一种基于二分区四分支树的高效时钟树综合方法

郭静静[1],刘润衍[1],杨君威[1],王嘉伟[1],王 冲[2*],蔡志匡[1]
(1. 南京邮电大学集成电路科学与工程学院(产教融合学院),江苏南京 210023;
2. 南京理工大学微电子学院(集成电路学院),江苏南京 210094)

摘 要: 在超大规模集成电路设计中,高效的时钟树综合对保障电路性能与可靠性至关重要. 为应对大规模电路中时钟偏差、延迟和功耗的协同优化挑战,本文提出一种基于二分区四分支类H树的高效时钟树综合方法. 该方法在自底向上阶段,首先采用贪婪聚类算法(Greedy-Based Clustering,GBC)提升底层缓冲器扇出利用率,显著减少了底层插入的缓冲器数量;随后,结合缓冲器重选位算法对局部时钟偏差进行精细控制. 在自顶向下阶段,首先通过理论推导证明沿路径均匀插入特定数量缓冲器可使时钟延迟最小化,并基于此构建了查找表以指导缓冲器的最优插入. 随后,本文将版图沿时钟源垂直划分为两个对称的半区,在每个半区内构建四分支的类H树结构. 该结构不仅应用长路径缓冲器插入算法来最小化全局时钟延迟,还利用其对称性对对称路径上的缓冲器进行合并,在保证低时钟偏差和低延迟的同时,进一步优化了缓冲器数量. 最后,针对综合过程中可能出现违反约束的情况,本文先基于布尔运算提取了缓冲器的可插入点,再根据曼哈顿矩形的性质确定了缓冲器的最优放置点. 本文算法在1×10^5–2×10^5数量触发器规模的电路进行实例验证,结果表明本算法优势显著. 相较于OpenROAD,时钟偏差与功耗分别降低32.3%和29.9%;相较于GH-Tree,时钟偏差与功耗分别降低59.9%和28.9%,同时全局时钟延迟均保持在同一水平.

关键词: 二分区四分支树;缓冲器插入;时钟偏差;时钟延迟;时钟树综合
基金项目: 江苏省自然科学基金(No.BK20240637)
中图分类号: TN47;TN401 **文献标识码:** A **文章编号:** 0372-2112(2025)08-2719-10
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250076

An Efficient Clock Tree Synthesis Method Based on Bi-Partition Four-Branch Tree

GUO Jing-jing[1], LIU Run-kan[1], YANG Jun-wei[1], WANG Jia-wei[1], WANG Chong[2*], CAI Zhi-kuang[1]
(1. *College of Integrated Circuit Science and Engineering* (*College of Industry-Education Integration*), *Nanjing University of Posts and Telecommunications*, *Nanjing*, *Jiangsu* 210023, *China*;
2. *School of Microelectronics* (*School of Integrated Circuits*), *Nanjing University of Science and Technology*, *Nanjing*, *Jiangsu* 210094, *China*)

Abstract: In very large scale integration (VLSI) design, efficient clock tree synthesis (CTS) is crucial for ensuring circuit performance and reliability. To address the co-optimization challenge of clock skew, latency, and power consumption in large-scale circuits, this paper proposes an efficient CTS method based on a bi-partition, four-branch H-like tree. In the bottom-up phase, the method first employs a greedy-based clustering (GBC) algorithm to enhance the fanout utilization of low-level buffers, significantly reducing the number of inserted buffers. Subsequently, it incorporates a buffer re-placement algorithm for the fine-grained control of local clock skew. During the top-down phase, it is first theoretically proven that uniformly inserting a specific number of buffers along a path minimizes clock latency, and a look-up table is constructed based on this principle to guide optimal buffer insertion. Next, the layout is vertically divided into two symmetrical half-regions from the clock source, and a four-branch H-like tree structure is constructed within each half-region. This structure not only applies the long-path buffer insertion algorithm to minimize global clock latency but also leverages its symmetry to merge buffers on symmetrical paths, further optimizing the buffer count while ensuring low clock skew and latency. Finally, to handle potential constraint violations during synthesis, the method first extracts insertable locations for buffers based on Boolean operations and then determines their optimal placement according to the properties of Manhattan rectangles. The

收稿日期:2025-01-22;录用日期:2025-08-20;责任编辑:朱梅玉
*通讯作者:王冲

proposed algorithm is validated on circuit instances with 1×10^5 to 2×10^5 flip-flops, and the results demonstrate its significant advantages. Compared to OpenROAD, our method reduces clock skew and power consumption by 32.3% and 29.9%, respectively. In comparison with GH-Tree, it achieves reductions of 59.9% in clock skew and 28.9% in power consumption, while maintaining a comparable global clock latency.

Key words: bi-partition four-branch tree; buffer insertion; clock skew; clock latency; clock tree synthesis
Foundation Item(s): Natural Science Foundation of Jiangsu Province (No.BK20240637)

1 引言

随着超大规模集成电路先进工艺技术的发展,芯片上集成的门电路数量快速增长[1],对高性能、快速的时钟树综合方法提出了更高的要求[2],其关键在于最小化时钟偏差和时钟延迟的同时,减小优化所需要的缓冲器数量[3],以降低芯片整体的功耗.

大规模电路的时钟树综合通常可以分为自底向上的设计方法、自顶向下的设计方法与自底向上和自顶向下相结合的设计方法. 在采用自底向上的设计方法时[4],由于缺乏对上游结构的了解,实现负载分布的均衡和时钟树的构建变得颇具挑战性. 而自顶向下的方法则受限于对下游结构的不确定性,特别是在平衡各分支的负载能力时遭遇难题. 现如今的大规模电路时钟树综合,多采用自底向上和自顶向下相结合的方式,稳定地平衡各分支的时钟延迟与偏移[5]. 然而,这种混合设计方法只是提供了宏观框架,其最终性能表现高度依赖于所选择的具体时钟树拓扑结构.

