第 21 卷第 3 期　　半　导　体　学　报　　Vol. 21, No. 3
2000 年 3 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Mar., 2000

# 高速多级时钟网布线*

李芝燕　　严晓浪

(浙江大学计算机系　杭州　310027)　　(杭州电子工业学院 CAD 所　杭州　310037)

**摘要**　提出了一种新的加载缓冲器的时钟布线算法. 该算法根据时钟汇点的分布情况, 在时钟布线之前对缓冲器进行预先布局, 并将时钟树的拓扑生成及实体嵌入和层次式的缓冲器布局方法有机结合起来, 使布线情况充分反映缓冲器对时钟网结构的影响. 实验证明, 与将缓冲器插入和布局作为后处理步骤相比, 缓冲器预先插入和布局在很大程度上避免了布线的盲目性, 并能更加有效地实现各时钟子树的延迟和负载的平衡.

**关键词**: 时钟布线, 延迟合并嵌入, 缓冲器插入

**EEACC**: 7410D, 5120

**文章编号**: 0253-4177(2000)03-0290-08

# High Speed Multilevel Staged Clock Routing*

LI Zhi-yan

(*Department Computer Sciences, Zhejiang University, Hangzhou*　310027, *China*)

YAN Xiao-lang

(*Hangzhou Institute of Electronic Engineering, Hangzhou*　310037, *China*)

Received 12 January 1999, revised manuscript received 20 July 1999

**Abstract**　Clock routing is one of major steps in high performance-driven layout design under deep sub-micron technology. Buffered clock tree construction is a key factor for clock routing. A novel buffered clock routing algorithm is proposed. The strategy is to perform buffer insertion and placement according to clock sink distribution before clock net routing, and to optimize clock tree topology generation, detailed embedding following the buffer insertion process. The influence of the placed buffers on routing will be significantly reflected. The experimental results show that buffer pre-placement will avoid blind routing in a great extent and achieve the balance of sub-tree delay and load capacitance efficiently.

---

* 国家九五攻关项目(95-738-01-08)资助[Project Supported by National ninth 5-year Plan of China Under Contract No. 95-738-01-08].

李芝燕　博士研究生, 从事方向集成电路 CAD 版图设计.

严晓浪　教授, 从事方向为集成 CAD 自动化设计的研究和教学.

1999-01-12 收到, 1999-07-20 定稿

**Key Words**: Clock Routing, Deferred Merging Embedding (DME), Buffer Insertion
**EEACC**: 7410D, 5120
**Article ID**: 0253-4177(2000)03-0290-08

## 1　引言

高速时钟布线是深亚微米下性能驱动的版图设计中的重要一环. 在大多数的 VLSI 电路中, 各个作用单元之间的数据传输是由时钟信号来进行同步控制的, 时钟频率决定了数据进程和数据传输的速度. 随着器件尺寸减小到深亚微米阶段, 时钟偏差(Clock Skew)已成为决定电路性能的关键因素. 时钟布线设计的主要目标就是时钟偏差、相位延迟最小化, 并尽量减少时钟网的功耗. 目前, 大多数时钟平衡算法通过加宽或拉长金属互连线获得[1~9]. 由于时钟网的总电容是决定时钟网功耗的决定因素, 因此以上方法导致了时钟网功耗的增加. 时钟网作为全局分布的线网, 具有连线长、扇出负载重的特点, 在实际布线中, 必须加入相应驱动能力的缓冲器来减少时钟网的电容负载效应. 在布线中有效地插入缓冲器不仅可使时钟偏差、时钟树的延迟及上升/下降时间减小; 并且, 由于加载缓冲器后, 为了保证时钟树可靠性设计所需的连线面积减小, 从而明显地降低了时钟网的功耗.

