

带缓冲器的 3D-IC 时钟布线

封成冬, 王 琴, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200240)

摘 要: 为了在基于 TSV 的 3D-IC 中实现成本效率高的时钟树布线, 介绍一个 3D 时钟树综合算法. 对于一个给定抽象时钟树的拓扑结构, 给出了一个 3D 时钟树嵌入算法来最小化 TSV 个数. 如果没有给定抽象树拓扑结构, 提出了一个 NN-3D 算法来生成抽象树. 最后, 插入缓冲器来进一步降低时钟树的延时以及最大负载电容. 这几个步骤连接起来就形成一个完整时钟树综合算法. 通过 Matlab 建模验证, 这个算法在布线总长度、延时、功耗以及 TSV 个数等各方面综合考量下获得了很好的效果, 进一步降低了 3D-IC 的成本以及功耗散热问题.

关键词: 3D-IC; 时钟树综合; TSV

中图分类号: TN4

文献标识码: A

文章编号: 1000-7180(2014)08-0098-06

Buffered Clock Tree Routing for 3D-IC

FENG Cheng-dong, WANG Qin, XIE Jing, MAO Zhi-gang

(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: To get a cost-effective implementation of clock trees in 3D-IC designs based on TSV (Through-Silicon Via), a 3D clock tree synthesis algorithm is proposed. For a topology given abstract tree, we introduce a 3D clock tree embedding algorithm to minimize the number of TSVs. If abstract tree is not given, we introduce a nearest-neighbor-3D algorithm to generate an abstract clock tree. Finally, buffers are inserted to reduce delay and maximum load capacitance. These steps make the clock tree synthesis algorithm. Through experiment, we confirm that the clock tree synthesis with this algorithm is very effective in terms of total wirelength, delay, power consumption and the number of TSVs. Thus, it relieves the thermal and cost issues of 3D-IC.

Key words: D-IC; clock tree synthesis; TSV

1 引言

3D-IC 堆叠技术在减小传输延迟、线长以及面积方面表现出了巨大的潜力, 可以改善芯片的整体表现和成本. 基于硅通孔 (TSV) 的 3D-IC 设计被认为是未来一代 IC 产品的关键技术, 但其高密度以及低导热等原因, 散热一直是一个需要面对的问题.

最近几年, 对于 3D-IC 的时钟树设计有很多研究, Mondal 等人^[1]提出了一个热适应的时钟方案来降低时钟漂移对温度的依赖性, 主要思想是通过动态调整温度敏感的时钟缓冲器的驱动力度来减小时钟漂移; Arunachalam 和 Burleson^[2]通过给时钟网

络加一个分离层来实现低功耗时钟设计. Minz 等人^[3]提出了一个考虑热的带缓冲器 CTS 方法来最小化以及平衡漂移, 主要方法是调整时钟树节点的位置使得时钟漂移在两个极端热条件下能够平衡并且最小化; Kim 和 Kim^[4]提出了零漂移时钟树嵌入算法保证 TSV 数目最小化. 以上研究没有综合考虑线长、延迟以及 TSV 数目的设计方案. 一般来说, 如图 1 所示, 3D 时钟树综合流程包含一系列的步骤: (1) 抽象树拓扑结构生成; (2) 把树节点分配到各层; (3) 时钟树布线 (嵌入); (4) 插入缓冲器. 其中时钟树布线和插入缓冲器经常同时进行, 以满足漂移的限制.