在宏观设计框架的基础上,针对时钟树综合不同指标的高性能需求,衍生出了多种拓扑结构. 其中包括非树形结构和树形结构[6]. 非树形的典型结构是网格型时钟拓扑[7],网格型时钟拓扑结构因其高对称性和高密度,可以稳定地传输时钟信号. 然而,构建网格型时钟拓扑需要大量缓冲器和布线资源,引起较多的资源浪费. 树形结构常见的有H树、X树和鱼骨树等,树形结构因其灵活和低功耗的特点,仍为VLSI(Very Large Scale Integration)时钟综合的主流. H树结构能够达到理论上的零偏差时钟树,通过等距离四分支的扇出,迭代地构建子树,以达到从源点到节点的路径长度相同. 然而,为适应不均匀分布的时钟单元分布,H树需要不断增加深度,以逼近零偏差,否则构建的时钟树是有界偏差的[8]. 在增加深度的过程中,构建H树所需的缓冲器数量将呈指数增长. 同时,路径上缓冲器数量的增多,会导致路径时钟延迟以缓冲器本征延迟为特征近似线性增长,使时钟树综合优化的关键指标严重恶化. X树也是一种实现等长互连线的方法,与H树相比,它采用了很多非直角的互连线. 由于实际布线过程中大多采用直角互连线,因此X树并不常见. 鱼骨树结构因其灵活的路径分配方式[7],在低时钟延迟为目标的时钟树综合中应用较多. 但由于其对时钟偏差的控制效果较差,并不能应用于时序敏感的电路中.

当前的时钟树综合已经愈发朝着“低偏差、低延迟、低资源”[9]的设计方向发展,而不是仅仅满足于单一指标的优化. 为了打破单一拓扑的局限性,现代时钟树综合算法在优化策略上进行了更深入的探索. 文献[10]提出了使用推迟合并嵌入算法(Deferred-Merge Embedding,DME)来构造时钟树. 先通过自底向上的合并方式得到可能的触发器合并点位置. 确定根节点位置后,自顶向下地选择合并点的精确位置. 该算法实现了严格的零时钟偏差,但是在更大规模的电路上会导致资源需求量大量增加,难以应用于大规模设计. 作为改进,文献[8]在DME算法基础上提出了有界时钟偏差的时钟树构造算法,通过扩展合并区域来进一步减少时钟树的总线长. 该方法适用于时钟偏差较不敏感的电路,但时钟树的延迟仍旧很大. 这些方法实现的时钟树的结构本质上是以二叉树的结构为基础的,其固有的分支限制和大量的缓冲器的插入数量会引起布线资源紧张的问题. 为解决这一问题,研究者开始转向分支更为灵活的多叉树结构,以实现低时钟偏差和低时钟延迟目标下设计资源的最小化. 文献[11]提出了对称多叉树结构,该结构在构造时,首先对每层的分支数进行分配,保证构造时钟树时每层互连线的线长相等,每层的缓冲器数量相等,缓冲器尺寸相等. 文献[12]提出以最小时钟延迟和时钟偏差为目标对缓冲器进行插入,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,通过调整缓冲器位置和层数分别实现了低时钟延迟和低时钟偏差,对时钟偏差控制的下整体时钟延迟优化有重要意义. 文献[13]则提出了一种基于灵活H树(Flexible H-tree)的时钟树综合方案,以应对版图中触发器分布不均匀的情况. 文献[14]提出了一种具有任意分支因子的广义H树(GH-Tree)拓扑,并以线性规划的方法对缓冲器的放置进行了优化. 此外,该工作还提出了一种平衡的K-means聚类方法和LP指导的缓冲区放置方法,以便根据给定的汇点放置来嵌入GH树. 而文献[15]则提出了一个名为iCTS的迭代式、分层级综合框架,将多目标优化推向了新的高度. 他们定义了“偏斜度”指标,旨在从拓扑构建层面就同时兼顾三个目标. 其提出的CBS(Conflict-Based Search)建树算法和一系列缓冲器优化技术构成了完整的协同

优化流程. 这些方法在平衡各项指标上取得了显著进步,实现了与商用时钟综合工具相近的时钟偏差和时钟延迟. 然而,这些方法或是在底层聚类时对缓冲器扇出利用率考虑不足,或是缺乏对局部偏差的精细控制,导致全局优化的空间受限. 因此,如何在设计初期就为全局优化奠定良好基础,通过高效的底层聚类和局部偏差控制,最终在多分支拓扑上实现三大指标的更优平衡,是当前时钟树综合领域亟待解决的关键问题.

总体来看,更为灵活的树形结构和基于缓冲器插入的整体时钟延迟优化算法逐渐占据主流[16]. 为了满足大规模集成电路低时钟偏差、低时钟延迟和低缓冲器数量的要求,本文提出了一种基于二分区四分支类H树拓扑的高效时钟树综合方法,递归地构建二分区四分支树控制时钟偏差,并结合缓冲器插入和合并策略在减小时钟延迟的同时实现缓冲器数量最小化. 该方法有以下创新点:(1)底层采用GBC(Greedy-Based Clustering)贪婪聚类算法,提高了缓冲器扇出利用率,减少了底层缓冲器数量;(2)基于低时钟偏差的缓冲器重选位算法在底层严格控制了本地时钟偏差,从而为上层综合提供更大的时钟延迟优化空间;(3)二分区四分支类H树拓扑结构结合时钟延迟优化缓冲器的插入和合并策略,实现了低时钟偏差的控制和全局时钟延迟的优化.