在以前的大多数算法[10~12]中, 将缓冲器定位和布局作为后处理步骤, 虽然在文献[9]中, 对拓扑布线和缓冲器插入同时进行了理论上的考虑, 但未能明确实际具体布线与缓冲器定位的关系. 以前大多算法还存在的一些不足之处是: (1) 在拓扑生成及布线过程中未将缓冲器及实际布线环境的影响考虑进去, 导致了布线的盲目性, 使得时钟树的延迟增加. (2) 每次在时钟布线中层次式插入缓冲器后, 都需要修改原布局网表(时钟汇点的位置可能发生偏移), 分阶段地修改网表有时会破坏已生成的时钟网. 随着插入缓冲器数目的增多, 计算效率也会受到影响. (3) 层次式插入缓冲器后, 通常需要加入额外的连线来平衡缓冲器的负载电容和延迟, 造成总电容和功耗的增加.

针对以上问题, 在本文中我们提出一种缓冲器预布局算法, 该算法可将拓扑生成、实体嵌入和缓冲器布局同时进行考虑, 布线情况将充分反映了缓冲器对时钟网结构和性能的影响. 实验证明, 在布线过程中加入缓冲器可以更加有效地实现各时钟子树的延迟和负载的平衡, 避免布线的盲目性. 由于在算法中对时钟电路采用了平衡划分策略, 各局部区域生成的时钟子树的负载和延迟值是相对平衡的, 保证了层次式缓冲器布局算法的有效性.

## 2　互连线和 BUFFER 的延迟模型及延迟计

从多级时钟树的源点到某一汇点的延迟由两部分组成: (1) 互连线延迟; (2) 通过缓冲器和驱动门的延迟. 在时钟树的构造中, 我们用 Elmore 延迟来估计互连线延迟. 对于缓冲器的电路模型, 我们用图 1 所示的等效模型来表示, 其中 $c_b$, $r_b$ 分别为缓冲器的输入电容和输出电阻, $d_b$ 为缓冲器的内部延迟.

要精确地计算一棵时钟树的 RC 延迟是相当困难的, 但近似地估计延迟并不困难. 我们利用 Elmore

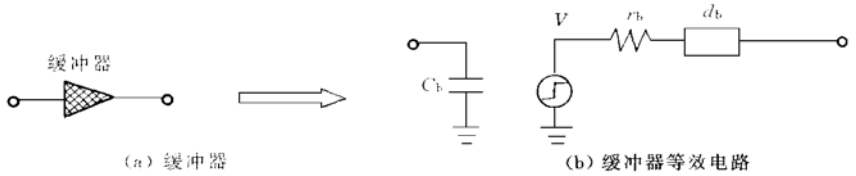



图 1　缓冲器等效电路延迟模型

FIG. 1　Delay Model for Buffer Equivalent Circuit

模型来计算 RC 树网络的延迟. 当波形近似单调时,该模型可以对信号延迟进行有效的估计. 我们以 $T$ 来代表一棵 RC 树,$T_i$ 代表以节点 $i$ 为根节点的 RC 子树. 令 $C_i$ 为子树 $T_i$ 的总电容,$r_i$ 为树枝 $i$ 的电阻(我们把以节点 $i$ 及它的父节点为端点的树枝记为树枝 $i$), 则任意两节点 $i$ 和 $j$ 之间的延迟时间 $t_{ij}$ 可以按下式递归地计算:

$$t(i,j) = \sum_{k \in N(i,j)} r_k C_k \tag{1}$$

式中 $N(i,j)$ 为节点 $i$ 与节点 $j$ 之间路径上所有节点的集合,包括 $j$ 但不包括 $i$.

若令 $s_0$ 为时钟源点,$s_i(1 \leqslant i \leqslant N)$ 为时钟汇点,则该时钟树的时钟偏差为:

$$\text{skew} = \max_{1 \leqslant i,j \leqslant N} \left| t(s_0, s_i) - t(s_0, s_j) \right| \tag{2}$$

如果从时钟源点到各时钟汇点的延迟时间都相等,则称时钟树是零偏差的. 在零偏差时钟树生成中,时钟树是由子树的两两合并生成的. 对于加载缓冲器的时钟树来说, 生成新的子树要将缓冲器延迟和缓冲器对子树负载电容的影响考虑进去,如图 2 所示. 令 $v_1$ 和 $v_2$ 分别为子树 $T_{v1}$ 和 $T_{v2}$ 的根节点,节点 $v$ 表示由