收稿日期: 2013-10-28; 修回日期: 2013-12-21

基金项目: 国家自然科学基金项目 (61176037); 教育部重点实验室主任基金资助

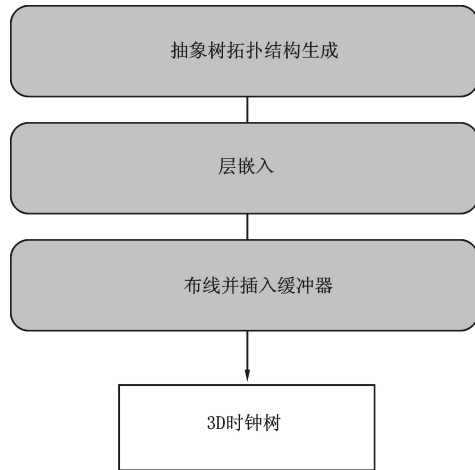


图 1 3D 时钟树综合流程

Kim T. Y. 和 Kim T^[5] 提出了一种 NN-3D (3D-IC 的最近邻选择) 算法来生成时钟树拓扑结构, 对于最近邻的选择, 他们做了一个假设, 合并节点的嵌入层假定为其两个子节点的中点. 本文介绍一种综合考虑线长、延迟以及 TSV 数目的时钟树综合算法, 对上述提出的假设进行改进, 使其更符合实际合并情况.

2 3D 时钟树嵌入算法

在使用 3D 设计空间的物理设计中, 时钟树的末端可以分配在多个 die 上面, 并且一个时钟树可以生长到所有的 die 上面. 因此, 在构建 3D 时钟树的时候不仅需要考虑到线长、缓冲器个数、能耗这些, 还需要考虑 TSV 个数. 表 1 所示为一些标记的定义, 需解决的问题定义如下所示.

问题 1: 3D 时钟树综合

给定一个由末端的位置构成的集合 S , 构建一个抽象树拓扑 $G(S)$ 以及一个基于 $G(S)$ 的嵌入后的时钟树 $T(S)$, 使得一个关于总线长、TSV 数目以及缓冲器数目的加权和最小, 即最小化下面的式子:

$$Cost = w_1 \cdot \sum_{\forall x} |e_x| + w_2 \cdot nv(T_{\forall}) + w_3 \cdot \# buffers \quad (1)$$

同时, 还要满足零漂移的限制. 其中, $|e_x|$ 、 $nv(T_{\forall})$ 以及 $\# buffers$ 分别是树节点 x 和其父节点之间的距离、TSV 的数目以及 $T(S)$ 中 buffer 的数目, w_1 、 w_2 、 w_3 是用户定义的权重参数.

在式(1)中, 权重参数可以是制造成本或者设计成本. 用单位电容作为权重参数, 这样可以产生一个不错的设计成本的评估, 因为面积和功耗往往正比于电容值.

表 1 标记的定义

标记	定义
s_0	时钟树的根, 可以理解为时钟源
S	末端 (sink) 组成的有限集 ($= (s_1, s_2, \dots, s_n) \subset R^3$)
$G(S)$	以 S 中的末端为叶子的二叉树拓扑
$T(S)$	$G(S)$ 嵌入以后的时钟树拓扑
T_x	以节点 x 为根的嵌入后时钟树
$parent(x)$	二叉树中节点 x 的父节点
$left(x)$	二叉树中节点 x 的左子节点
$right(x)$	二叉树中节点 x 的右子节点
$t(a, b)$	点 a 以到点 b 的传输延迟
t_x	从 T_x 的根到其末端的延时
$t(v, T_x)$	从节点 v 到 T_x 末端的延时
$skew(T_x)$	T_x 的所有末端产生的时钟漂移 ($= \max\{ t(x, s_i) - t(x, s_j) \}$)
r_w, c_w	单位线长的电阻和电容
r_v, c_v	单位 TSV 的电阻和电容
r_b, c_b	缓冲器的输出电阻和输入电容
C_x	节点 x 的下游电容
$d(a, b)$	节点 a 和节点 b 之间的曼哈顿 (Manhattan) 距离
$d(A, B)$	两个点集 A 和 B 之间的曼哈顿距离 ($= \min\{d(a, b); a \in A \text{ and } b \in B\}$)
$l_x (= i)$	节点 x 的放置层或嵌入层, 为从底层算起的第 i 层
$el(x)$	节点 x 的可能嵌入层, 代表从 l_1 到 l_2 的一个层集合
$nv(x_1, x_2)$	连接节点 x_1 和 x_2 的 TSV 数量 (按照单位 TSV 来计量), 可以认为是 $ l_{x_1} - l_{x_2} $
$nv(T_x)$	T_x 中的 TSV 数量 (按照单位 TSV 来计量)
$pl(x)$	节点 x 的坐标 (x, y) 放置位置
$ms(x)$	节点 x 的合并段
e_x	从 $parent(x)$ 到节点 x 的有向边
$ e_x $	从 $parent(x)$ 到节点 x 的线长
trr_x	以 x 的合并段 $ms(x)$ 为核形成的倾斜长方形区域 所有和 $ms(x)$ 的曼哈顿距离在一个确定值以内的点构成这个区域

