 路径的长度，W 为线网总连线长度，R_d 为导通电阻，C_{in}为线网总的负载电容，r_1 的 c_1 分别为单位连线的电阻和电容，则有

$$R(s,t) = R_d + r_1 L(s,t)$$

$$C_{\mathrm{sotal}} = C_{in} + c_1 W$$

将它们代入(2)，得到

$$\mathrm{del}(s,t) \leqslant \mathrm{Del}(s,t) = (R_d + r_1 L(s,t)(C_{in} + c_1 W) \qquad (3)$$

下面我们给出以时延优化为目标的 Steiner 树问题的表述.

假设总体布线在一个二维曼哈顿平面上进行，总体布线空间是图 $G=<V,E>$，其中 $V=\{v_1, v_2 \cdots v_n\}$表示顶点集合，它是一个二维阵列. $E=\{e_1, e_2 \cdots e_m\}$表示相邻顶点间的边集合. 又假定 $T \leqslant V$ 是一个具有单个源点和多个漏点的信号网引线端集合，S 和 t 分别表示源点和线网关键漏点(即与 S 有最长路径的漏点). 以时延优化为目标的 Steiner 树问题是在图 G 上求解一棵树，它连接 T 中所有顶点，并且有最小的 del(s,t).

3 构造性力指向(CFD)Steiner 树算法

这是一个启发式算法. 由(3)我们知道，Del(s,t)是连线总长 W 和 S 到 t 的路径长度 $L(s,t)$的函数. CFD 算法在构造 Steiner 树过程中力图使 W 和 L 最小化.

3.1 若干定义

定义 1　CFD 算法在 G 上构造一棵连接 T 中所有顶点的树，并使 Del(s,t)尽可能小.

因为 Del(s,t)是 del(s,t)的上界，根据定义 1 构造的 Steiner 树将使 del(s,t)尽可能小.

定义 2　T 中所有顶点为 CFD 算法过程中的起始节点，正在生长中的节点 $v_j \in V$ 为当

前节点，算法启动时 $v_j \in T$ 是当前节点.

定义 3 设 $v_j \in V$ 是一个当前节点，它的坐标分别为 x_j 和 y_j，则对应 v_j 的带权重心(WMP)C_j 定义如下：

设 $R \leqslant V$ 为当前节点集，$v_j \in R$ 且 $i \neq j$，

$$\begin{aligned} d_{ij} &= (x_j - x_j)^2 + (y_i - y_j)^2 \\ D_{ij} &= 1/d_{ij} \\ x_{c_j} &= \sum_{i \neq j, Vi \in R} (x_i D_{ij}) / \sum_{i \neq j, Vi \in R} D_{ij} \\ y_{c_j} &= \sum_{i \neq j, Vi \in R} (y_i D_{ij}) / \sum_{i \neq j, Vi \in R} D_{ij} \end{aligned} \tag{4}$$

其中 x_{c_j} 和 y_{c_j} 分别是 v_j 的带权重心 C_j 的 x 和 y 坐标. 权 D_{ij} 的作用是使 v_j 的 WMP 更接近哪些离 v_j 近的节点. 值得注意的是，v_j 的 WMP 不一定落在 Steiner 树上，它只用来确定当前节点的长方向.

在 Steiner 树构造过程中，CFD 算法在每一步处理一个当前节点，使它向一个方向生长一段(一个边长 e). 对每一个当前节点，通常有 4 个生长方向：上、下、左和右. 我们希望构造有尽可能小的连线总长 W 和源点 S 到 t 的路径长度 L 的 Steiner 树. 为此，我们定义与 WMP 相关的生长方向和与源点 S 相关的生长方向. 沿着与 WMP 相关方向生长有利于减小 W[6]，沿着与 S 相关方向生长有利于减小 L. 我们对不同方向设置不同的权值.

定义 4 设 WMP 是当前节点 v_j 的带权重心，e 是 v_j 的邻接边，则与 WMP 相关的权值由下式确定：

$$W_{\mathrm{WMP}}(e) = \alpha_1 L(v_j, \mathrm{WMP}) \tag{5}$$

其中

$$\alpha_1 = \begin{cases} 1 & \text{如果 } e \text{ 在朝向 WMP 的方向上} \\ 5 & \text{如果 } e \text{ 在背离 WMP 的方向上} \\ 3 & \text{其它} \end{cases} \tag{5}$$

$L(v_j, \mathrm{WMP})$ 为 v_j 与 WMP 之间的最短路径长度.