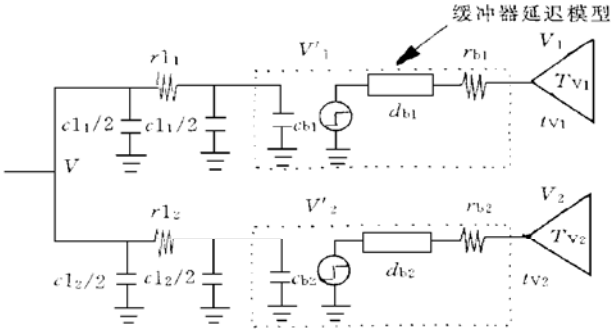



图 2 加载缓冲器的零偏差合并

FIG. 2 Zero-Offset of Buffer with Loads

子树 $T_{v1}$ 和 $T_{v2}$ 生成的新子树的根节点,$t_v$ 分别表示子树 $T_v$ 的负载电容和延迟,则有:

$$t_v = rl_1\left(\frac{cl_1}{2} + c_{b1}\right) + d_{b1} + r_{b1} \times C_{v1} + t_{v1} = rl_2\left(\frac{cl_2}{2} + c_{b2}\right) + d_{b2} + r_{b2} \times C_{v2} + t_{v2} \tag{3}$$

## 3 算法思想

本文提出一种新的缓冲器布局算法,它的基本思想是: 根据时钟汇点在布线平面上的几何分布和负载电容值,对时钟汇点集合进行划分,通过局部的拓扑生成完成时钟子树的拓扑布线; 而后,为每个子树分配一个适当驱动能力的缓冲器,缓冲器的精确尺寸将在完成子树的详细布线后决定,而缓冲器的布局位置可以在布线前确定. 由于在布线前,对时钟汇点集合进行了相对平衡的划分,所以每个子集可由一个驱动能力相当的缓冲器来驱动,并由子集内时钟汇点的分布情况决定缓冲器的位置. 缓冲器预布局的一个关键问题是如何控制缓冲器的数目,因为它直接影响到面积和功耗的优化. 本文算法中对于缓冲器数目的控制主要由划分的两个约束条件决定,一是汇点子集的电容负载约束,二是各子集结群的最大半径约束.

### 3.1 时钟汇点集的划分

我们首先对时钟汇点集进行自顶向下的划分. 划分策略类似于文献[6]中的平衡划分方法,将给定的时钟汇点集划分成负载相对平衡的若干子集. 划分过程是一个递归的过程, 但是本文中并不将子集划分到只剩一个结点为止, 而是根据电路规模使划分后各子集结群的电容负载满足一定的结群约束条件; 同时,为了使各子集结群尽可能平衡,我们还要求子集满足相应的结群半径限制条件. 对于给定一个时钟汇

点集 $S$,定义 $S$ 的半径 $\mathrm{dia}(S)$ 为 $\max\{\mathrm{Distance}(p_i,p_j)\mid p_i,p_j\in S\}$. $S$ 的一个八边形闭包 $\mathrm{Oct}(S)$ 是由以下八条线段围成的闭包: $x=\min\{p_i.x\mid p_i\in S\}$, $x=\max\{p_i.x\mid p_i\in S\}$, $y=\min\{p_i.y\mid p_i\in S\}$, $y=\max\{p_i.y\mid p_i\in S\}$, $y-x=\min\{p_i.y-p_i.x\mid p_i\in S\}$, $y-x=\max\{p_i.y-p_i.x\mid p_i\in S\}$, $y+x=\min\{p_i.y+p_i.x\mid p_i\in S\}$, $y+x=\max\{p_i.y-p_i.x\mid p_i\in S\}$,其中 $p_i.y,p_i.x$ 分别为点 $p_i$ 的 $x,y$ 的坐标值. 如果一个划分 $P$ 将点集 $S$ 划分成两个子集 $S_1$ 和 $S_2$, 则划分 $P$ 的代价为: $\mathrm{dia}(S_1)+\mathrm{dia}(S_2)$.