为了解决问题 1, 介绍一个给定时钟树拓扑以后的 3D 时钟树嵌入算法 (之后还会介绍一个时钟树拓扑生成的算法).

我们按两个步骤来为时钟树进行嵌入. 首先需要确定树节点的嵌入层, 保证可以最小化 TSV 数目; 其次, 是在最少 TSV 的基础上用最短的线长进行时钟树布线. 这两步可以将其变为下面两个问题:

问题 2: 最小 TSV 数目的层嵌入

给定一个由末端的位置构成的集合 S 以及一个树拓扑 $G(S)$, 确定树节点的嵌入层使得需要安放的 TSV 个数最少. 注意到树节点的嵌入层直接决定了哪条边是需要 TSV 的.

问题 3: 3D 零漂移时钟布线

给定一个由末端的位置构成的集合 S 以及一个确定节点嵌入层的树拓扑 $G(S)$, 构建一个嵌入后的时钟树 $T(S)$, 在保证零漂移的前提下最小化总的线长.

2.1 最小 TSV 个数的层嵌入法

在 3D 时钟树嵌入过程中,需要插入 TSV 来连接处于不同晶圆的末端,图 2(a)是一个二叉时钟树,有四个末端,其中 s_1 和 s_3 在第一层, s_2 和 s_4 在第二层. 对于这样的一个结构,图 2(b)给出了树节点 a, b 和 v 可能被放置的嵌入层,总共有 8 种可能的分配方式,最后一行给出了每一种分配方式所需要的 TSV 个数,可以看出,最佳的分配方式是 $\langle a, b, v \rangle = \langle 1, 1, 1 \rangle$ 以及 $\langle 2, 2, 2 \rangle$,所需要的 TSV 个数都是两个.

将二叉树 T_x 中 TSV 的个数以递归的形式表达:

$$nw(T_x) = \begin{cases} 0, & \text{if } x \text{ is sink} \\ nw(T_{left(x)}) + nw(T_{right(x)}) + \\ nw(x, left(x)) + nw(x, right(x)), & \text{otherwise} \end{cases} \quad (2)$$

也就是以 x 的子节点作为根的子树里分布的 TSV 个数加上从 x 点连接到它的子节点所需要的 TSV 个数. 如果 x 的子节点都是末端,那么根据末端所在层,就可以找到一个关于 x 所在层的集合 $el(x)$,只要 x 在这个集合内,就可以使得 $nw(T_x)$ 最小. 另一方面,如果 x 的子节点不都是末端,那么通过自底向上递归过程,子节点的所在层集合是可以得到的,这样,通过子节点的所在层集合,就可以得到 x 的所在层集合 $el(x)$,使得 $nw(T_x)$ 最小. 因此,用自底向上递归的方法来计算每个点的嵌入层集合:

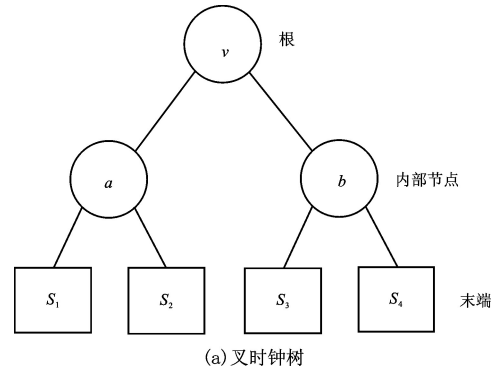
$$\begin{aligned} el(x) &= \langle \min(l_1, l_2), \max(l_1, l_2) \rangle \\ l_1 &= \max(\min(el(left(x))), \min(el(right(x)))) \\ l_2 &= \min(\max(el(left(x))), \max(el(right(x)))) \end{aligned} \quad (3)$$

下面定义帮助我们证明这个算法的最优性.