2　延迟模型

在信号延迟分析中,Elmore延迟模型[17]因其简单性和高效性而被广泛应用于RC网络的延迟近似计算. 传统的Elmore延迟公式会考虑下游电容 C_v 的累积效应,以反映子树负载对路径延迟的影响. 从源点 s_0 到接收点 s_i 的延迟 t_{ED} 可近似表示为

$$t_{\mathrm{ED}}(s_0,s_i)=\sum_{e_v\in \mathrm{path}(s_0,s_i)} R_{e_v}\left(\frac{1}{2}\cdot C_{e_v}+C_v\right) \tag{1}$$

其中,e_v 是源点 s_0 到接收点 s_i 的互连线,R_{e_v} 和 C_{e_v} 分别是指互连线 e_v 上的电阻和电容.

本文的走线均采取曼哈顿走线,不考虑蛇形走线的情况,将互连线的电阻 R_{e_v} 和电容 C_{e_v} 表示为从源点 s_0 到接收点 s_i 的曼哈顿距离 D 的函数,得到互连线延迟的函数. 其中,r、c 分别为单位线长电阻、电容.

$$R_{e_v}=rD \tag{2}$$

$$C_{e_v}=cD \tag{3}$$

$$t_{\mathrm{ED}}(s_0,s_i)=\sum_{e_v\in \mathrm{path}(s_0,s_i)}\left(\frac{1}{2}r\cdot c\cdot D^2+r\cdot D\cdot C_v\right) \tag{4}$$

相较于精确地考虑时延模型下的计算问题,本文更注重时钟树拓扑的生成和缓冲器的插入方法,因此在本文中重点关注集总模型的互连线延时和缓冲器本征延时,将延迟 t_{ED} 表示如下:

$$t_{\mathrm{ED}}(s_0,s_i)\approx\sum_{e_v\in \mathrm{path}(s_0,s_i)} 0.69\cdot\frac{1}{2}r\cdot c\cdot D^2 \tag{5}$$

3　二分区四分支类H树构建

时钟树综合自底向上与自顶向下相结合的方法对于时钟树的关键指标优化效果显著,本文提出了一种基于二分区四分支树的高效时钟树综合方法,整体流程如图1所示.

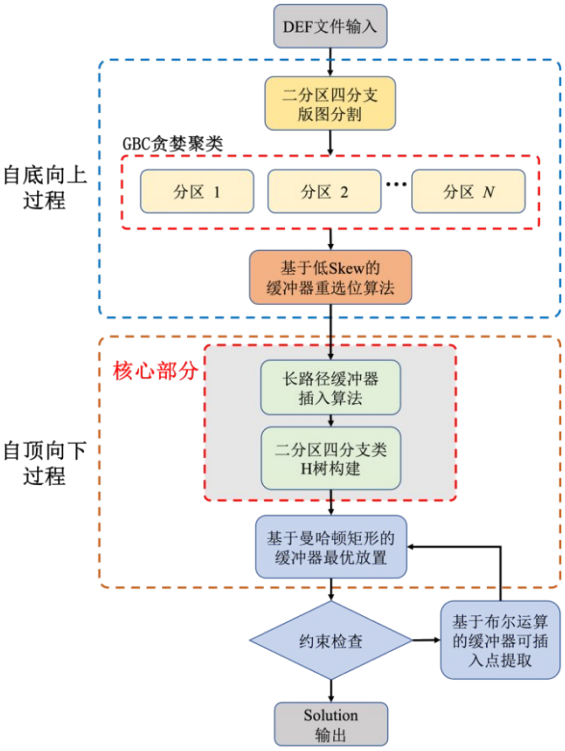


图1　整体流程图

以下着重介绍自底向上过程和自顶向下过程中采取的综合方法.

3.1　自底向上过程

时钟树自底向上过程一般先通过聚类算法形成触发器集群,再构建底层拓扑结构. 常见的触发器聚类算法包括Kmeans算法、二分Kmeans算法(Bi-Kmeans)和K分法(K-splitting)等. 这类算法通常需要指定初始聚类个数,再对形成的聚类结果进行约束合法化的调整. 本文提出了基于贪婪策略的GBC聚类算法,以适应大规模电路的底层聚类. 针对聚类后形成的触发器集群,合理选取集群内缓冲器的位置对于本地时钟偏移降低有重要作用,因此本文提出在聚类后采用重选位算法确定缓冲器的位置.

3.1.1　GBC贪婪聚类

针对大规模电路,合并触发器形成集群显得尤为

重要[18],直接影响了上层时钟树的构建规模. 事实上,底层插入缓冲器的数量取决于缓冲器扇出的利用率. 较高的扇出利用率可以减少连接触发器所需要的缓冲器数量,将原本复杂度较高的时钟树综合问题的规模缩小至缓冲器的数量,因此该问题可以描述为最小化缓冲器数量$\mathrm{Buf}_{\mathrm{count}}$的优化问题:

$$\min \mathrm{Buf}_{\mathrm{count}} \tag{6}$$

$$\text{s.t.}\begin{cases} \mathrm{rc}_{C_j} \leqslant \mathrm{rc}_{\max} \\ \left|C_j\right| \leqslant \mathrm{fanout}_{\max} \\ \sum_{j=1}^{m}\left|C_j\right| = \left|S_{\mathrm{ff}}\right| \end{cases} \tag{7}$$

其中,S_{ff}为触发器的集合,C_j为聚类后形成的触发器集群,$\mathrm{rc}_{\max}$为负载约束,$\mathrm{fanout}_{\max}$为扇出约束.

