时钟延时及偏差最小化的缓冲器插入新算法

曾 璇[1], 周丽丽[1], 黄 晟[1], 周 电[2], 李 威[2]
(1. 复旦大学微电子系专用集成电路与系统国家重点实验室, 上海 200433; 2. 美国德州大学达纳斯分校电子工程系, TX 75083)

摘 要: 本文提出了以最小时钟延时和时钟偏差为目标的缓冲器插入新算法. 基于 Elmore 延时模型, 我们得到相邻缓冲器间的延时是缓冲器在时钟树中位置的凸函数. 当缓冲器布局使所有缓冲器间延时函数具有相同导数值时, 时钟延时达到最小; 当所有源到各接收端点路径的延时函数值相等时, 时钟偏差达到最小. 对一棵给定的时钟树, 我们在所有从源点到各接收端点路径上插入相同层数的缓冲器, 通过优化缓冲器的位置实现时钟延时最小; 通过调整缓冲器尺寸和增加缓冲器层数, 实现时钟偏差最小.

关键词: 时钟分布; 缓冲器插入; 时钟布线; 时钟延时; 时钟偏差
中图分类号: TN79+1 文献标识码: A 文章编号: 0372-2112 (2001) 11-1458-05

A Novel Buffer Insertion Algorithm for Clock Delay and Skew Minimization

ZENG Xuan[1], ZHOU Li-li[1], HUANG Sheng[1], ZHOU Dian[2], LI Wei[2]
(1. *Department of Electronic Engineering, Fudan University, Shanghai 200433, China;*
2. *Department of Electronic Engineering, The University of Texas at Dallas, Richardson, TX 75083, U.S.A.*)

Abstract: In this paper, we propose a novel buffer insertion theory for clock delay and skew minimization. Based on the Elmore delay model, buffer to buffer delay is a convex function of buffer positions in a clock tree. The optimal buffer placement for delay minimization is achieved when all delay functions have the same derivative values. The minimal skew can be obtained by equalizing delay functions of different source to sink paths. For a given clock routing tree, we initially insert the same level of buffers in all the source to sinks paths, then minimize the clock delay by optimizing buffer positions, and minimize skew by simultaneous buffer level augment and buffer sizing.

Key words: clock distribution; buffer insertion; clock routing; clock delay; clock skew

1 引言

随着 VLSI 工艺技术的突飞猛进, 集成电路工作频率越来越高, 微处理器时钟频率已经达到数 GHz. 高性能、高速度 VLSI 急剧增长的性能需求对高速时钟网络设计提出了严峻挑战, 其关键问题是时钟延时和时钟偏差的最小化. 优化时钟延时和时钟偏差可以通过良好的布线方法[2]和插入缓冲器[3,4]来实现. 前期研究工作[2]曾提出使时钟源到各接受端点的路径长度相等来获得最小时钟偏差. 这些算法需要降低最快路径的信号传输速度来减小时钟偏差, 因而不能有效地缩短最大时钟延时. 插入缓冲器对减小时钟延时、时钟偏差以及减少传输线效应十分有效, 但也会占用更多的硅片面积, 增加芯片的功耗. 对给定的时钟树, 为了获得最小的时钟延时、时钟偏差和芯片功耗, 必须确定最优的缓冲器插入位置、数目和尺寸.

文献[5,6]提出在一棵时钟树的分支点上或分支点后直接加入缓冲器. 本文提出的最优缓冲器布局理论表明, 为获取最小时钟延时, 最优缓冲器插入位置并不仅仅局限于这些分支点处. 文献[7,8]提出在每条源至各接受端点的路径上插入等数目的缓冲器, 且同层缓冲器尺寸相同. 这种算法有助于降低时钟偏差对工艺参数变化的灵敏度[9]. 但它要求布线树具有平衡的拓扑结构, 否则不能有效减小时钟偏差至任意给定的设计指标. 然而在实际芯片设计中, 当存在预布局单元模块时, 构建等长路径树或完全平衡树往往无法实现[10].

本文将针对时钟树既非平衡也非路径等长的实际电路设计环境, 研究以获取最小时钟延时和时钟偏差为目标的最优的缓冲器布局和尺寸设计的理论及算法. 本文第二节介绍相关术语和延时模型, 第三节简述延时和偏差最小化的总体缓冲器插入方法, 第四节阐述初始缓冲器插入算法, 第五节提出延时最小化的最优缓冲器位置理论和算法, 第六节详述偏差最小化算法, 第七节给出实验结果和结论.