定义 1 ① $|l_a - el(b)|$ 定义为 $\min\{|l_a - l_b|, "l_b \hat{=} el(b)\}$, 比如说 $|1 - \langle 2, 3 \rangle| = 1, |3 - \langle 2, 4 \rangle| = 0$; ② $|el(a) - el(b)|$ 定义为 $\min\{|l_a - l_b|, "l_a \hat{=} el(a), "l_b \hat{=} el(b)\}$, 比如说 $|\langle 1, 3 \rangle - \langle 2, 4 \rangle| = 0, |\langle 1, 2 \rangle - \langle 3, 4 \rangle| = 1$

定义 2 如果每个 $G(S)$ 中的节点 x 被分配的层都在式(3)所计算出的集合 $el(x)$ 中,那么,这样的一个关于 $G(S)$ 的节点的嵌入层分配就被称为是安全的.

引理 1 假设 $G(S)$ 的根 v 的高度 $h > 0$,那么 $G(S)$ 的一个不安全的嵌入层分配所得到的 $nw(T_v)$ 比安全的嵌入层分配中 $nw(T_v)$ 的最小值大 $|l_v - el(v)|$.



可能被放置的嵌入层							
a	1				2		
b	1		2		1		2
v	1	2	1	2	1	2	1
TSVs	2	4	3	3	3	3	4

(b) 对于给定的末端所在层,树节点 a, b 和 v

所有可能被放置的嵌入层

图 2 TSV 分配的一个例子

引理 2 一个 TSV 个数最小的安全嵌入层分配是最优的.

一旦通过式(3)确定了每个树节点的嵌入层集合,那么由引理 2,我们就希望为每一个节点分配一个 $el(v)$ 中的层,可以最小化 TSV 的个数. 我们以自顶向下,从根 s_0 到末端的顺序来为每个节点确定这个层. 现在,假设某个节点 v 的嵌入层已经确定了,那么其某个子节点 a 的嵌入层的确定方法就是: $el(a)$ 中,和 v 所在层垂直距离最近,也就是连接两点所需要穿过的层最少的那层,称这种确定方法为“最近嵌入层分配”.

定理 1 使用最近嵌入层分配的 3D 层嵌入算法所确定的 TSV 位置分配是最优的,也就是个数最少的.

这样总结起来,这个算法就可以分为两个阶段. 在第一个阶段,按照自底向上的遍历顺序,依次求出每个节点的嵌入层集合;在第二个阶段,按照自顶向下的遍历顺序,求出确切的嵌入层,也就是 $el(v)$ 中离 $parent(v)$ 所在层最近的那一层. 如果 v 是根,那么 l_v 可以取 $el(v)$ 中任意一个层. 因为每个节点都经历了两次计算,而且每次计算时间是常数,因此这个算法的时间复杂度为 $O(|V|)$,其中 V 是图 G 的点集合.

2.2 带 TSV 布局的 3D 时钟树布线

一旦树节点的嵌入层通过 3D 层嵌入算法确定以后,下一步就是完成时钟树嵌入,也就是确定每一层的树节点的精确坐标. 在描述 3D 时钟布线之前,先说明一下 3D 设计空间中两个子树的嵌入模型,

如图 3 所示. 为了方便起见, 2D 的曼哈顿平面被画成了 1D 的线, 节点 v 有两个子节点 a 和 b , 每个子节点都是通过 TSV 来连接的, TSV 的位置距离节点 a 为 x_{v1} 和 x_{v2} . 依靠这个模型, 将会得到节点 v 的位置并且使得线长在零漂移的限制下最小.