为了尽可能地提高缓冲器扇出的利用率,本文提出在满足负载约束和扇出约束的条件下,用贪婪思想迭代选取触发器形成聚类,并在聚类中心插入缓冲器连接聚类内触发器. 其中聚类中心的公式由下式给出:

$$x_{c_i} = \left(\sum_{i=1}^{m} x_{f_i}\right) \Big/ m \tag{8}$$

$$y_{c_i} = \left(\sum_{i=1}^{m} y_{f_i}\right) \Big/ m \tag{9}$$

GBC贪婪聚类算法的具体实现如算法1所示.

第一步,将已知的触发器集合S_{ff}根据其位置(x_{f_i},y_{f_i})构建二维KD树,通过初始网格分割参数$2N\times N$将版图区域矩形均匀划分,并分配触发器至各个网格G_i. 第二步,从第一个网格G_1开始,新建一个空聚类c_1,将该网格中距离网格中心最远处的触发器$f_{\max}$加入聚类,将聚类中心设置为触发器$f_{\max}$的坐标. 第三步,迭代地从聚类中心开始搜索邻近的触发器尝试加入集群. 若加入新触发器后聚类仍满足负载和扇出约束,则接收该解并通过式(8)和式(9)更新聚类中心;否则,创建新聚类,并将该触发器加入新聚类中. 重复上述第二步和第三步,即可获得最终的聚类结果.

3.1.2 基于低时钟偏移的缓冲器重选位算法

初始触发器集群形成后,在该集群的中心插入缓冲器,连接集群内的所有触发器. 由于插入的缓冲器位置处于触发器集群的中心,底层的本地时钟偏差产生于不同集群距离网格中心的距离不同,因此本文提出了如图2所示的缓冲器重选位算法. 算法的核心是远近搭配原则,距离网格中心点较远的触发器集群应该倾向于连接距离网格中心点较近的缓冲器,反之亦然. 其中黄色圆圈代表网格中心缓冲器,橙色方框代表底层缓冲器,灰色圆圈代表触发器,d_1是指网格中心缓冲器至本地参考延迟基准底层缓冲器的曼哈顿距离,$d_{1,i}$是指网格中心缓冲器至其他底层缓冲器的曼哈顿距离,d_2是指底层缓冲器至本地参考延迟基准触发器的曼哈顿距离,$d_{2,j}$是指底层缓冲器至其他触发器的曼哈顿距离.

算法1 GBC贪婪聚类算法

输入:节点集合$S_{\mathrm{ff}}=\{f_1,f_2,\cdots,f_n\}$

节点位置集合$\left\{\left(x_{f_1},y_{f_1}\right),\left(x_{f_2},y_{f_2}\right),\cdots,\left(x_{f_n},y_{f_n}\right)\right\}$

输出:聚类集合$S_{\mathrm{cluster}}=\{c_1,c_2,\cdots,c_m\}$

聚类中心集合$\left\{\left(x_{c_1},y_{c_1}\right),\left(x_{c_2},y_{c_2}\right),\cdots,\left(x_{c_n},y_{c_n}\right)\right\}$

1. 计算所有节点到指定网格中心点的距离:

$$d_i = \left\|\left(x_f,y_f\right)-\left(x_{\mathrm{center}},y_{\mathrm{center}}\right)\right\|, \forall f_i \in S_{\mathrm{ff}}$$

2. 为节点位置集合$\left\{\left(x_{f_1},y_{f_1}\right),\left(x_{f_2},y_{f_2}\right),\cdots,\left(x_{f_n},y_{f_n}\right)\right\}$构建KD树;
3. 初始化未聚类节点集合$U=S_{\mathrm{ff}}$;
4. 当$U\neq\varnothing$时执行:
5. 选择具有最大距离d_i的节点$f_{\max}$,并从U中移除;
6. 初始化聚类$C=\{f_{\max}\}$,聚类C的中心坐标$\left(x_C,y_C\right)=F_{\mathrm{center}}(C)$;
7. 将C添加到聚类集合S_{cluster}中;
8. 当$U\neq\varnothing$时执行:
9. 使用KD树找到离$\left(x_C,y_C\right)$最近的节点f_{nearest};
10. 如果$\mathrm{RC}_C+\Delta\mathrm{RC}\left(f_{\mathrm{nearest}}\right)\leqslant\max_{\mathrm{rc}}$且$|C|+1\leqslant\mathrm{fanout}_{\max}$,则:
11. 将f_{nearest}添加到C中;
12. 更新$\left(x_C,y_C\right)$和RC_C;
13. 从U中移除f_{nearest};
14. 否则: 跳出循环;
15. 返回S_{cluster}和$\left\{\left(x_{c_1},y_{c_1}\right),\left(x_{c_2},y_{c_2}\right),\cdots,\left(x_{c_n},y_{c_n}\right)\right\}$

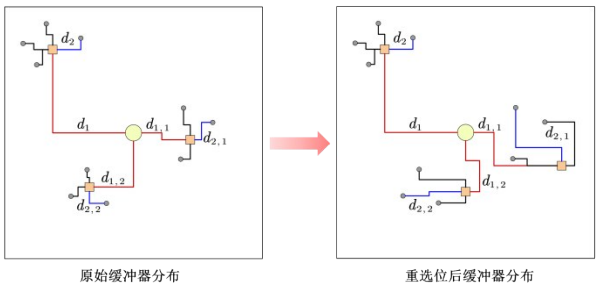


图2 缓冲器重选位示意图

该算法包括两个主要步骤. 步骤1:确定单元网格内的触发器本地延迟基准值. 基于单元网格内缓冲器和触发器的分布与连接关系,可以找到网格内时钟延迟最长的路径. 由于触发器的位置固定,要实现本地的有界时钟偏差甚至零偏差,必须调整网格内连接触发器路径上的时钟延迟与距离网格中心最远处触发器的延迟相近. 步骤2:单元网格内的触发器延迟基准值确定后,需要对其他偏离基准值较大的触发器集群进

行调整,即对集群内缓冲器进行重选位,使得每一路径下的时钟延迟接近,从而减小本地时钟偏移.

