第 20 卷第 1 期　　半 导 体 学 报　　Vol. 20, No. 1
1999 年 1 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Jan., 1999

基于 Sakurai 模型的时延驱动 Steiner 树算法*

鲍海云　洪先龙　蔡懿慈　乔长阁

(清华大学计算机科学与技术系　北京　100084)

摘要　时延驱动的 Steiner 树构造算法是时延驱动总体布线的基础. 本文首先简介了求解最佳 Steiner 树的 Dreyfus-Wagner 算法. 随后通过引入 Sakurai 时延模型, 提出了直接基于 Sakurai 模型的提高线网时延性能的时延驱动 DW 算法. 当集成电路工艺的特征宽度较小时, 该算法求得的 Steiner 树中关键点的时延值, 明显小于 IDW 和 CFD 算法的结果.

EEACC: 7410D, 5120

1　引言

随着集成电路工艺的飞速发展, 芯片面积正在不断增大, 线条尺寸不断减小, 布线的层数增多, 连线造成的延迟已不能忽略[1]. 在深亚微米工艺下设计芯片时, 连线造成的延迟已逐渐在总延迟中占了很大的比例, 甚至超过了门延迟[2]. 传统的总体布线算法仅以总的线长为优化目标, 而对于多端线网来说, 总的线长最短并不意味着时延最短[3]. 因此如果希望优化整个芯片的电性能, 就必须在布局和总体布线时考虑电性能的优化.

求解 Steiner 树的算法是多数总体布线算法的核心算法[4,5], 它是一个多端网络最基本的布线问题到图论问题的直接转化. 总体布线算法是从整体上控制所有总体布线树(Steiner 树)的性质, 而每个总体布线树的求解就必须利用求解 Steiner 树的算法. 因此要构造时延驱动的总体布线算法, 就必须首先构造以时延为优化目标的 Steiner 树构造算法. 为此清华大学的洪先龙教授在文献[6]中提出了基于 Dreyfus-Wagner 的时延驱动 Steiner 树迭代构造算法(简称 IDW 算法), 在文献[7]中提出了构造性力指向的 Steiner 树构造算法(简称 CFD 算法). CFD 算法与 IDW 算法相比, 大大加快了求解速度, 同时线网时延性能有所降低.

IDW 算法中使用的时延模型是简化的 Elmore 时延模型[8]. 该模型计算便捷, 但有一定的局限性, 对形状较复杂线网的时延估计会有较大的误差. 另外当线网中的节点数较多(大

* 国家"九五"科技攻关(项目编号: 96-738-01-08-03)和高等学校博士学科点专项科研基金资助项目(项目编号: 96000330)
鲍海云　男, 1972 年出生, 博士研究生, 从事大规模集成电路自动布线算法的研究
洪先龙　男, 1940 年出生, 教授, 目前从事计算机设计自动化的教学和科研工作
蔡懿慈　女, 1960 年出生, 副教授, 目前从事大规模集成电路计算机辅助设计的教学和科研工作
1997-10-25 收到, 1998-02-23 定稿

于 6) 时, DW 算法的求解时间将变得相当长. 本文将 Sakurai 时延模型用于 Dreyfus-Wagner 算法中, 并针对 Sakurai 时延模型的特点, 避免了重复求解, 实现了速度、性能均优于 DW 和 CFD 算法的时延驱动 Steiner 树的构造算法.

2 应用 Sakurai 时延模型的时延驱动 Dreyfus-Wagner 算法

2.1 Dreyfus-Wagner 算法

Dreyfus-Wagner 算法(简称 DW 算法)[9]应用动态规划技术求解 Steiner 树, 它反复使用两个递推公式, 从所有可能的 Steiner 树中, 找出连通给定顶点集 Z 的最小(长度最短) Steiner 树. 若用 $p(v,w)$ 表示图 G 中点 v 到 w 的最短距离, K 表示 G 对应点集的子集, $S_v(K\cup\{v\})$ 表示 G 中 $K\cup\{v\}$ 的最小 Steiner 树的长度, $P_v(K\cup\{v\})$ 表示 G 中 $K\cup\{v\}$ 的 v 度大于 1 的最小 Steiner 树的长度, 则这两个递推公式分别为:

$$P_v(K\cup\{v\})=\min\{S_v(K'\cup\{v\})+S_v(K-K'\cup\{v\})\mid K'\subset K\text{ 且 }K'\neq\Phi\} \quad (1)$$

$$S_v(K\cup\{v\})=\min\{\min\{p(v,w)+S_w(K)\mid w\in K\},\min\{p(v,w)+P_w(K\cup\{w\})\mid w\notin K\}\} \quad (2)$$

用 DW 算法求解最短 Steiner 树的整个过程如下:

(1) 初始化: 计算 G 中所有顶点之间的最短路径 $p(v,w)$;

(2) 递推: 下面的过程从 $k=1,2,3,\ldots,|Z|-1$ 依次处理

· $\forall K\subset Z$, $|K|=k$, $\forall v\in Z/K$, 依照公式(1), 寻找 $P_v(K\cup\{v\})$.

· $\forall K\subset Z$, $|K|=k$, $\forall v\in Z/K$, 依照公式(2), 寻找 $S_v(K\cup\{v\})$.

2.2 Sakurai 时延模型

要设计以时延为优化目标的 Steiner 树构造算法, 就必须对不同 Steiner 树的时延进行估计, 也就不可避免地要用到时延模型. 对于总体布线来说, 时延模型的主要任务是确定如何连线可以使时延短, 而不是必须得到准确的时延值. 因此对时延模型的要求中, 精确是相对次要的; 而与实际时延长短的对应关系, 或说一致性则是比较重要的; 计算简单、快捷也是比较重要的因素. 因此我们选用了 Sakurai 时延模型[10].

在 Sakurai 时延模型中, 一条连线被看作具有分布电阻 r_e 和电容 c_e 的传输线. 该传输线的输入是内阻为 R_s 的单位电压源, 电容负载为 C_z. 这种模型与布线问题中的实际情况基本相同. 文献[11]导出了该模型下, 经过连线后的输出 V_o 从 0 增加到 $0.9V_{DD}$ 的时延 T_D:

$$T_D=\beta R_s(C_e+C_z)+\alpha r_e c_e+\beta r_e C_z,\text{ 其中 }\alpha=1.02,\beta=2.21$$

如果规定 T_{DZ} 为 V_o 从 0 增加到 $0.7V_{DD}$ 的时延[12], 则 $\alpha=0.59$, $\beta=1.21$. 如果规定 T_{DZ} 为 V_o 从 0 增加到 $0.62V_{DD}$ 的时延[13], 则 $\alpha=0.5$, $\beta=1.0$. 对于树型多端连线结构, 从源到各结点的时延可根据其前驱结点时延确定:

$$T_D(s)=\beta R_s C_s$$

$$T_D(w)=T_D(v)+\alpha\hat{r}\hat{c}L_{vw}^2+\beta\hat{r}L_{vw}C_w$$

其中 s 为线网的源; R_s 为源的输出电阻; C_s 为整个线网的负载电容; v 为 w 的前驱结点; L_{vw} 为 v 到 w 的连线长度; C_w 为结点 w 之后的总电容; $\hat{r}$ 为单位线长的电阻值; $\hat{c}$ 为单位线长的电容值. Sakurai 时延模型比 Elmore 时延模型更精确, 能够较正确地反映 Steiner 树的形

状对时延的影响, 适用面更广, 而计算复杂度增加不大, 所以用于时延驱动的总体布线中是较为合适的.