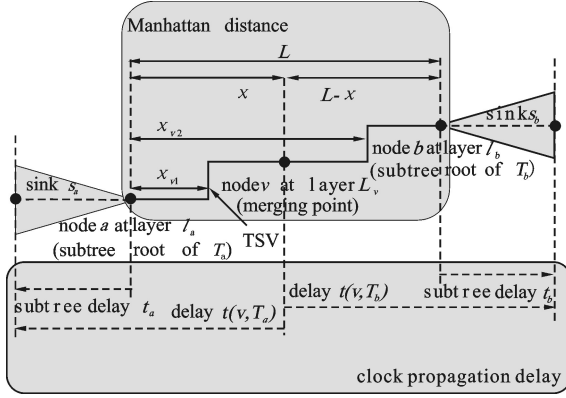


图 3 两个位于不同层的子树根到合并点的嵌入过程横切面视图

为了找到树节点的合并位置, 扩展了 2D DME 算法^[6], 只是在考虑过程中增加了 TSV 的延时. 在自底向上的阶段, 计算出 G 中树节点可能的使得线长 ($= |e_a| + |e_b|$) 最短的合并位置, 称为合并 (线) 段 $ms(v)$. 因为在最小 TSV 个数层嵌入算法中确定了每个树节点的嵌入层, 所以才可以用 2D DME 的算法来最小化线长, 当然, 由于 TSV 延时的存在, 计算出的合并段和 2D 的情况是不一样的, 所以, 就要建立一个新的考虑 TSV 以后的边长计算方法. 下面给出不同寄生参数以及位置的 TSV 所对应的不同的合并点计算公式.

为了确定 Elmore 延时模型下 3D 设计空间的合并段, 从节点 v 到子节点 a 和 b 的时钟传输延时公式由式 (4)、式 (5) 给出, 里面融合了 TSV 的位置和寄生参数信息. 注意到传输延时在 $-(r_w \cdot c_v - r_v \cdot c_w) \cdot (l_v - l_a) \cdot x_{v1}$ 和 $-(r_w \cdot c_v - r_v \cdot c_w) \cdot (l_b - l_v) \cdot (L - x_{v2})$ 最小的时候取到最小值, 因此, 对于 a 和 v 的连接来说, 如果 $(r_w \cdot c_v - r_v \cdot c_w) \geq 0$, 我们就会将 TSV 放在合并点 v 的地方来让 x_{v1} 最大; 如果 $(r_w \cdot c_v - r_v \cdot c_w) < 0$, 我们就会把 TSV 放在节点 a 的位置来减小 x_{v1} 到 0, 对于 b 和 v 的连接也类似. 这样, 在 $0 \leq x \leq L = d(ms(a), ms(b))$ 的情况下, 式 (6) 通过设定 $x_{v1} = x_{v2} = x$ 给出了合并点在 $(r_w \cdot c_v - r_v \cdot c_w) \geq 0$ 时的计算方法, 类似的, 可以设定 $x_{v1} = 0$ 以及 $x_{v2} = L$ 来处理 $(r_w \cdot c_v - r_v \cdot c_w) < 0$ 的情况. 另一方面, 如果 $x < 0$ 或者 $x > L$, 就需要进行绕路布线来实现零漂移合

并. 如果 $x < 0$, 那么我们通过设定 $|e_a| = 0$ 得到式 (7) 来计算 $|e_b| = L'$, 当然式 (7) 是在 $(r_w \cdot c_v - r_v \cdot c_w) \geq 0$ 的情况下. 如果 $x > L$, 那么类似的, 我们通过设定 $|e_b| = 0$ 来计算 $|e_a| = L'$.

$$t(v, T_a) = t_a + \frac{1}{2} r_w \cdot c_w \cdot x^2 + r_w \cdot (C_a + c_v \cdot (l_v - l_a)) \cdot x + r_v \cdot \left[C_a \cdot (l_v - l_a) + \frac{1}{2} c_v \cdot (l_v - l_a)^2 \right] - (r_w \cdot c_v - r_v \cdot c_w) \cdot (l_v - l_a) \cdot x_{v1} \quad (4)$$