记网格中心缓冲器与触发器集群中心缓冲器的距离为d_i,集群中心缓冲器与集群中触发器的距离为$d_{i,j}$,基准值为d_{st_1}和d_{st_2}. 那么该问题可以表示为一个优化模型:

$$\min \Delta_{i,j}=\left(d_{\mathrm{st}_1}^2+d_{\mathrm{st}_2}^2\right)-\left(d_i^2+d_{i,j}^2\right) \tag{10}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}\dfrac{1}{2}r\cdot c\cdot d_i^2\leqslant \mathrm{rc}_{\max}\\ \sum\limits_{j=1}\dfrac{1}{2}r\cdot c\cdot d_{i,j}^2\leqslant \mathrm{rc}_{\max}\end{cases} \tag{11}$$

上式约束条件表明,在缓冲器重选位的过程中,每个缓冲器的负载仍应该满足最大负载的约束. 由于时钟树在上层构建时采用了零偏差的类H树结构,上层不会产生时钟偏差,时钟偏差的产生全都来源于底层,因此通过缓冲器的重选位,进一步降低了底层时钟偏差的产生.

3.2　自顶向下过程

自顶向下综合过程实际上是拓扑的生成,上层拓扑对于时钟偏差的影响显著,长路径时钟延迟若不平衡,会使下游触发器时钟偏差剧增. H树结构理论上能实现零偏差时钟树,但为适应不均匀分布的时钟单元,H树需增加深度逼近零偏差,这会使构建时钟树所需的缓冲器数量指数增长,严重恶化时钟树的关键指标. 总结前人的工作后,发现在源节点到叶节点的路径上插入缓冲器,能够降低该路径的延迟. 本文基于此缓冲器插入优化时钟延迟的方法,提出了二分区四分支类H树的拓扑结构,以应对大规模电路的时钟树综合问题.

3.2.1　长路径缓冲器插入算法

在已知长路径下插入缓冲器能够实现路径延迟减少的基础上,本文对长路径中缓冲器的插入位置和个数进行推导,产生了长路径缓冲器插入的查找表,应用于路径时钟延迟优化中.

定理1　缓冲器最优位置定理长路径下均匀插入缓冲器能够获得插入缓冲器后当前路径下延迟最小值.

证明　不失一般性,考虑一条连接两个缓冲器的路径,在该路径上插入一个缓冲器. 如图3所示,缓冲器A和缓冲器B为路径的起点和终点,在两者之间插入缓冲器C.

缓冲器A与缓冲器B的距离为$d_{A,B}$,缓冲器A与缓冲器C的距离为$d_{A,C}$,缓冲器B与缓冲器C的距离为$d_{B,C}$,那么缓冲器A到B的路径延迟$\mathrm{latency}_{A,B}$可以表示为

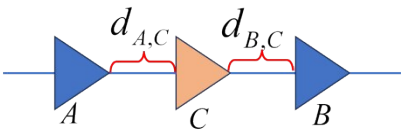


图3　缓冲器插入示意图

$$k=0.69\cdot\frac{1}{2}r\cdot c \tag{12}$$

$$\mathrm{latency}_{A,B}=k\cdot d_{A,C}^2+k\cdot d_{B,C}^2+\mathrm{delay}_{\mathrm{buf}} \tag{13}$$

其中,缓冲器A和缓冲器B位置固定. 因此引入比例因子p,将路径时钟延迟转化为关于p的单变量函数:

$$\mathrm{latency}_{A,B}=k\cdot\left[p^2+(1-p)^2\right]\cdot d_{A,B}^2+\mathrm{delay}_{\mathrm{buf}} \tag{14}$$

显然,在$p=1/2$即均匀插入缓冲器时,$\mathrm{latency}_{A,B}$取得最小值. 该结论可推广至单一路径下插入多个缓冲器的情况,下面给出证明.

给定一系列比例系数p_i,需要求得p_i平方和的最小值.

$$\min\sum_i^n p_i^2 \tag{15}$$

$$\text{s.t.}\sum_i^n p_i=1 \tag{16}$$

已知柯西-施瓦茨不等式的一般形式为

$$\left(\sum_{i=1}^n a_ib_i\right)^2\leqslant\left(\sum_{i=1}^n a_i^2\right)\left(\sum_{i=1}^n b_i^2\right) \tag{17}$$

令$a_i=1$和$b_i=p_i$,则可得到:

$$\left(\sum_{i=1}^n p_i\right)^2\leqslant n\cdot\sum_{i=1}^n p_i^2 \tag{18}$$

由于$\sum\limits_{i=1}^n p_i=1$,上述不等式关系为

$$\sum_{i=1}^n p_i^2\geqslant\frac{1}{n} \tag{19}$$

不等式取等条件为$p_i=1/n$. 上述推导得到了插入固定数量的缓冲器后,路径延迟最小的缓冲器位置分布,即在路径上均匀地插入缓冲器.

实际时钟树构建过程中,由于缓冲器存在本征延时,插入数量过多的缓冲器反而会导致路径延迟的增加,因此下文基于定理一的缓冲器插入位置策略,对不同长度路径下缓冲器插入的个数进行推导.