算法的基本思想如下:

(1) 计算 $\mathrm{Oct}(S)$, 对所有属于 $S$ 点集并位于闭包 $\mathrm{Oct}(S)$ 上的点依次进行排序, 取包含 $1/2|\mathrm{Oct}(S)|$ 个连续点集作为参考集 $S_1$;

(2) 对每个属于 $S$ 的点 $p_i$, 计算其权值: $W_{pi}=\min\{\mathrm{distance}(p_i,p_j)\mid p_j\in S_1\}+\max\{\mathrm{distance}(p_i,p_j)\mid p_j\in S_1\}$, 按照权值大小对 $S$ 中的汇点进行排序, 并按此顺序将汇点依次归入 $S_1$ 直到划分后的两个子集的汇点总电容负载 $C_1$、$C_2$ 的差值满足: $|C_1-C_2|\leqslant\max\{C_{Li}\}$ 且 $|C_1+C_2-2C_{Lim}|\leqslant\Delta C$ 为止, 其中 $\{C_{Li}\}$ 为这两个子集中所有时钟汇点的负载电容值集合, $C_{Lim}$ 为给定的各结群的总负载电容限制. 当 $S$ 内的所有时钟汇点都被划分到参考集 $S_1$ 及另一个集合 $S_2$ 中后, 计算该划分的代价;

(3) 对当前 $\mathrm{Oct}(S)$ 的所有参考集, 取满足子集结群电容和半径约束条件, 并且划分代价最小的划分作为 $\mathrm{Oct}(S)$ 的划分结果;

(4) 当所有递归划分后的子集的电容负载值均满足(2) 中的约束条件且每个子集的半径小于预定半径时, 算法结束; 否则回到(1).

由于在(1) 中参考集 $S_1$ 的选取不是唯一的, 经过(2) 后, 参考集 $S_1$ 有多种可能性, 我们将选择满足子集约束条件且划分代价最小的参考集.

以上时钟汇点的划分是一个不断递归的过程, 直到将所有的时钟汇点子集都满足条件为止. 另外, 还要注意以下两点: (1) 当时钟电路的规模较小时, 可不必经过划分步骤; (2) 考虑到时钟树靠近叶子节点的下层节点对整棵时钟树的延迟影响较小, 应设置上层划分的结群总电容值大于下层划分的结群总电容, 使得上层合并点有更大的搜索空间.

### 3.2　时钟子树拓扑生成和实体嵌入

当时钟汇点集被划分为若干满足条件的子集后, 可并行地对各子区域内的汇点进行局部拓扑生成和实体嵌入. 本文提出一种层次式的加权平衡算法, 与目前较好的算法[8] 相比, 我们的算法可以在几乎不增加连线长度的情况下, 得到延迟更加均匀的时钟树.

拓扑生成是一个由底向上的过程. 根据延迟合并嵌入原则[7], 在由底向上的拓扑构造过程中, 时钟树每个中间节点的布图轨迹由生成的曼哈顿弧, 即合并段决定. 对于参数均匀的金属布线平面, 曼哈顿弧是一条斜率为+ 1 或− 1 的线段, 有时它可能退化成一点.

层次式加权平衡算法的基本思想是, 首先将当前集合 $S$ 内的全部时钟汇点封闭在 $\mathrm{Oct}(S)$ 内, 建立相应的拓扑表来按序存放所有当前时钟汇点及合并线段的信息; 然后, 取 $\mathrm{Oct}(S)$ 上任一点并计算其与拓扑表中其它合并点(段) 的权值. 我们定义当前待合并点(段) 与集合中任一合并点(段) 之间的权值为它们之间的距离和延迟差的加权和, 加权系数可根据要求进行调整. 我们取最小权值所对应的合并点(段) 作为优先的合并对象. 这是由于延迟相近的线网优先考虑合并, 可使生成的时钟树中同层子树的延迟较为平衡. 同时, 距离较近的子树优先进行合并有利于优化连线长度. 合并次序是从 $\mathrm{Oct}(S)$ 闭包上依次选择待合并点, 最终达到整体时钟树的均衡.