$$t(v, T_b) = t_b + r_w \cdot L \cdot \left[C_b + \frac{1}{2} c_w \cdot L \right] + \frac{1}{2} r_w \cdot c_w \cdot x^2 - r_w \cdot (C_b + c_v \cdot (l_b - l_v) + c_w \cdot L) \cdot x + r_v \cdot \left[C_b \cdot (l_b - l_v) + \frac{1}{2} c_v \cdot (l_b - l_v)^2 \right] + r_w \cdot c_v \cdot (l_b - l_v) \cdot L - (r_w \cdot c_v - r_v \cdot c_w) \cdot (l_b - l_v) \cdot (L - x_{v2}) \quad (5)$$

$$x = \frac{(l_b - l_a) + r_w \cdot L \cdot \left[C_b + \frac{1}{2} c_w \cdot L \right] + \beta + r_v \cdot c_w \cdot (l_b - l_v) \cdot L}{r_w \cdot (C_a + C_b + c_w \cdot L) + r_v \cdot c_w \cdot (l_b - l_a)} \quad (r_w \cdot c_v - r_v \cdot c_w) \geq 0, x_{v1} = x_{v2} = x \quad (6)$$

$$|e_b| = L' = \frac{\sqrt{2 r_w \cdot c_w \cdot (t_a - t_b) + \alpha_1^2 - 2 r_w \cdot c_w \cdot \beta - \alpha_1}}{r_w \cdot c_w} \quad (r_w \cdot c_v - r_v \cdot c_w) \geq 0, x_{v1} = x_{v2} = x = 0, \alpha_1 = r_w \cdot C_b + r_v \cdot c_w \cdot (l_b - l_v) \quad \beta = r_v \cdot \left[(l_b - l_v) \cdot C_b - (l_v - l_a) \cdot C_a + \frac{c_v}{2} \cdot ((l_b - l_v)^2 - (l_v - l_a)^2) \right] \quad (7)$$

利用这些考虑 TSV 的合并段公式, 零漂移时钟布线很容易通过现有的 2D DME 框架建立起来, 称为 DME-3D 算法. 通过这个嵌入过程, 也可以进行缓冲器的插入来提高信号传输的可靠性以及减短线长, 说得明白些, 在电容负载超过最大负载限制或者很大的绕线产生的时候插入缓冲器.

将 DME-3D 算法和前面的层嵌入算法拼接起来形成一个大的算法框架, DME-3D 算法也把每个节点处理了两次, 自底向上和自顶向下, 并且每次都是常数时间, 所以时间复杂度也是 $O(|V|)$, 所以总的时间复杂度就是 $O(|V|)$.

3 3D 抽象树生成算法

由于抽象树的拓扑结构对于 TSV 最终的数目有着巨大的影响, 因此设计一个成本效率高的 3D

抽象树拓扑生成算法是非常重要的. 现有的 3D 抽象树生成算法主要是 3D-MMM 算法^[3], 这是基于 2D MMM 算法^[7]扩展的, 它以递归的方式划分末端集合, 从一个所有末端都包括的大集合划分到每个集合只有一个末端, 每次划分都是基于 2D 平面末端坐标的几何平均值. 在划分过程中, 一旦达到某一个特定的阈值, 那么将把末端子集按所在层进行垂直方向的划分. 如果末端在不同层的分布不均匀, 那么这个算法容易导致 TSV 的分布不均匀, 从而导致线长、功耗方面的不利情况.

介绍的 3D 拓扑生成算法 NN-3D 基本上沿袭了 2D 自底向上的一个方法, NN-3D 以递归的方式选择一对子树或者末端并合并, 而选择的方法基于合并成本, 这个概念会在后面介绍. 首先, 所有子树或末端的最近邻按照合并成本计算出来, 形成最近邻图, 将最近邻图中的所有边按照合并成本递增的顺序排列起来, 再按照这个顺序来合并子树或末端, 不过如果边的两端中任意一端已经跟其他点合并了, 那么这条边就被跳过, 这个最近邻选择过程以及合并过程一直重复执行, 直到最后只剩下一个树.