令不插入缓冲器下某一路径延时为$\mathrm{latency}_1$,均匀插入n个缓冲器后该路径延时为$\mathrm{latency}_2$:

$$\mathrm{latency}_1=k\cdot D^2 \tag{20}$$

$$\mathrm{latency}_2=k\cdot\frac{D^2}{n+1}+n\cdot\mathrm{delay}_{\mathrm{buf}} \tag{21}$$

若要满足插入缓冲器后的时钟延迟相较于原始路

径时钟延迟要小，即满足 $\text{latency}_2 < \text{latency}_1$，则存在以下关系式：

$$k \cdot \frac{D^2}{n+1} + n \cdot \text{delay}_{\text{buf}} \leqslant k \cdot D^2 \tag{22}$$

解得某一曼哈顿距离 D 下最佳插入个数 n_{best}：

$$\begin{cases} n_{\text{best}} = D \cdot \sqrt{\dfrac{k}{\text{delay}_{\text{buf}}}} - 1 \\ D \geqslant \sqrt{\dfrac{(n+1)\text{delay}_{\text{buf}}}{k}} \end{cases} \tag{23}$$

由于插入个数变量 n 是离散的，在这种情况下，为了能够准确且高效地确定当前路径下应当插入缓冲器的数量，一种可行且有效的方法是采用查找表的方式．

图4展示了查找表方法确定路径最优缓冲器插入个数所对应的函数图像，横坐标为路径曼哈顿长度 D，纵坐标为路径最优缓冲器插入个数 n_{best}．函数呈现阶梯状，且每一个曼哈顿长度均对应唯一的缓冲器插入个数．

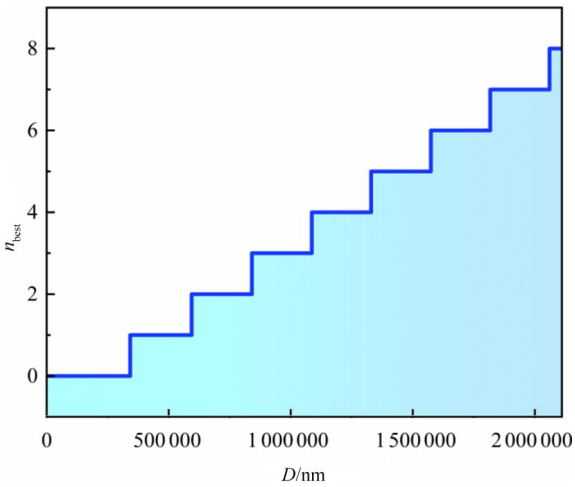


图4 缓冲器插入查找表函数

3.2.2 二分区四分支类H树的构建

基于上一小节提出的长路径缓冲器插入算法，本小节将缓冲器插入作为构建类H树中的时钟延迟优化过程，构建出H树结构的变体，以在控制低时钟偏差的同时降低全局时钟延迟．

图5展示了时钟延迟优化缓冲器的插入算法和缓冲器合并的过程．首先，对于初始的缓冲器插入，自顶向下找到长路径并在路径中点插入缓冲器以降低延迟．如图5中缓冲器插入过程所示，从中心线缓冲器连接上下半区两个缓冲器会产生两个路径，在这两个路径中点插入缓冲器可以降低两个路径的延迟．由此可以得到一个重要结论：该方法下插入的缓冲器数量等于路径的数量及其相应的倍数．

为减少插入的缓冲器数量，可以合并相似路径，使

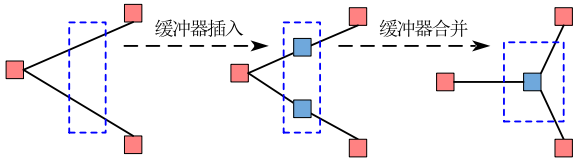


图5 缓冲器插入和合并示意图

一个缓冲器匹配多条路径．图5缓冲器合并过程展示了该过程：在满足均匀插入的原则下，插入一个缓冲器同时连接上下半区的缓冲器，实现两条路径延迟同时减少的同时，减少了插入的缓冲器的数量．

基于此缓冲器插入与合并方法，本文构建了如图6结构的二分区四分支类H树，定义如下：

(1)二分区：将版图区域按时钟源坐标分为上下对称的两个半区．

(2)四分支：每个半区内自顶向下构建时钟树，其中关键节点的缓冲器扇出四个分支连接下一层缓冲器，逐级往下迭代．

与H树不同之处在于，该拓扑结合了长路径缓冲器的插入算法．在保证均匀插入缓冲器的前提下，合并了对称路径的缓冲器，减少了构建类H树所需要的缓冲器数量．

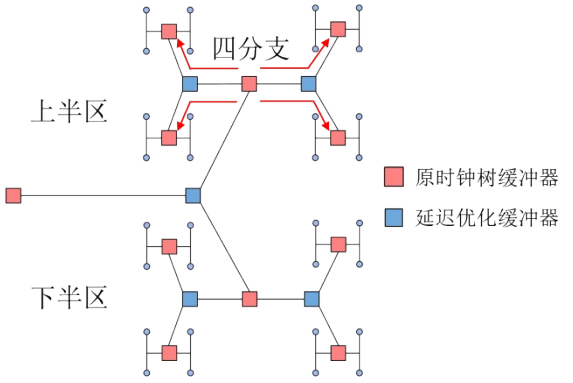


图6 二分区四分支类H树示意图

综上所述，自顶向下构建二分区四分支类H树的过程可以描述为四个步骤．步骤1：对版图区域进行分区，分为上下半区．步骤2：构建初始类H树，连接所有自底向上综合过程产生的缓冲器．步骤3：自顶向下地查找长路径，对照查找表，均匀地插入对应数量的缓冲器．步骤4：合并对称路径上的缓冲器．至此，二分区四分支类H树构建完成，实现了低时钟偏移控制下，全局时钟延迟和缓冲器数量的优化．

3.2.3 重叠约束下缓冲器位置最优化

在前文提到的综合过程中，选定的缓冲器位置满足了最大负载约束和最大扇出约束，但并未对可能产生的重叠情况进行合法化．考虑到合法化的过程可能会对现有的时钟偏差和时钟延迟造成影响[19]，本文提

出了一种基于几何布尔运算的缓冲器位置调整策略,以减少合法化过程对综合指标的负面影响.