定义 5 设 S 是线网的源点，e 是 v_j 的邻接边，则与源点 S 相关的权值由下式确定：

$$W_s(e) = \alpha_2 L(v_j, s) \tag{6}$$

其中

$$\alpha_2 = \begin{cases} 1 & \text{如果 } e \text{ 在朝向 } S \text{ 的方向上} \\ 5 & \text{如果 } e \text{ 在背离 } S \text{ 的方向上} \\ 3 & \text{其它} \end{cases}$$

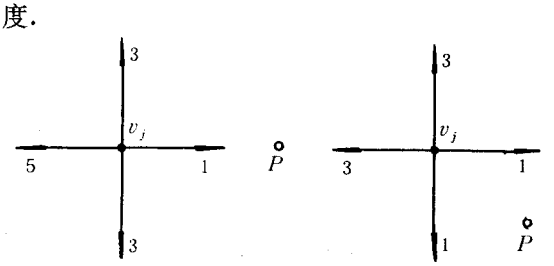

图 2 W_{WMP} 和 W_s 示意图

$L(v_j, S)$ 为 v_j 与 S 有最短路长度. W_{WMP} 和 W_s 的定义如图 2 所示，图中 P 表示 WMP 或 S.

根据(3)，我们应同时考虑上述两个方向以使时延最小化. 我们引入合成方向并定义相应的权值 W_{cd}：

$$W_{cd}(\in) = g_1 W_{\mathrm{WMP}}(e) + g_2 W_s(e) \tag{7}$$

$$g_1 + g_2 = 1$$

对当前节点 v_j，通常它有 4 个邻接边，选择其中 $W_{cd}(e)$最小的那个 e 作为生长方向. 它既考虑了有利于减少连线总长 W，又考虑了有利于减少源和漏路径长度 L. 我们可以调节 g_1 和 g_2 的值以改变两个权值在 W_{cd}中的贡献.

3.2 CFD 算法

设总体布线图 $G=<V,E>$，线网引线端集合为 T，当前节点集合为 R，CFD 算法过程叙述如下：

(1)把 T 中所有顶点映射到图 G 上，它们为超始节点；

(2)$R \Leftarrow T$；

(3)While $|R| \neq 1$ do

```
{ for each v_j ∈ R
  { 对所有与 v_j 邻接的边 e 计算 W_cd(e)；
    选择最小权值的方向为生长方向；
    在所选择的方向上生长一段并更新树数据和当前节点集 R；
    如果当前节点的移动是前一次移动的后退，则删去此段路径；
    如果生长这一段后搭接上另一部分子树，执行节点合并过程
    并更新 R 及选择新的当前节点；
    }
}
```

(4)检查已生成的树中是否存在环，如有，则删去环中一段；

(5)输出 CFD 方向生成的树；

在节点合并过程中，如两个要合并的节点中有 一个不是当前节点，则把那个当前节点变成非当前节点，并从 R 中删掉它. 如果两个都是当前节点，则合并两个成为一个.

3.3 CFD 算法的计算复杂性分析

对给定的图 $G=<V,E>$和线网顶点集 T，在算法的每一步，最多只可能有 $|T|$ 个当前节点. 因此，在每一步权值计算的次数是 $O(|T|)$. 在生长过程中，最坏情况是 E 中每一边都遍历过一次. 因此，算法计算复杂性上界是 $O(|T| \cdot |E|)$.

4 实验结果

CFD 算法用 C 语言在 Sun 工作站上实现. 我们首先测试了 g_1 和 g_2 对结果的影响，并选择了最好的 g_1 和 g_2 值. 图 3 给出了 4 组不同 g_1 值的结果. 表 1 列出了这 4 棵树的总长和延迟等数据. 图 4 给出了连线总长、关键路径(S 到 t_1)长度和 t_1 端延迟等相对于 g_1 的变化曲线. 从图 4 可以看到，连线总长 W 随 g_1 增大而单调递降，同时关键路径长度 $L(s,t)$单调上升. 但 t_1 点的延迟值变化不是单调的，它是 W 和 $L(s,t_1)$的函数. 从实验数据分析，g_1 取 0.5 到 0.6 之间比较合适. 在 CFD 算法中取 $g_1=0.5$. 图 5 给出了当 $g_1=1.0$ 时用 CFD 算法求得的一个 14 顶点的 Steiner 树，即按连线总长度最小化求得的 Steiner 树. 它的连线总长为 9680，最佳 Steiner 树的结果为 9444，误差只有 2.4%.

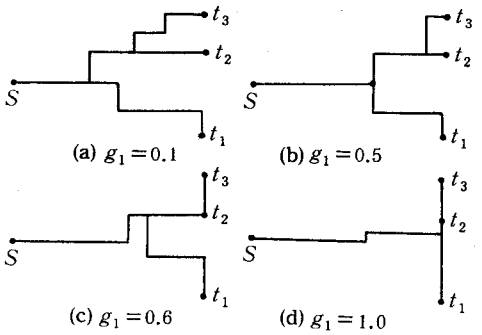

图 3 不同 g_1 对结果的影响

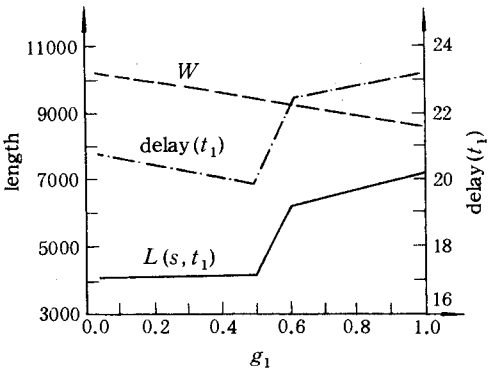

图 4 W、$L(s,t_1)$和 del(s,t_1)对 g_1 的变化曲线

表 1

g_1	连线总长	关键路径长度	t_1 延迟
0.1	10053	4141	20.89
0.5	9326	4141	19.96
0.6	9139	6335	22.69
1.0	8750	7213	23.28