生成拓扑结构的基本过程可表述如下:

(1) 对于给定的时钟汇点集合, 计算 $\mathrm{Oct}(S)$;

(2) 顺序选取 $\mathrm{Oct}(S)$ 闭包各边上的点, 添入拓扑表中进行拓扑排序;

(3) 从 $\mathrm{Oct}(S)$ 将已经排序过的点删除, 回到(1), 直到所有汇点都完成拓扑排序;

(4) 按序取出拓扑表中的元素, 得到与其相匹配的权值最小的合并点(段), 生成新的合并段并将其放入拓扑表中, 直到生成完整的合并段树.

该算法总的时间复杂度为 $O(n^2)$,空间上只需存储拓扑表和两个障碍表(分别为水平和垂直障碍表),计算效率是比较高的.与目前较好的时钟树生成算法[7]等相比,本文的拓扑生成算法由于在合并过程中考虑了延迟平衡,有效地抑制了可能遗留下的"孤点"对整体时钟树平衡性的破坏.

局部拓扑生成以后,我们对生成的各时钟子树进行自顶向下的斯坦纳点的嵌入.以零偏差合并为例,嵌入过程的基本思想是:在曼哈顿平面上,依照已生成的合并线段树(该树实际上同时记录了时钟树的拓扑结构信息),从根节点所对应的合并段开始,在该合并段上选择距离时钟源点最近的一点作为实际布线点;然后,在该布线点的两个子合并段上分别找到距离当前布线点最近的合并点位置,调用详细布线器进行布线;而后,再分别从这两个节点开始,通过自顶向下的方式确定它的下级合并点的位置,如此类推即可实现整棵时钟子树的精确定位.

### 3.3 缓冲器的插入和布局

在时钟树中插入缓冲器有两种形式:一种级联式,另一种是分布式.本章采用的是分布式插入方式.在某些情形下,用级联式就可以满足所有的设计约束,然而,随着时钟树规模的增大,分布式更加合理和易行.

本文的缓冲器插入和布局算法与上述汇点划分算法是紧密结合的.我们根据电路规模的大小和实验的经验值,通过平衡划分策略,将划分后的各个汇点集的电容负载值和规模限定在一定范围内,通过局部的拓扑生成完成时钟子树的拓扑布线;而后,为每个子树分配一个适当驱动能力的缓冲器,缓冲器的精确尺寸将在完成子树的布线后决定,而缓冲器的布局位置可以在布线前确定.由于在布线前,对时钟汇点集合进行了相对平衡的划分,每个子集可由一个驱动能力相当的缓冲器来驱动,并由子集内时钟汇点的分布情况决定缓冲器的位置.缓冲器的具体布局位置根据以下原则来确定:(1)缓冲器应尽可能靠近其子树根节点;(2)缓冲器作为布局网表中新的实体单元,其布局位置不能与已有的实体发生重叠,并尽量使各缓冲器在整个布线平面上均匀分布.

由于在本文算法中,划分后各子集在布线平面上均匀分布且没有交集,从而保证了各缓冲器可以无重叠地均匀分布.为了使缓冲器应尽可能靠近其子树根节点,我们对各子集内的汇点进行拓扑布线,由延迟合并原则生成合并线段树.根据合并线段树中子树根节点对应的合并段位置,确定缓冲器的插入位置.在时钟分布网络中加入缓冲器的层数取决于扇出大小、互连线负载以及延迟/面积的折中优化.在满足各项性能要求的条件下,应使得加入的缓冲器尽量少.由于缓冲器本身带有一定的延迟并消耗一定的能量,过多的缓冲器将导致延迟和功耗的增加.在时钟树中加载缓冲器的层数与功耗的关系示意图如图3所示.由图可知,随着时钟树中加载缓冲器层数增加,功耗明显下降;当缓冲器层数增加到一定数时,功耗又明显增加.在本文中,我们根据电路规模的大小和实验的经验值,通过划分策略将划分后各个汇点集的电容负载值和相对规模限定在一定范围内,并由局部的拓扑生成和布线近似估计功耗的下降趋势,避免加入过多的缓冲器.