收稿日期: 2000-07-26; 修回日期: 2001-03-20
基金项目: 国家 863 计划 (863-SOC-Y-2-61; 863-SOC-Y-3-3); 自然科学基金海外青年学者合作研究基金 (No. 69928402); 自然科学基金 (No. 69806004); 教育部高等学校博士学科点专项科研基金 (No. 2000024628)

2　术语定义及延时模型

2.1　术语定义

定义 **1**(无缓冲器布线树)　无缓冲器布线树是有向二叉树 $T(V, E)$, 它包括连线集合 E 和节点集合 $V=\{\{s_o\}\cup SI\cup IN\}$, 其中 s_o 是唯一的源结点, IN 是分支结点集合, SI 是接收端点集合.

定义 **2**(带缓冲器布线树)　插入缓冲器后的布线树记为带缓冲器布线树 BT, 插入的缓冲器成为新的结点, 称之为缓冲器结点. BT 的结点集合重新定义为$\{\{s_o\}\cup SI\cup IN\cup BN\cup\}$, 其中 BN 是缓冲器结点集合.

定义 **3**(缓冲器层次)　在时钟树 T 里, 如果 k 个缓冲器插入到一条源点到接收端点的路径 $P(s_0, s_i)$ 中, 我们就说在路径 $P(s_0, s_i)$ 上有 k 层缓冲器. 从源点 S_0 到接受端点 s_i 路径上的第j 个缓冲器称为第j 层缓冲器.

2.2　时延模型

图 1 给出本文使用的分布式 RC 连线时延模型(图 1(a), (b))和缓冲器 RC 网络模型(图 1(c), (d)). 线电阻 r_e 和线电容 c_e 由方程(1)(2)确定.

$$r_e(l_e)=r_s\cdot l_e/w_e=r_d l_e \tag{1}$$

$$c_e(l_e)=c_a l_e w_e+2c_f(l_e+w_e)=(c_a w_e+2c_f)l_e=c_o l_e \tag{2}$$

其中, $l_e\gg w_e$, l_e 和 w_e 分别是连线 e 的长和宽. 常量 r_s, c_a 和 c_f 分别是单位长度和宽度连线的方块电阻、面电容和边缘电容. r_o 和 c_o 相应为单位长度连线的电阻和电容.

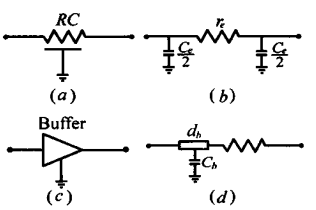

图 1　时延模型. (a)分布式 RC 连线; (b)等效 π 模型; (c)缓冲器; (d)缓冲器模型

设路径 $P(v_0, v_n)$ 由 n 条连线 e_{v_1}, e_{v_2}, …, e_{v_n} 构成, 则路径 $P(v_0, v_n)$ 的延时由式(3)确定, 这一延时就是 Elmore 时延[1]. 其中 $r_{e_{vi}}$ 和 $c_{e_{vi}}$ 分别是连线 e_{vi} 的电阻和电容, $C(T_{V_i})$ 是以 v_i 为根结点的子树 T_{V_i} 的集总电容.

$$\tau(v_0, v_n)=\sum_{i=1}^{n}\tau(e_{v_i})=\sum_{i=1}^{n}r_{e_{vi}}\cdot\left(\frac{c_{e_{vi}}}{2}+c(T_{v_i})\right) \tag{3}$$

缓冲器延时 $\tau_b=d_b+r_b\cdot c_b$, 其中 d_b, r_b, c_b 分别是缓冲器的本征延时、输出电阻和输入电容.

3　缓冲器优化布局及变尺寸理论和算法

本节首先定义优化缓冲器插入问题, 然后给出获得最小时钟延时和偏差的三步算法.

缓冲器插入问题　给定一棵已布线的时钟树 T^0 和缓冲器尺寸范围 $r_{b\max}\sim r_{b\min}$, 确定待插入的缓冲器的个数、位置及其大小, 以满足时钟延时和偏差最小的设计要求.