2.3　分解 Sakurai 时延公式

分解 Sakurai 时延公式, 是为了分离出 Steiner 树各部分对某个结点的时延贡献, 以确定每一步的优化目标, 即用 Sakurai 时延模型的时延最小值, 代替 DW 算法中的线长最小值. 公式可将 Sakurai 时延变形如下:

$$\begin{aligned}T_D(t) &= T_D(s) + \alpha\hat{r}\hat{c}\sum_{xy\in path(s,t)} L_{xy}^2 + \beta\hat{r}\sum_{xy\in path(s,t)} L_{xy}C_y\\ &= \beta R_s C_s + \alpha\hat{r}\hat{c}\sum_{xy\in path(s,t)} L_{xy}^2 + \beta\hat{r}\sum_{xy\in path(s,t)} L_{xy}C_y\\ &= [\beta R_s C_{s-w} + \alpha\hat{r}\hat{c}\sum_{xy\in path(s,w)} L_{xy}^2 + \beta\hat{r}\sum_{xy\in path(s,w)} L_{xy}C_{y-w}]\\ &\quad + [\beta(R_s + \hat{r}\sum_{xy\in path(s,w)} L_{xy})C_w + \alpha\hat{r}\hat{c}\sum_{xy\in path(w,t)} L_{xy}^2 + \beta\hat{r}\sum_{xy\in path(w,t)} L_{xy}C_y]\end{aligned}$$

其中 x 为 y 的前驱结点; xy 表示连接 x 和 y 的边, path(s, t) 表示 s 到 t 的路径, 即连接点 s 和 t 的边的集合; C_{v-w} 表示 v 点以下 w 点以上部分的电容值, 可以看出用方括号分隔出的前一部分与 w 以下的 Steiner 树没有任何关系; 而后一部分中只有 C_w 的系数与非 w 以下的部分有关. 若令 $R_{sw} = R_s + \hat{r}\sum_{xy\in path(s,w)} L_{xy}$, 设 t 为关键点(要控制时延的结点), 若 $w \in$ path(s, t) 则求解 w 以下 Steiner 树时, 只需以下式表示的 $T_D(t)$ 后半部分为优化目标即可:

$$T_d(w, t) = \beta R_{sw} C_w + \alpha\hat{r}\hat{c}\sum_{xy\in path(w,t)} L_{xy}^2 + \beta\hat{r}\sum_{xy\in path(w,t)} L_{xy}C_y \tag{3}$$

从该式中可以看出, 可将 w 看作是其下树的假想源, 其驱动电阻为 R_{sw}.

而对 $\forall v \notin$ path(s, t), 则由 $T_D(t)$ 计算式的前一部分知: v 以下的 Steiner 树对 $T_D(t)$ 的影响仅仅表现在两个电容项 C_{s-w} 和 C_{y-w} 上, 因此在求解 v 以下的 Steiner 树时, 只需使其总电容最小, 即总线长最小即可.

扩展到多个关键点的情况, 可以用向量 t$=(t_1, t_2, \cdots, t_n)$ 表示所有关键点; 用 $S_v(\mathbf{K}\cup\{v\}, R_{sv})$ 表示对应电阻值 R_{sv} 的 G 中 $\mathbf{K}\cup\{v\}$ 的时延最小 Steiner 树, 其中 v 是该树中距源 s 最近的点; 以 $P_v(\mathbf{K}\cup\{v\}, R_{sv})$ 表示再附加上条件"v 的度大于 1"后的时延最小 Steiner 树; 用 $T_d(S_v, t_i)$ 表示 Steiner 树 S_v 中 v 到 t_i 的时延(若 t_i 不在 S_v 中则该值为 0), 其中 v 是该树中距源 s 最近的点(假想源), 并令时延向量(黑体字母表示向量):

$$\mathbf{T}_d(S_v, \mathbf{t}) = (T_d(S_v, t_1), T_d(S_v, t_2), \cdots, T_d(S_v, t_n))$$

则对应 DW 算法的(1)、(2)两式, $\mathbf{T}_d(S_v, \mathbf{t})$ 分别有如下传递关系:

$$\begin{aligned}\mathbf{T}_d(P_v(\mathbf{K}\cup\{v\}, R_{sv}), \mathbf{t}) = \min\{&\mathbf{T}_d(S_v(\mathbf{K}\cup\{v\}, R_{sv}), \mathbf{t}) + \beta R_{sv}C_{\mathbf{K}-\mathbf{K}'\cup\{v\}}\mathbf{I}_{\mathbf{K}-\mathbf{K}'}\\ &+ \mathbf{T}_d(S_v(\mathbf{K}-\mathbf{K}'\cup\{v\}, R_{sv}), \mathbf{t}) + \beta R_{sv}C_{\mathbf{K}'\cup\{v\}}\mathbf{I}_{\mathbf{K}'}\}\end{aligned}$$