合并成本由式(8)决定, 为了将线长和 TSV 同时纳入考虑, 用电容作为成本的计算对象. 另外, 为了让电容小的网络可以较早合并以降低延时, 增加了一个下游电容的权重, 也就是式(8)的最后一项, 在这个递归合并过程中, 文献[5]作了一个假设: 合并点 v 的嵌入层设为两个子节点的中间层, 也就是 $el(v)$ 的中间层. 这个假设比较粗糙, 本文进行一定的改进, 合并点所在层假设为 $el(v)$ 中离 $el(parent(v))$ 最近的那层, 如果 $el(v)$ 和 $el(parent(v))$ 有交集, 那么选择它们交集的中间层. 这个假设和原假设相比, 更接近实际情况, 能得到更准确的成本估计, 可以使最终的成本更接近最优值.

$$NN_Cost(a, b) = c_w \cdot (|e_a| + |e_b|) + c_v \cdot |l_b - l_a| + \alpha \cdot (C_a + C_b) \quad (8)$$

外循环需要运行最多 $\log |V|$ 次, 最近邻计算需要 $O(|V|^2)$, 排序需要 $(|V| \log |V|)$ 次, 内循环需要 $O(|V|)$, 那么总时间复杂度就是 $O(|V|^2 \log |V|)$.

4 实验结果

我们的算法是用 Matlab 实现的, 测试基准使用的是 IBM 的 $r1 \sim r5$, 3D 末端所在的层是随机分布在两层的, 比例是原比例, 金属线参数为: $r = 0.003$

$\Omega/\mu\text{m}$, $c = 2.0 \times 10^{-17} \text{ F}/\mu\text{m}$, 缓冲器参数为: $r_d = 122 \Omega$, $c_d = 400 \text{ fF}$, $t_d = 75 \text{ ps}$, $C_{\max} = 4 \text{ pF}$, 3D 通孔的参数为: $r_v = 0.053 \Omega$, $c_v = 27.9 \text{ fF}$.

表 2、3、4 是我们实现后的计算结果, 其中的长度单位是 μm , 时间单位是 ns, 通孔(vias)指的是单位个数.

表 2 3D-MMM 算法生成的不带缓冲器的时钟树结果

	Via Bound=1			Via Bound=10%			Via Bound=100%		
	wire	diy		wire	diy	vias	wire	diy	vias
r1	1 934 487	1.3		1 680 766	2.4	27	1 497 192	2.2	97
r2	4 148 464	3.8		3 535 557	6.2	60	2 996 564	5.9	217
r3	5 319 061	6.1		4 469 137	9.5	87	3 870 671	9.1	306
r4	10 790 865	15.6		9 146 692	28.9	191	7 785 551	26.9	716
r5	15 849 880	28.9		1 323 081	50.0	310	11 388 363	48.6	1 135

表 3 NN-3D 算法生成的不带缓冲器的时钟树结果

	Via Bound=1			Via Bound=10%			Via Bound=100%		
	wire	diy		wire	diy	vias	wire	diy	vias
r1	1 687 692	0.9		1 496 535	1.2	26	1 270 005	1.2	86
r2	3 472 492	2.7		2 940 542	3.7	53	2 550 206	3.2	199
r3	4 466 428	2.8		3 806 305	4.3	79	3 239 577	4.2	284
r4	9 170 222	9.0		7 634 801	13.2	186	6 679 686	13.3	629
r5	13 835 922	17.6		11 478 704	27.9	290	9 935 929	25.1	1 014

表 4 NN-3D 算法生成的带缓冲器的时钟树结果

	# sink	# buf	unbuffered			buffered		
			wire	diy	vias	wire	diy	vias
r1	267	15	1 496 535	1.2	26	1 586 248	1.4	26
r2	598	30	2 940 542	3.7	53	3 402 584	3.0	53
r3	862	45	3 806 305	4.3	79	4 572 659	3.1	79
r4	1 903	103	7 634 801	13.2	186	9 257 013	4.0	186
r5	3 101	122	11 478 704	27.9	290	14 098 352	6.8	290

表 2 给出的是 3D-MMM 算法在不带缓冲器的情况下得出的结果. 表 3 给出的是 NN-3D 算法在不带缓冲器的情况下得出的结果, 通孔个数限制的百分比是相对于末端的个数而言的, 可以调整式(8)中 TSV 个数的权重 c_v 来改变通孔个数限制. 通过两张表的比对, 发现 NN-3D 算法所得到的时钟布线结果在线长、延时等多方面的特性上都要明显优于 3D-MMM 算法, 综合考量下表现提升 15% 左右, 而线长、延时上的优势也就意味着成本、功耗以及信号可靠性上的优势.