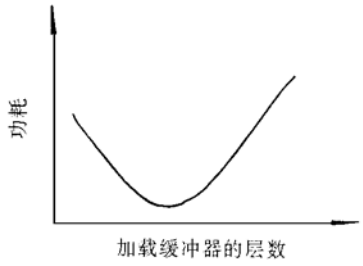



图3 功耗与缓冲器层数的关系

FIG.3 Qualitative Dependence of Power and Layer-Number of Buffer with Loads

缓冲器的插入不一定能一次完成,尤其是对于大规模的时钟电路来说,缓冲器的插入是层次式逐级完成的.为了实现缓冲器的层次式插入和布局,我们采用一种启发式方法:将上一次划分过程中插入的所有缓冲器的输入端作为新的汇点集合,然后根据新汇点集合的几何分布、延迟和负载情况对该汇点集合进行相应的划分操作并同步完成缓冲器插入和布局.该过程也是一个递归过程,可以方便地同汇点划分算法结合起来.

在拓扑布线中进行缓冲器定位的基本过程如下:

(1) 计算 $\mathrm{Oct}(S)$,取包含 $1/2|\mathrm{Oct}(S)|$ 个连续点集作为参考集;

(2) 对每个属于 $S$ 的点 $p_i$,按照权值大小对 $S$ 中的汇点进行排序,并依次归入 $S_1$ 直到所划分后的两

个子集满足电容和半径约束条件,计算该划分的代价;对当前的 Oct($S$) 的所有参考集,取满足子集约束条件,且划分代价最小的划分作为该 Oct($S$) 的划分结果;

(3) 对每个子集内的汇点进行子树拓扑布线,生成相应合并线段子树;

(4) 为每棵时钟子树确定缓冲器的插入点,在各合并线段树中根节点对应的合并段上取一点,作为驱动该子树的缓冲器的布局位置,根据该位置对原布局网表进行修改;

(5) 以当前插入的所有缓冲器的布局点作为新汇点集合 $S$ 并估算各汇点的电容和延迟值,回到(1),直到所有递归划分后的子集均满足约束条件时,算法结束.

从算法中可以看到,我们在拓扑布线过程中完成所有层次的缓冲器插入,也就是说在进行实体布线前所有缓冲器的布局位置已经确定了.在线网的延迟合并嵌入阶段,我们不再对各实体单元的布局位置进行调整,从而减少了后处理的工作量. 由于在整个芯片时钟汇点的划分中, 对每个结群的总电容负载进行了相对平衡的划分, 因此对于时钟汇点分布比较均匀的电路,我们可以为每个结群加入驱动能力相同的缓冲器,与非层次地加入缓冲器或加入驱动能力各异的缓冲器相比,每层加入驱动能力相同的缓冲器可以有效地降低时钟树的时钟偏差敏感度, 提高其可靠性.

## 4　实验结果

我们将以上的算法在 SUN SPARC-10 工作站上实现. 我们首先用了四组例子对算法进行了测试. 互连线的单位电阻和单位电容分别为 2mΩ 和 0.02fF, 时钟缓冲器所用的参数同文献[11]. 缓冲器的功耗采用 $k$ 参数[9]进行估计. 为了进行性能比较,我们用文献[12]中的方法与本文的算法进行了比较(见表 1). 在文献[12]中 BUFFER 是在拓扑生成以后插入时钟树的,两种情况下,时钟偏差均小于延迟的 2%,故不再在表中列出.

**表 1　布线结果的比较**

**Table 1　Comparison for Layout Results**

| Example | | BUFFER 插入作为后处理步骤[12] | | | 本文的算法 | | |
|---|---|---|---|---|---|---|---|
| | 汇点数 | 线长/mm | 延迟/ns | 功耗/W | 线长/mm | 延迟/ns | 功耗/W |
| E1 | 118 | 78 | 0.52 | 0.264 | 76 | 0.46 | 0.261 |
| E2 | 216 | 104 | 0.84 | 0.547 | 83 | 0.83 | 0.534 |
| E3 | 512 | 225 | 1.75 | 0.821 | 228 | 1.64 | 0.715 |
| E4 | 2048 | 792 | 3.32 | 2.429 | 791 | 3.15 | 2.406 |