(4) 对应(1)式

$$\begin{aligned}\mathbf{T}_d(S_v(\mathbf{K}\cup\{v\}, R_{sv}), \mathbf{t}) = \min\{&\min\{\mathbf{T}_d(S_w(\mathbf{K}), R_{sv}) + \hat{r}L_{vw}), \mathbf{t})\\ &+ (\beta\hat{r}L_{vw})C_w + \alpha\hat{r}\hat{c}L_{vw}^2)\mathbf{I}_{\mathbf{K}} \mid w \in \mathbf{K}\}\\ &\min\{\mathbf{T}_d(P_w(\mathbf{K}\cup\{w\}), R_{sv}) + \hat{r}L_{vw}), \mathbf{t})\\ &+ (\beta\hat{r}L_{vw})C_w + \alpha\hat{r}\hat{c}L_{vw}^2)\mathbf{I}_{\mathbf{K}} \mid w \notin \mathbf{K}\}\end{aligned}$$

(5) 对应(2)式

其中 $\mathbf{I}_{\mathbf{K}} = (i_{1\mathbf{K}}, i_{2\mathbf{K}}, \cdots, i_{n\mathbf{K}})$, 若 $t_j \in \mathbf{K}$ 则 $i_{j\mathbf{K}} = 1$, 否则 $i_{j\mathbf{K}} = 0$. 请注意用于向量的 min 函数可

以根据实际需要赋予不同的含义

2.4 利用已求得的部分解加快求解过程

将DW算法的迭代公式中的线长对应地替换为Sakurai时延公式,就能够求得使关键节点时延最短的Steiner树,但与以线长为优化目标的DW算法有所不同.在原先的算法中,$P_v(\mathbf{K})$和$S_v(\mathbf{K})$对应的最小Steiner树的形状只与$\mathbf{K}$和v有关,且最小值具有传递性,所以可以通过自下而上求解,达到避免重复构造子Steiner树的目的.而由(3)~(5)式知,考虑时延后$P_v(\mathbf{K})$和$S_v(\mathbf{K})$对应的Steiner树的形状与R_{sv}有关,分别记为$P_v(\mathbf{K}, R_{sv})$和$S_v(\mathbf{K}, R_{sv})$.并且时延的最小值不再具有传递性,不能简单地利用已求解过的Steiner树.这样当结点数较多时,运算量将远远大于原DW算法.为加快求解过程,要尽可能地利用已求得的部分解.可以证明若$R_{sv1}<R_{sv2}$,且$S_v(\mathbf{K}, R_{sv1})=S_v(\mathbf{K}, R_{sv2})$,则对$\forall R_{sv}\in[R_{sv1}, R_{sv2}]$可以有$S_v(\mathbf{K}, R_{sv})=S_v(\mathbf{K}, R_{sv1})$.对$P_v(\mathbf{K}, R_{sv})$也有类似的特性.利用这一特性可以加快Steiner树的求解过程.具体做法是:定义一个栈数据结构保存$\mathbf{K}$,v,R_{sv1},R_{sv2}以及对应的$S_v(\mathbf{K}, R_{sv1})$的降级分解方法,该结构按$\mathbf{K}$和v分层保存构成结果栈,同层的数据按R_{sv1}顺序保存.加速后的整个求解过程自上而下递归进行,在求解$S_v(\mathbf{K}, R_{sv})$之前先在栈中查找已有的解,如果有合适的解则将其返回,否则遍历其所有的降级分解方式求解得到$S_v(\mathbf{K}, R_{sv})$,更新结果栈中的数据,并返回.