为解决上述缓冲器插入问题, 只需考虑 k 层缓冲器插入问题, 即当每条路径仅有 k 个缓冲器时, 寻找最优的缓冲器位置和大小问题. 因为只要使 k 值从 1 变化到任意期望的数值, 重复调用 k 层缓冲器插入算法就可以得到最优的 k 值. 以下给出 k 层缓冲器插入的三步方法.

(1) 初始缓冲器插入　对给定的无缓冲器时钟树 T^0, 初始缓冲器插入算法在每条源到各接收端点路径上插入 k 层缓冲器, 生成带缓冲器时钟树记为 BT. (2) 延时最小化　在不增加缓冲器个数和不改变缓冲器尺寸的情况下, 通过移动缓冲器位置实现时钟延时最小化. 这一步的目标是得到具有最小延时的带缓冲器时钟树 BTD. (3) 偏差最小化　保持最小延时路径上的缓冲器的尺寸和位置不变. 通过改变其他缓冲器的尺寸或者在必要的情况下增加缓冲器个数来减小偏差. 注意, 这个过程中的最小延时是保持不变的. 本步骤得到的最小延时和最小偏差的时钟树记为 BTDS. 下面我们依次给出上述三步过程的详细算法.

4　初始缓冲器插入

采用自上而下的方法, 在每条从源到各接收端点路径上插入 k 层缓冲器. 开始所有缓冲器的大小均选择中间值. 初始插入算法的伪代码见表(1).

表 **1**　k 层初始缓冲器插入算法 IBI

Input: An unbuffered tree T^o and buffer level k;
Δx: buffer moving step;

Output: Buffered Tree BT;

```
Procedure InitialInsertion (T^o, k)
  for i ←0 to k− 1 do
    BuffSetNew ←NULL;
    if i= 0 then
      T_p ←T^o;
      InsertBuffer();
    else
      for each buffer B' ∈ BuffSetOl do
      T_p ←subtree rooted at B';
      InsertBuffer();
    end for;
  end if;
    BuffSetOld ←BuffSetNew;
  end for;
end Procedure;
Procedure InsertBuffer()
  Repeat
In T_p find path p(u, v) with the shortest path Length
  l_min, excluding the path which contains a buffer;
  Insert a buffer B_new into p(u, v) at 1/(k− i+1) l_min;
  Add B_new to the set of BuffSetNew;
Until there is one buffer inserted into every path of T_p
end Procedure;
```

定理 **1**　逐层初始缓冲器插入算法 IBI 在每条源到各接收端点路径上插入相同数目的缓冲器.

证明　按照 IBI 算法, 当插入某一层缓冲器后, 每一条从源点到各接收端点的路径上属于该层的缓冲器数目不多于一个. 同时, 当插入某一层缓冲器时, 每一条源到接收端点路径都作为候选路径, 除非这条路径已经插入了属于该层的缓冲器. 这就意味着当一层缓冲器插入完成后, 每条源到接收端点的路径上只插入了一个属于该层的缓冲器, 故经过 k 层缓冲器插入后, 每一条源到各接收端点的路径上有相同数目的缓

冲器.

5 延时最小的优化缓冲器布局

在延时最小的缓冲器布局中, 当一条路径上缓冲器的位置移动时, 其他路径的时延会受到影响. 为有效解决这一问题, 我们首先研究带缓冲器布线树中单一路径上的缓冲器布局问题, 提出了优化缓冲器位置理论和最小化路径延时的缓冲器分裂理论. 在此基础上, 我们提出了一个带缓冲器树延时最小化的迭代算法.

5.1 单路径延时最小化

下面我们研究当其它缓冲器位置不变时, 如何确定两个分支结点之间的缓冲器最优位置来获得最小延时.

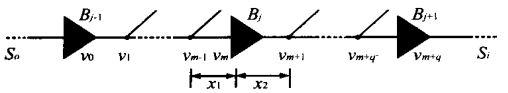

图 2 一条带缓冲器的子路径

5.1.1 延时最小化的最优缓冲器位置 不失一般性, 考虑一棵带缓冲器的时钟树中一条从源点 s_o 到接收端点 s_i 的路径 $P(s_o, s_i)$(如图 2), 在缓冲器 B_{j-1} 和 B_j 之间有 $m-1(m \geq 1)$ 个分支结点 $v_1, v_2, \cdots, v_{m-1}$, 在 B_j 和 B_{j+1} 之间有 $q-1(q \geq 1)$ 分支结点 $v_{m+1}, v_{m+2}, \cdots, v_{m+q-1}$. 令 v_0, v_m 和 v_{m+q} 分别表示三个缓冲器结点, 设缓冲器 B_j 位于 v_{m-1} 和 v_{m+1} 之间, 并且 x_1 和 x_2 分别表示路径 $e(v_{m-1}, v_m)$ 和 $e(v_m, v_{m+1})$ 的长度, 则路径 $e(v_{m-1}, v_{m+1})$ 的长度 L_j 为一常量:

$$x_1 + x_2 = L_j \tag{4}$$

利用第二节的延时模型, 可以得到从 v_0 到 v_{m+q} 的延时 $\tau(v_0, v_{m+q})$ 如方程(5), 它是变量 x_1 的二次函数:

$$f(x_1) = \tau(v_0, v_{m+q}) = \alpha_1 x_1^2 + \beta_1 x_1 + \Gamma_1 + \alpha_2 (L_j - x_1)^2 + \beta_2 (L_j - x_1) + \Gamma_2 \tag{5}$$

对给定的路径, 参数 $\alpha_1, \alpha_2, \beta_1, \beta_2, \Gamma_1, \Gamma_2$ 是常数. 当函数 $f(x_1)$ 的导数为 0 时, 延时 $\tau(v_0, v_{m+q})$ 具有最小值. 此时缓冲器的位置由下式求出:

$$x_1 = \frac{\beta_2 - \beta_1 + 2\alpha_2 L_j}{2(\alpha_1 + \alpha_2)} \tag{6}$$

我们得到以下最小延时的缓冲器优化位置理论.

定理 **2**(缓冲器优化位置定理) 在带缓冲器的时钟树中的一条从源到接收端点的路径上, 相邻缓冲器之间(或者是源点到第一个缓冲器, 或者是最后一个缓冲器与接收端点之间)的延时是缓冲器在这条路径上位置的凸函数; 当所有延时函数的导数相同时, 路径的延时最小.

定理 2 告诉我们, 一个缓冲器能够连续移动到使路径时延最小的最优位置. 但是如果最优位置不在两个相邻的分支点之间, 则可能需要移动缓冲器到分支点之外. 在这种情况下, 延时函数不再是缓冲器位置的连续函数, 为此需要采取以下提出的缓冲器分裂理论来保证延时函数的连续性.

5.1.2 缓冲器分裂理论 考察如图 3 所示的一般情况的例子, 由定理 2, 为了减小路径延时, 缓冲器 B_j 沿路径向右连续移动, 当缓冲器移过分支点 v'_m 时, 为了保证延时函数的连续性, 需要将一个缓冲器分裂为两个更小的缓冲器, 分别位于两个分支路径上. 如图 2 所示, 缓冲器分裂之前, 函数 $\tau(v_0, v_{m+1})$ 中由缓冲器 B_j 的输出电阻 r_{b_j} 和输入电容 C_{b_j} 决定的延时部分由下式给出

$$\tau_1 = R_p C_{b_j} + r_{b_j}(C_{p1} + C_{p2}) \tag{7}$$

其中 R_P 是路径 $P(v_0, v_m)$ 的电阻, C_{P1}, C_{P2} 是以 v'_m 为根结点的两棵子树的集总电容. 当缓冲器 B_j 移到结点 v'_m 后的两个分支上时, 它分裂为两个缓冲器 B_{j1}, B_{j2}. 这时延时函数 $\tau(v_0, v_{m+q})$ 中由缓冲器 B_{j2} 的输入电容 $C_{b_{j2}}$ 和输出电阻 $r_{b_{j2}}$ 决定的延时部分 τ_2 为

$$\tau_2 = R_p(C_{b_{j1}} + C_{b_{j2}}) + r_{b_{j1}} C_{p2} \tag{8}$$

其中 $C_{b_{j1}}$ 是缓冲器 B_{j1} 的输入电容. 为保证连续性, 必须使 $\tau_2 = \tau_1$, 即

$$C_{b_j} = C_{b_{j1}} + C_{b_{j2}} \tag{9}$$

$$r_{b_j}(C_{p1} + C_{p2}) = r_{b_{j2}} C_{p2} \tag{10}$$

设 $C_{p1} = \gamma C_{p2}$, 我们得到分裂的两个缓冲器的尺寸关系:

$$W_{b_{j1}} = \frac{\gamma}{1+\gamma} W_{b_j} \quad W_{b_{j2}} = \frac{1}{1+\gamma} W_{b_j} \quad L_{b_{j1}} = L_{b_{j2}} = L_{b_j} \tag{11}$$

式中 L_{b_j}, $L_{b_{j1}}$, $L_{b_{j2}}$ 是缓冲器 B_j, B_{j1}, B_{j2} 的晶体管栅长, W_{b_j}, $W_{b_{j1}}$, $W_{b_{j2}}$ 是晶体管的栅宽.