表 4 给出的是带缓冲器的 NN-3D 算法所得出的结果, 其中通孔个数限制为 10%. 在加上缓冲器以后, 延时在末端数量多的情况下有明显地改善, 这是由于增加缓冲器使得负载电容下降. 线长的改变主要由于产生大的绕线的地方加了缓冲器来改善甚至防止绕线, 以及缓冲器的加入对于布线的改变.

5 结束语

文中所介绍的3D-IC时钟树综合布线算法和已有的3D-IC布线算法相比,在保证零漂移的同时,有效降低了布线的开销,提高了信号传输表现,这其中包括线长和延时的缩短以及TSV个数的缩减。

要让3DIC时钟树布线结果在各方面都有好的表现,就必须对它的每个步骤都进行最优化,抽象树生成、树节点的层嵌入以及缓冲器插入缺一不可。本文虽实现了零漂移布线,但在实际情况下时钟延时会因为温度的改变而改变,不可能让时钟漂移为零,因此还需要进一步深入研究以及优化来最小化各种情况下的时钟漂移。

参考文献:

- [1] Mondal M, Ricketts A J, Kirolos S, et al. Thermally robust clocking schemes for 3D integrated circuits [C]// Proceedings of the Conference on Design, Automation and Test in Europe. Los Alamitos, CA: IEEE, 2007: 1206-1211.
- [2] Arunachalam V, Burleson W. Low-power clock distribution in a multilayer core 3d microprocessor [C]// Proceedings of the 18th ACM Great Lakes Symposium on VLSI. New York: IEEE, 2008: 429-434.
- [3] Minz J, Zhao X, Lim S K. Buffered clock tree synthesis for 3D ics under thermal variations [C]// Proceed-

ings of the 13th Asia and South Pacific Design Automation Conference. Los Alamitos, CA: IEEE, 2008: 504-509.

- [4] Kim T-Y, Kim T. Clock tree embedding for 3d ics [C]// Proceedings of the 15th Asia and South Pacific Design Automation Conference. Los Alamitos, CA: IEEE, 2010: 486-491.
- [5] Kim T-Y, Kim T. Clock tree synthesis for TSV-based 3D IC designs [J]. Transactions on Design Automation of Electronic Systems (TODAES), 2011, 16 (4): 21-48.
- [6] Chao T-H, Hsu Y-C, Ho J-M, et al. Zero-skew clock routing with minimum wirelength. [J]. IEEE Transactions on Circuits and Systems, 1992, 39 (11), 799-814.
- [7] Jackson M A B, Srinnivasan A, Kuh E S. Clock routing for high-performance ics. [C]// Proceedings of the 27th Design Automation Conference. [s. l.]: IEEE, 1990, 573-579.

作者简介:

封成冬 男, (1989-), 硕士研究生. 研究方向为集成电路研究.

王琴 女, (1975-), 博士, 副教授. 研究方向为进行集成电路研究.

谢憬 男, (1981-), 博士研究生. 研究方向为进行集成电路研究.

毛志刚 男, (1962-), 博士, 教授. 研究方向为进行集成电路研究.

(上接第97页)

- [10] Moradi A, Poschmann A, Ling S, et al. Pushing the limits: A very compact and a threshold implementation of AES [J]. Eurocrypt 2011, 2011 (6632): 69-85.
- [11] Satoh A, Morioka S, Takano K, et al. A compact rijndael hardware architecture with S-Box optimization [J]. Asiacrypt 2001, 2001 (2248): 239-248.
- [12] P Hamalainen, Alho T, Hannikainen T, et al. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core [C]// DSD [s. l.]: IEEE, 2006: 577 - 583.

作者简介:



彭浩 男, (1987-), 硕士. 研究方向为半导体芯片设计与工艺.