合法化过程可以分为两个主要过程:缓冲器可插入点提取和缓冲器位置最优化. 如图7所示,在缓冲器可插入点提取中首先对某一片版图区域作几何布尔运算,得到空闲区域,膨胀该空闲区域,之后用缓冲器大小的网格对区域进行分割,得到缓冲器的可插入点. 由曼哈顿矩形的性质可得,缓冲器在图7中两个曼哈顿矩形的交线上移动时,能保证中间的缓冲器到两端缓冲器的曼哈顿距离不变. 基于此调整策略,在合法化缓冲器的过程中,可以实现调整路径上的时钟延迟不变,不会进一步引起时钟偏差的增大.

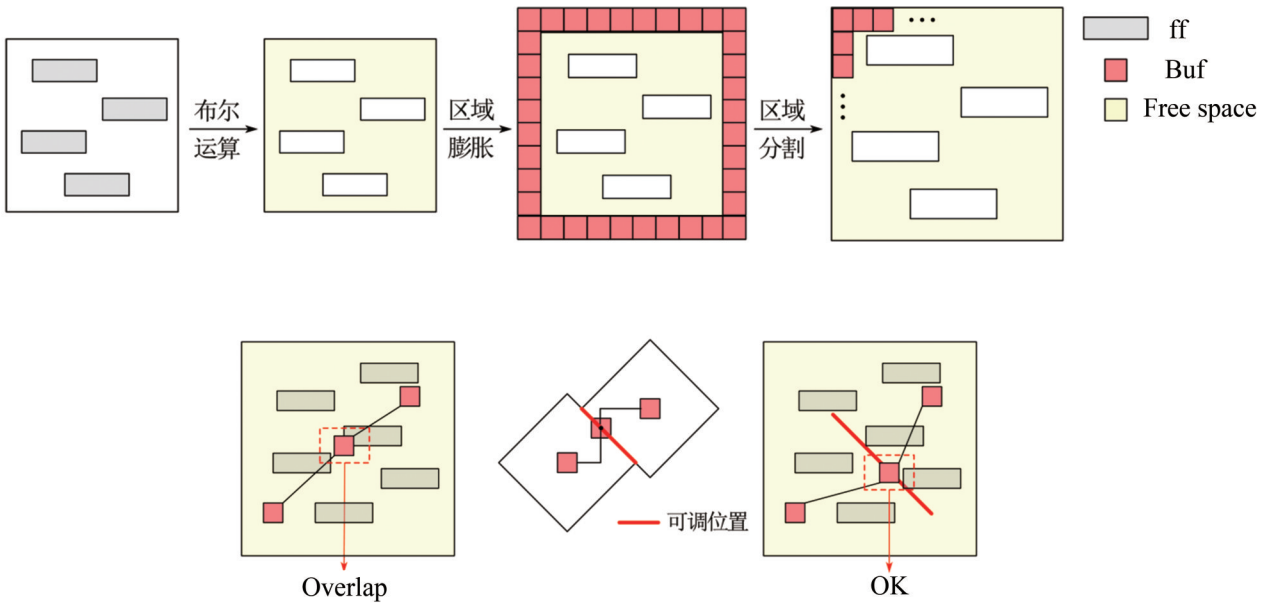


图7　缓冲器位置合法化示意图

4　实验结果

本文提出的算法用Python语言在AMD RYZEN 5000 SERIES,主频为3.2 GHz的CPU(Central Processing Unit)上实现,对芯行纪公司于2024年EDA精英挑战赛公开的十个电路实例进行测试. 十个电路实例均包含十万数量级不均匀分布的触发器电路,实例1数量最少,包含121 949个触发器;实例5数量最多,包含190 600个触发器. 图8给出了实例1的版图,可见在触发器数量最少的实例上触发器已经占据了版图的绝大部分区域,因此本文采用的十个实例对时钟树综合提出了极高要求,适合用于对本文提出的时钟树综合算法进行测试.

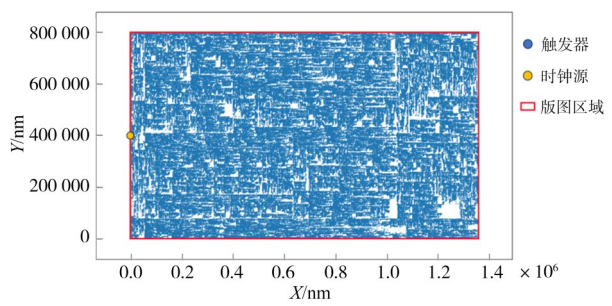


图8　测试案例1版图

本文主要分两个角度进行测试. 首先,对比了GBC贪婪算法、KSR(K-splitting)算法[20]和Balanced-Kmeans算法[15]对于时钟偏差和缓冲器数量的影响. 其次,分析了缓冲器重选位算法对于时钟偏差的影响. 最后,基于本文提出的二分区四分支类H树,与广泛使用的开源工具OpenROAD[21]和GH-Tree[14]拓扑进行比较,并分析了如何在控制低时钟偏差的基础上降低全局时钟延迟和缓冲器数量.