定理 **3**(缓冲器分裂定理) 在一棵带缓冲器的时钟树中, 当一个缓冲器移过分支点时, 如果将其分裂为两个尺寸满足方程(11)的缓冲器, 则时钟树中所有路径的延时保持不变.

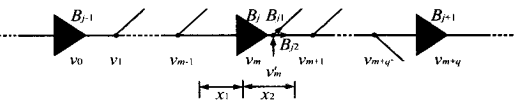

图 3 缓冲器分裂方法

5.1.3 单路径延时最小化 运用定理 2、3, 我们给出如下的单路径延时最小化的迭代算法. 从源到各接收端的路径上, 算法依次逐个选取缓冲器, 对选取的缓冲器移动其位置并测试这样的移动是否能减小路径延时. 如果改变缓冲器的位置能够减小延时, 则将缓冲器移动到这个新位置; 否则其位置保持不变. 当路径上最后一个缓冲器的测试完成时, 算法回到第一个缓冲器, 开始新的迭代, 直到路径延时不再减小. 如果缓冲器移过分支点, 按式(11)分裂缓冲器.

5.2 时钟树延时最小化

时钟树延时最小化算法是基于单路径延时最小化算法. 但是与单路径延时最小化相比, 树延时最小化的处理更加困难, 原因是对某一路径的延时进行最小化处理将影响其它已经过延时最小化处理的路径时延. 为此必须提出为保证最小化过程收敛性的有效算法.

5.2.1 树延时最小化算法 设一棵带缓冲器的时钟树有 q 条路径, $P_{\min}$ 为最小延时路径, $\tau_{\min}$ 是最小延时. 令 P_i 为第 i 条待进行最小化处理的路径, τ_{p_i} 是路径 P_i 的延时, 算法如下:

步骤 **1** 按路径延时递增顺序 $\tau_{p_i}(\tau_{\min}) < \cdots < \tau_{p_i} < \cdots < \tau_{p_q}$ 排列带缓冲器时钟树中所有路径.

步骤 **2** 令 i 从 1 变化到 q, 在缓冲器移动的约束条件

(详见(5.2.2))下, 用单路径延时最小化算法优化路径 P_i 的延时.

步骤 3　重复步骤 1 和步骤 2, 直到时钟树的延时不再减少.

5.2.2　延时单调降低的移动约束　设 P_i 是待优化延时的路径, 为保证延时单调递减和算法收敛, 缓冲器移动过程必须受以下三个约束条件的限制. (1) 如果缓冲器同时在路径 P_i 和最小路径 $P_{\min}$ 上, 缓冲器不能移动; (2) 在路径 P_i 上, 从源到接收端方向的第一个不在路径 $P_{\min}$ 上的缓冲器不能沿路径移动; (3) 路径 P_i 上不属于(1)和(2)两种情况的缓冲器可以上下移动, 但不能移出分支节点, 如果移出分支节点需要按定理 3 将其分裂为两个缓冲器.

时钟树延时最小化算法利用单路径延时最小化算法逐条优化路径延时来优化树延时. 显然, 由于在调整缓冲器的位置时引入上述约束条件, 保证了时钟树延时在移动过程中单调减少. 优化后生成具有最小延时的带缓冲器的时钟树 BTD.

6　时钟偏差最小化算法

通过在非最小延时路径上, 增加缓冲器尺寸或者增加缓冲器数目(如果有必要的话), 使这些路径的延时趋向最小的路径延时, 实现时钟偏差最小化. 为了保证这种方法不增加最小延时, BTD 树最小延时路径上的缓冲器的大小和位置都必须保持不变, 同时为了保证算法的收敛性, 我们将对缓冲器的尺寸和数目作一些约束.