2.5 实验结果

理论上可以证明当线网中仅有一个关键点时,用基于Sakurai模型的时延驱动Steiner算法必能取得使关键点时延最短的解.我们在Sun Sparc20上用C语言实现了时延驱动的DW算法,并用MCNC的C2、C5和C7三个测试例子加以测试.这三个例子都是标准单元集成电路实例,线网数从600到1900多,线网的结点数由2到17不等.该算法所得结果的时延性能优于DW算法和CFD算法.

图1是C2:718线网的实际布线结果.由于Sakurai时延模型较好地反映了Steiner树的形状和时延之间的关系,算法随特征宽度由8μm逐渐减小到0.6μm始终能够得到相应的时延最优解.而使用简化Elmore时延模型的DW算法得到的解,则始终是与8μm工艺对应的最优解(图中s点为源点,t点为关键点,由于总体布线图所限,所有漏点只能通过纵向连线连通).

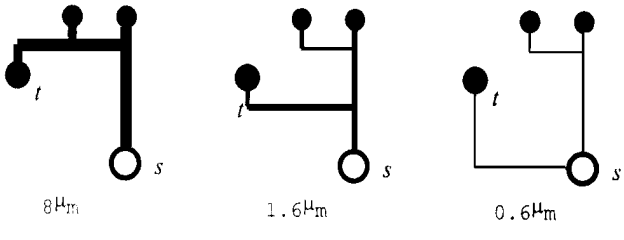

图1 时延最短的Steiner树随工艺变化的情况

表1和表2中分别给出了对应1.6μm工艺和0.6μm工艺的部分结果(8μm工艺下DW算法的结果与基于Sakurai模型的结果相同).可以看出随特征宽度的减小,与DW算法结果相比,基于Sakurai模型结果总的线长增加,同时关键点的时延明显减少.特征宽度越小,时延的减少就越明显.

表 1　1.6μm 工艺下 IDW 算法与基于 Sakurai 模型的 Steiner 算法结果的性能对比

线网号	关键点时延/ns			总的线长/μm		
	IDW	Sakurai	增量/%	IDW	Sakurai	增量/%
C2: 718	1.06	1.05	-0.94	814	965	18.55
C2: 122	2.00	1.99	-0.50	1187	1192	0.42
C5: 1094	2.23	2.18	-2.24	2110	2186	3.60
C5: 1084	2.39	2.35	-1.67	1449	1528	5.45
C5: 637	4.67	4.08	-12.63	3510	4136	17.83
C5: 569	4.96	3.81	-23.19	3969	4222	6.37
C5: 553	7.12	6.40	-10.11	4751	5966	25.57
C5: 332	1.80	1.79	-0.56	1531	1647	7.58
C7: 1761	1.27	1.23	-3.15	839	891	6.20
C7: 918	1.39	1.35	-2.88	963	1015	5.40
C7: 613	3.39	3.26	-3.83	3067	3627	18.26
C7: 611	3.40	3.35	-1.47	2819	2821	0.07
平均	2.97	2.74	-7.74	2251	2516	11.77

表 2　0.6μm 工艺下 IDW 算法与基于 Sakurai 模型的 Steiner 算法结果的性能对比

线网号	关键点时延/ns			总的线长/μm		
	IDW	Sakurai	增量/%	IDW	Sakurai	增量/%
C2: 718	1.81	1.30	-28.18	814	1112	36.61
C2: 122	3.43	3.33	-2.92	1187	1192	0.42
C5: 1094	4.56	3.39	-25.66	2145	2377	10.82
C5: 1084	4.39	3.59	-18.22	1449	2104	45.20
C5: 637	11.33	6.71	-40.78	3510	4136	17.83
C5: 569	11.72	5.17	-55.89	3969	4422	11.41
C5: 553	17.47	10.27	-41.21	4751	5966	25.57
C5: 332	2.94	2.67	-9.18	1531	1643	7.32
C7: 1761	1.78	1.47	-17.42	839	891	6.20
C7: 918	2.02	1.68	-16.83	963	1015	5.40
C7: 613	8.02	6.25	-22.07	3067	3627	18.26
C7: 611	8.53	5.64	-33.88	2819	4372	55.09
平均	6.50	4.29	-34.00	2254	2738	21.47