除上述例子外, 我们对 MCNC 中的标准单元模式实例 E1~ E3 进行了测试,E1~ E3 的有关参数见表 2. 实验结果见表 3,以 E1~ E2 为例,加载缓冲器的布线图见图 4 和图 5. 图中淡颜色的单元模块为布局后的标准单元,深颜色的模块为布局后插入的缓冲器单元,布线采用水平/垂直两层金属布线模式,所有单元

**表 2　测试实例及其参数**

**Table 2　Measured Example and Its Parameters**

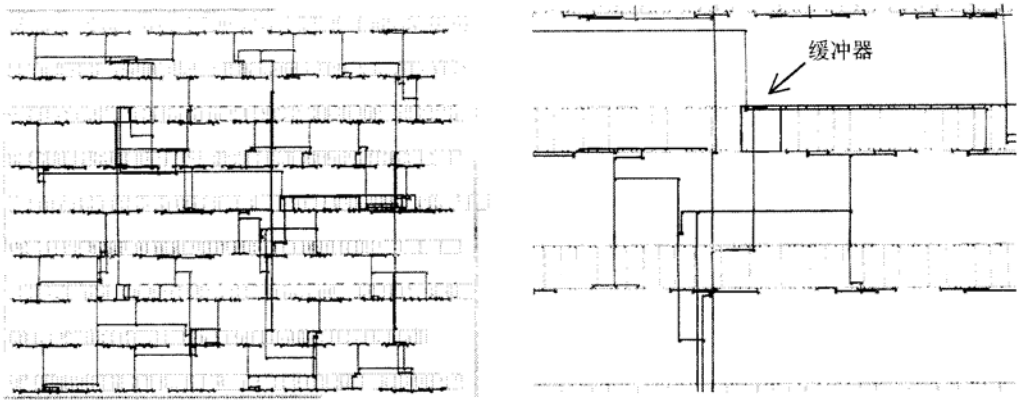
| Cir. | # CELL | 时钟汇点数 | Pin 障碍数 | 面积/$\mu m^2$ | 汇点负载电容范围/fF |
|---|---|---|---|---|---|
| R1 | 589 | 286 | 3538 | 640×520 | 70~ 100 |
| R2 | 589 | 568 | 3538 | 640×510 | 70~ 100 |
| R3 | 21849 | 1210 | 180902 | 2820×2810 | 50~ 80 |

表 3 加载缓冲器前与加载缓冲器后结果对比

Table 3 Results Comparison Without and with Buffer

| Cir. | 加载缓冲器前 | | | | 加载缓冲器后 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 总线长/μm | 延迟/ns | 偏差/ps | CPU/min | 总线长/μm | 延迟/ns | 偏差/ps | CPU/min | # Buffer/个 |
| E1 | 6826 | 0.90 | 3 | 0.40 | 7417 | 0.52 | 3 | 0.21 | 4 |
| E2 | 14616 | 2.87 | 5 | 1.58 | 14780 | 0.78 | 6 | 1.73 | 9 |
| E3 | 201295 | 12.62 | 10 | 19.73 | 203118 | 1.36 | 12 | 22.98 | 32 |

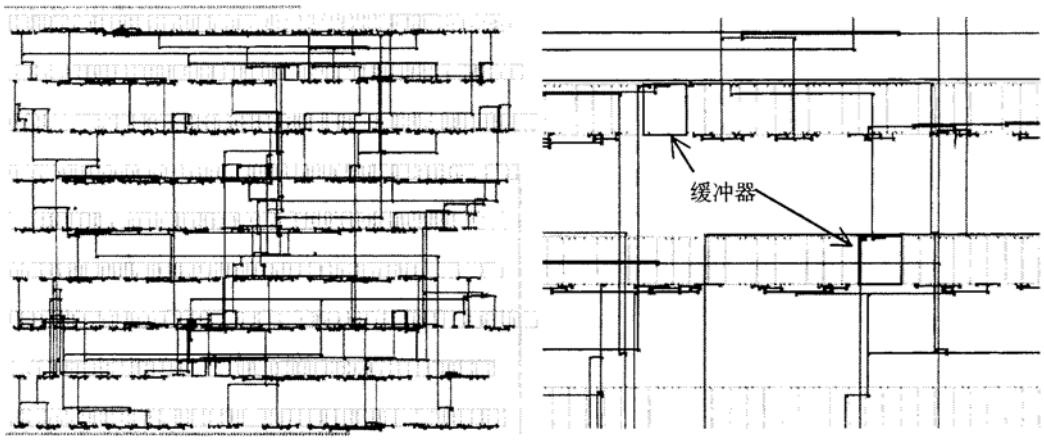
上的金属引脚作为障碍处理. 与加载缓冲器前的结果相比, 加入缓冲器后的连线总长度有所增加, 但是延迟得到了相当显著的改善. 特别是对于大规模的电路, 在时钟树中加载缓冲器是提高性能的一种有效方法, 这是 RC 树的规模越大, 缓冲器对电容的去耦合效应越明显.