6.1　时钟偏差最小化算法

步骤 1　在 BTD 树中按延时递增顺序 $\tau_{p_i}(\tau_{\min}) < \cdots < \tau_{p_i} < \cdots < \tau_{p_q}$ 排列所有路径;

步骤 2　令 i 从 2 变化到 q, 在当前路径 P_i 上, 找到增加缓冲器数目或调整其尺寸大小的子路径. 在子路径上增加缓冲器个数或调整缓冲器尺寸来减小路径 P_i 的延时, 以趋向最小延时;

步骤 3　重复步骤 1 和步骤 2 直到时钟偏差满足设计指标.

算法首先按路径延时递增顺序对路径排序, 然后依次从队列中逐条选出待延时最小化的路径(除了最小延时路径); 在当前路径 P_i 上, 从接收端点到源点搜索到第一个已进行优化处理的缓冲器, 则从该缓冲器到接收端点的子路径为待优化的子路径. 待优化子路径上的缓冲器为待优化缓冲器. 当待优化的子路径上的缓冲器的优化完毕后, 路径 P_i 上所有的缓冲器的大小和位置将固定下来, 这样后继路径的优化不改变早先已优化的路径的时延.

方程(12)用于测试当所有待优化缓冲器的尺寸设为最大时, 所考察的路径的时延是否能够减小到指定的偏差容限之内. 如果式(12)成立, 通过调整缓冲器大小就可以满足偏差指标要求, 否则需要在路径中插入缓冲器.

$$\sum_{j=1}^{m}(r_{lj} - r_{bmin})C_j \geq \tau_{pi} - \tau_{\min} - \tau_{skew} \qquad (12)$$

式(12)中, m 为待优化的缓冲器数目, C_j 表示以第 j 个缓冲器为根结点的子树的集总电容, r_{lj} 表示缓冲器输出电阻, τ_{skew} 为偏差容限. 详细的缓冲器变尺寸算法将在下面给出.

6.2　缓冲器尺寸调整过程

假设子路径上有 m 个待优化缓冲器, 沿着这条子路径自上而下, 逐个增加待优化缓冲器的尺寸至最大值, 直至当前路径的延时小于树的最小路径延时. 当前缓冲器的尺寸由式(13)确定:

$$\sum_{j=1}^{n-1}(r_{lj} - r_{bmin})C_j + (r_{bn} - \tilde{r}_{bn})C_n = \tau_{p_i} - \tau_{\min} \qquad (13)$$

这里 $n-1(1 \leq n \leq m)$ 是增加到最大尺寸 r_{bmin} 的缓冲器的数目, $\tilde{r}_{bn}$ 是当前考察(记为第 n 个缓冲器)的缓冲器尺寸.

6.3　增加缓冲器的算法

当待优化子路径上所有缓冲器都增加到最大尺寸时, 偏差仍在指定容限之外, 则需要在子路径上增加额外的缓冲器, 下面给出增加缓冲器数目的算法:

步骤 1　应用 k 层初始缓冲器插入算法 IBI 在待优化子路径上多插入一层缓冲器;

步骤 2　应用树延时最小化算法优化待优化子路径上缓冲器的位置;

步骤 3　应用缓冲器变尺寸的算法优化待优化子路径上缓冲器的大小;

步骤 4　重复步骤 1 到步骤 3 直到偏差减小到指定容限之内.

7　实验结果

本文提出的算法用 C 语言在 SPARC20 工作站上实现. 对存在布线障碍的设计实例, 采用文献[10]的时钟布线算法, 得到既非平衡树又非等路径长的时钟树. 对这样的时钟树, 利用本文的算法进行缓冲器插入, 得到下面的实验结果. 测试芯片面积为 1cm × 1cm, 时钟布线树的接收端点数目分别为 50, 100, 300 和 500. 表 2 列出了实验用到的工艺参数. 表 3 给出插入缓冲器前后的延时和偏差结果. 对插入不同层数的缓冲器进行测试, 得到了最优的缓冲器插入层数. 表中延时为源点到接收端点的信号传输时间, 也就是各子树的 Elmore 延时和该源到接收端路径上的缓冲器的延时之和. 结果表明, 采用我们提出的方法后时钟延时和偏差减小了至少一个数量级. 测试例子的时钟偏差在 100ps 范围内, 满足工作于 GHZ 的时钟网络所必须的偏差要求.