对该结果要说明四点: (1) Steiner 树中结点的位置没有随工艺变化而变化, 所以时延驱动算法的优势被放大了一些(芯片面积没有随线宽变细而变小). (2) $\hat{r}$和 $\hat{c}$的实际值与走线层有关, 而在总体布线过程中又不能预先确定走线层, 所以只能使用近似值, 对应 1.6μm 工艺, 取 $\hat{r}=0.5\Omega/\mu m$, $\hat{c}=1.8\times10^{-16}F/\mu m$ 对应 0.6μm 工艺, 取 $\hat{r}=3\Omega/\mu m$, $\hat{c}=1.4\times10^{-16}F/\mu m$. (3) 时延模型用的是输出 V_o 从 0 增加到 $0.9V_{DD}$ 的 Sakurai 时延计算公式 (4) 对上述线网用 IDW 算法求得的的解与线长最优解差别不大

在运行效率上, 基于 Sakurai 的算法, 比以线长为优化目标的 DW 算法差, 而与 IDW 算法相当. 前面提出的加速算法, 在线网较大时, 可以提高求解速度 4～7 倍. 即使这样, 随线网大小不同, 时延驱动的算法的求解时间也大约为原 DW 算法的 2～4 倍. 因此需要与 Steiner 树分层构造算法相结合, 构造出效率较高, 而时延性能降低不多的时延驱动的 Steiner 树分层构造算法. 进一步的工作是将基于 Sakurai 模型的时延驱动 Steiner 树算法用于基于线网

或关键路径的时延驱动总体布线算法,从而得到时延性能较佳的总体布线结果

参 考 文 献

[1] 洪先龙,严晓浪,乔长阁,VLSI布图理论和方法讲义,北京:科学出版社(待出版),第一章

[2] H. B. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI", Addison-Welsley, 1990.

[3] Prasitjutrakul S., Kubitz W. J., "A Timing Driven Global Router for Custom Chip Design", Proc. International Conference on CAD, Santa Clara, 1990, 48～51.

[4] C. Chiang, M. Sarrafzadeh, C. K. Wong, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 1990, **9** (12): 1318～1325.

[5] C. Chiang, C. K. Wong, M. Sarrafzadeh, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 1994, **13** (12): 1461～1469.

[6] 洪先龙, 计算机学报, 1995, **18**(4): 266～272.

[7] 洪先龙, 半导体学报, 1995, **16**(3): 218～223.

[8] W. C. Elmore, J. Appl. Phys., 1948, **19**: 55～63.

[9] S. E. Deryfus, R. A. Wagner, Networks, 1972, **1**: 195～207.

[10] 乔长阁,洪先龙, 微电子学与计算机, 1996, **13**(4): 8～10.

[11] T. Sacurai, IEEE J. Solod-State Circuits, 1983, **18**(4): 418～426.

[12] E. Horneber *et al.*, "A Closed-Form Expression for Signal Delay in CMOS Driven Branched Transmission Lines", Proc. VLSI'87, 1988, 353～362.

[13] D. L. Carter *et al.*, "Analysis of Signal Propagation Delays and Chip Level Performance Due to On-Chip Interconnections", Proc. Computer Design: VLSI in Computers Conference, 1983, 218～221.

Timing-Driven Steiner Tree Algorithm Based on Sakurai Model

Bao Haiyun, Hong Xianlong, Cai Yici, Qiao Changge

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Received 25 October 1997, revised manuscript received 23 February 1998

Abstract This paper presents a new Timing-Driven Steiner Tree algorithm. The algorithm estimates the delay from source to sink of a net based on the Sakurai Delay Model, and traverses all available Steiner trees via Dreyfus-Wagner Steiner approach. Under submicron technology, this algorithm will provide much better solution for the delay of critical vertices of net compared to IDW and CFD algorithm.

EEACC: 7410D, 5120