(a) 加载缓冲器的 E1 布线图 (b) E1 的局部放大图

图 4 E1 布线结果图

FIG. 4 Layout of Example E1




(a) 加载缓冲器的 E2 布线图 (b) E2 的局部放大图

图 5 E2 布线结果图

FIG. 5 Layout of Example E2

与目前已发表的缓冲器布局算法相比较,本文算法采用阶段布线策略使之可以较准确地估计负载电容,并且缓冲器与时钟子树根节点之间不必在布线过程中或布线后再加入额外的延迟线来平衡各子树的负载和延迟.由于在布线过程中完成缓冲器插入和定位,这种缓冲器的布局方法大大减少了后处理的工作量.

## 5　结论

本文提出一种层次式多级时钟网布线算法.该方法将时钟网拓扑生成、实体嵌入和 BUFFER 布局有机地结合在一起,避免了布线的盲目性,提高了布线质量.今后,我们将考虑将变线宽技术结合在我们的算法中,并探索更加有效精确的延迟模型.

## 参　考　文　献


[1] A. L. Fishburn and H. T. Kung, Synchronous Large Systolic Arrays, Proc. of SPIE, 1982, 45~ 52.

[2] M. Jackson, A. Srinivasan, and E. S. Kuh, Clock Routing for High Performance ICs, Proc. of 27th ACM/IEEE DAC, 1990, 573~ 579.

[3] A. B. Kahng, J. Cong and Robins, Hign-Performance Clock Routing Based on Recursive Geometric Matching. Proc. of 28th ACM/IEEE DAC, 1991, 322~ 327.

[4] R. S. Tsay, Exact Zero Skew, Proc. of IEEE ICCAD, 1991, 336~ 339.

[5] Q. Zhu and W. W.-M. Dai, Perfect-Balance Planar Clock Routing With Minimal Path-Length, Proc. IEEE Int. Conf. CAD, Nov., 1992, 473~ 476.

[6] T. H. Chao, Y. C. Hsu and J. M. Ho, Zero Skew Clock Net Routing Proc. of 29th ACM/IEEE DAC, CA, 1992, 518~ 523.

[7] M. Edahiro, A Clustering Based Optimization Algorithm in Zero Skew Routing, Proc. of 30th ACM/IEEE DAC, TX, 1993, 612~ 616.

[8] M. Edahiro, Minimum Skew and Minimum Path Length Routing in VLSI Layout Design, NEC RES. DECEL. Oct., 1991, 569~ 575.

[9] T. H. Chao, Y. C. Hsu and J. M. Ho, Zero Skew Clock Net Routing, Proc. ACM/IEEE DAC, 1992, 518~ 523.

[10] J. D. Cho and M. Sarrafzadeh, A Buffer Distribution Algorithm for High-Speed Clock Routing, Proc. ACM/IEEE DAC, 1993, 537~ 543.

[11] S. Pullela, N. Menezes, J. Omar and L. T. Pillage, Skew and Delay Optimization for Reliable Buffered Clock Trees, Proc. of the IEEE ICCAD, 1993, 556~ 562.

[12] Y. P. Chen and D. F. Wong, An Algorithm for Zero-Skew Clock Tree Routing With Buffer Insertion, Proc. European Design and Test Conf., 1996.