表 2　工艺参数

片电阻 0.14Ω/□	面电容 0.08fF/μm²	最大缓冲器输出电阻 R 5Ω	最大缓冲器输入电容 C 0.01pF	最大缓冲器本征延时 100ps
线宽 10μm	边缘电容 0.03fF/μm	最小缓冲器输出电阻 R 100Ω	最小缓冲器输入电容 C 0.01pF	最大缓冲器本征延时 30ps

表 3　插入缓冲器前后的结果

芯片面积	宿点 #	最长路径线长(cm)	初始插入缓冲器层数	缓冲器插入前的延时, 偏差 τ_0	τ_{s0}	最小化后的值延时, 偏差 τ…	τ_s	改进后的延时和偏差 impv1	impv2
1cm × 1cm	50	2.24	4	4422.08	430.62	707.48	47.25	6.25	9.11
	100	2.06	4	5668.06	169.39	674.31	52.27	8.41	3.24
	300	2.01	5	11445.65	124.60	710.10	48.82	16.12	2.55
	500	2.32	4	14794.05	1248.5	660.70	72.32	22.39	17.26

* 延时和偏差的单位为 ps, 其中 impv1 = τ_0/τ, impv2 = τ_{s0}/τ_s

8 结论

本文提出了一种最优化的缓冲器布局和尺寸调整算法. 我们导出在时钟树中缓冲器之间的延时是缓冲器位置变量的凸函数; 通过使凸函数的导数相等, 可以实现时钟延时最小; 通过使不同源到接收端路径的延时函数相等, 可以实现时钟偏差最小. 我们进一步提出初始缓冲器插入算法, 时钟延时最小的缓冲器布局算法, 以及时钟偏差最小的缓冲器的层数增加和尺寸调整算法, 以同时获得最小的时钟延时和时钟偏差. 实验结果证明了方法的正确性和有效性.

参考文献:

[1] W C Elmore. The transient response of damped linear network with particular regard to wideband amplifier [J]. J. Applied Physics, 1948, 19: 55- 63.

[2] A Kahng, J Cong, G Robins. High-performance clock routing based on recursive geometric matching [A]. 28th ACM/IEEE Design Automation Conference [C], 1991, 332- 337.

[3] B Wu, N Sherwani. Effective Buffer insertion of Clock Tree for High-Speed VLSI Circuits [J]. Microelectronics Journal, July 1992, 23: 291 - 300.

[4] J Lillis, C K Chen, T T Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model [A]. Proc. IEEE Int. Conf. on Computer-Aided Design [C], 1995: 138- 143.

[5] S Pullela, N Menezes, J Omar, L Pillage. Skew and delay optimization for reliable buffered clock trees [A]. Proc. IEEE Int. Conf. on Computer-Aided Design [C], 1993: 556- 562.

[6] J Chung, C K Cheng. Skew sensitivity minimization of buffered clock tree [A]. Proc. Int. Conf. on Computer-Aided Design [C], 1994: 280 - 283.

[7] Y P Chen, D F Wong. An algorithm for zero-skew clock tree routing with buffer insertion [A]. Proc. European on Computer-Aided Design [C], 1994: 219- 223.

[8] J D Cho, M Sarrafzadeh. A buffer distribution algorithm for high-speed clock routing [A]. Proc. ACM/IEEE Design Automation Conf [C], 1993: 537- 543.

[9] Joe G Xi, Wayne W M Dai. Buffer insertion and sizing under process variations for low power clock distribution [A]. ACM/IEEE Design Automation Conference [C], 1995: 491- 496.

[10] H Kim, D Zhou. An automatic clock tree design system for high-speed VLSI designs: planar clock routing with the treatment of obstacles [A]. International Symposium on Circuits and Systems [C], 1999.

作者简介:

曾 璇 女. 1969 年 4 月出生于湖北省黄石市. 1991 年获复旦大学电子工程系学士学位, 1997 年获复旦大学电子工程系博士学位, 现为复旦大学电子工程系副教授. 研究方向包括 VLSI 高速互连电路设计, 模拟电路行为级模拟、建模和版图设计自动化.

周丽丽 女. 1977 年 12 月出生于江苏省盐城市. 1999 年获复旦大学电子工程系学士学位, 现为复旦大学电子工程系硕士研究生. 研究方向为集成电路计算机辅助设计.