我们用 CFD 的算法测试了来自工业界的三个试验电路 C_2、C_7 和 S_{13207}. 我们同时开发了另一个以性能优化为目标的 Steiner 树算法—基于 Dreyfus-Wangner 的迭代算法(IDW)[7,8]. 它采用了动态规划和非线性优化技术，采用了枚举策略. 因而可获得最佳解. 但其时间复杂性是指数增长，不能求解顶点数多的 Steiner 树. 在我们的总体布线程序中，用 IDW 求解顶点数不超过 5 的线网，用 CFD 求解顶点数大于 5 的线网. 表 2 列出了用 IDW 和 CFD 求解 C_2、C_7 和 S_{13207}三个电路的结果. 其中方案 1 采用了 IDW 计算不超过 5 个顶点的线网和 CFD 计算其余网. 方案 2 采用了 CFD 计算全部线网. "误差"表示两个方案结果的差别. 从表 2 可见，无论从连线总长或者延迟值总和，这两个方案的误差分别不超过 1.6%和 1.2%. 它表明 CFD 算法是一个十分有效而实用的算法.

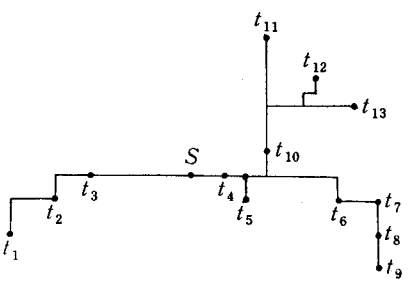

图 5 14 顶点的 Steiner 树

表 2

电路名		C_2	C_7	S_{13207}
线网数		647	2076	4158
连线总长	方案 1	472764	2189763	6347409
	方案 2	478074	2216938	6485923
	误 差	1.1%	1.4%	1.6%
延 迟	方案 1	695.4	2786.6	6393.9
	方案 2	700.2	2846.4	6480.2
	误 差	0.4%	0.6%	1.2%
cpu 时间	方案 1	0.7 秒	2.1 秒	4.9 秒
	方案 2	0.9 秒	3.0 秒	8.7 秒

5 结论

本文提出的构造性力指向 Steiner 树算法，它可以求解以时延最小为目标的 Steiner 树. 它已作为我们开发的以性能优化为目标的总体布线程序的一个重要组成部分[9]. 算法具有计算复杂性低和求解质量高的优点. 实验表明它是一个十分有效而实用的算法.

致谢 本文工作是作者在访问美国加州柏克利大学时完成的，葛守仁教授给予了关心和指导，他的研究生薛天雄参加了本文工作.

参　考　文　献

[1] S. Prasitjutrakul and W. J. Kubitz, Proc. of ICCAD. 90 1990, pp. 48.
[2] Jason Cong, A. Kahng, G. Robins, M. Sarrafzadeh and C. K. Wong, IEEE Trans. on CAD, 1991.
[3] W. C. Elmore, Journal of Applied Physics, 1948, **19**(1): 55.
[4] H. B. Bakonglu, Reading, MA, Addison-Wesley, 1990.
[5] J. Rubinstein, P. Penfield, Jr. and M. A. Horowitz, IEEE Trans. on CAD, CAD-2, 3, 1983, pp. 202.
[6] C. S. Ying, J. S. L. Wong, X. L. Hong and E. Q. Wang, Proc. of The First EDAC, 1990.
[7] X. L. Hong, T. X. Xue, E. S. Kuh and C. K. Cheng, Proc. of 30th DAC 1993, pp. 177.
[8] S. E. Dreyfus and R. A. Wagner, Network I, 1972, pp. 195.
[9] J. Huang, X. L. Hong, C. K. Cheng and E. S. Kuh, Proc. of 30th DAC 1993, pp. 596.

A performance-Driven Steiner Tree Algorithm Using Constructed Force Directed Approach for Global Routing

Hong Xianlong
(*Department of Computer Science and Technology Tsinghua universith, Beijing* 100084)
Received 25 October 1993, revised manuscript received 12 January 1994

Abstract This paper presents a performance-driven steiner tree algorithm for global routing. It creates the steiner tree considering both the minimization of total wire length and distance of the path from the source to sink of the given net in order to obtain the steiner tree with minimum timing delay. A timing model for multi-terminal net is given first, then its upper bound is deducted. Based on this model a constructed force directed (CFD) steiner tree algorithm is proposed. The CFD algorithm is implemented on Sun workstation by C language and has been used in our performance-driven global router. Experiments are given to demonstrate the effectiveness of the algorithm.

CCACC: 7410D, 5120