表1给出了实验用到的工艺参数,时钟树综合环境将基于给出的工艺参数进行测试.

表1　工艺参数

参数	值
单位长度电阻/(Ω/μm)	0.008 545 72
单位长度电容/(pF/μm)	$1.476\,24\times10^{-7}$
缓冲器最大负载/ps	446.569
缓冲器最大扇出	65
缓冲器本征延时/ps	25.690 1

工艺参数确定后,对测试案例采用本文算法进行时钟树综合. 表2给出了十个测试案例中GBC贪婪算法和先前工作采用的聚类算法的对比结果. 相较于KSR算法和Balanced-Kmeans算法,本文提出的GBC贪

婪聚类算法在满足负载约束和扇出约束的条件下，缓冲器数量较KSR算法平均减少了30.04%，较Balanced-Kmeans算法平均减少了29.14%.

进一步观察算法在不同规模电路上的表现时，可以发现在本文的测试案例中，随着触发器数量和版图复杂度的增加，本文方法在缓冲器数量上的优势相较于KSR和Balanced-Kmeans算法进一步扩大了. 在触发器数量最少的案例1中，GBC相较KSR和Balanced-Kmeans的缓冲器数量分别减少了34.2%和24.6%，而在触发器数量最多的案例5上优化程度扩大到了44.3%和29.8%. GBC算法的贪婪策略不仅没有因为电路规模和触发器分布复杂度的增加而失效，反而实现了更好的性能指标，展现了良好的鲁棒性.

表3给出了十个测试案例时钟偏差、时钟延迟和缓冲器数量的指标，并和相对应指标的下界进行了对比. 表3中每一个时钟树关键指标的最左列给出了相对应的参考数值. 其中，重选位前时钟偏差代表初始触发器集群形成后的底层时钟偏差；时钟延迟下界代表距离时钟源最远处触发器在缓冲器插入优化下的最小时钟延迟；缓冲器数量下界代表全部触发器均与时钟源形成路径时最少需要的缓冲器数量.

对于时钟树综合问题来说，同时实现三个指标达到最优解是不可能实现的，因此需要针对不同使用情况选取合适的策略，实现关键指标的优化. 表3中给出的延迟下界和数量下界是单一指标最优化的程度，实际求解时并不能严格达到同时的最优化. 对于时钟偏差指标而言，缓冲器层数越多，时钟偏差越小，存在的理论下界为零偏差. 但激进地增加缓冲器层数以逼近零偏差，会导致时钟延迟和缓冲器数量指数上涨，浪费大量资源，导致时钟树性能退化.

表2 三种聚类算法对比

测试案例	底层缓冲器数量			
	数量下界	KSR	Balanced-Kmeans	GBC
case1	1 876	2 846	2 813	2 121
case2	1 758	2 796	2 782	1 824
case3	2 144	3 365	3 343	2 390
case4	2 120	3 245	3 304	2 367
case5	2 834	4 439	4 378	3 075
case6	2 752	4 286	4 244	2 991
case7	2 222	3 512	3 467	2 469
case8	2 134	3 383	3 332	2 372
case9	2 555	4 095	3 942	2 797
case10	2 846	4 577	4 420	3 115
平均	2 324	3 554	3 602	2 552

本文的时钟树综合算法，在综合考量三个指标的优化下，选择基于低时钟偏差前提的时钟延迟和缓冲器数量优化策略. 本质上是通过在类H树构建过程中增加了一层缓冲器，用部分时钟延迟换取了更低的时钟偏差. 同理，通过上层缓冲器类H树结构的构建和长路径延迟优化的缓冲器插入，结合缓冲器合并算法，实现了时钟偏差、时钟延迟和缓冲器数量的协同优化.

表3 本文时钟树综合算法与OpenROAD和GH-Tree算法的比较

测试案例	时钟偏差/ps				时钟延迟/ps				功耗/mW		
	重选位前	Open ROAD	GH-Tree	本文工作	延迟下界	Open ROAD	GH-Tree	本文工作	Open ROAD	GH-Tree	本文工作
case1	7.927	10.526	17.568	6.302	431.436	445.692	434.854	460.671	57.530	56.870	43.030
case2	8.037	9.854	15.694	5.635	407.703	416.554	412.633	439.476	56.491	56.211	37.051
case3	6.784	9.495	15.987	5.753	385.898	402.856	394.549	416.153	67.997	67.557	48.497
case4	10.306	10.532	18.548	8.016	476.233	494.561	484.561	504.305	65.589	66.769	48.029
case5	10.308	12.635	19.652	7.427	512.687	521.883	521.413	538.717	89.701	88.481	62.421
case6	10.680	11.659	19.660	8.691	469.936	476.291	480.002	498.837	86.614	85.774	60.714
case7	7.586	10.285	15.922	7.090	402.401	419.542	411.024	434.716	70.962	70.062	50.102
case8	8.339	9.481	16.328	6.421	422.251	441.227	429.669	452.558	68.354	67.334	48.134
case9	7.890	9.547	16.199	6.266	439.253	451.964	448.214	470.454	82.730	79.670	56.770
case10	11.679	9.548	19.112	8.445	558.194	569.695	564.816	582.198	92.465	89.325	63.225
平均	8.954	10.356	17.467	7.005	450.599	464.027	458.174	479.809	73.843	72.805	51.797

表3显示，本文提出的时钟树综合方法的关键指标为时钟偏差，在十个测试案例中均小于10 ps. 在时钟偏差方面，本文的解决方案在对底层缓冲器进行重选位后，显著降低了时钟偏差，与OpenROAD相比减少了32.3%，与GH-Tree相比减少了59.9%；在时钟延迟方面，本文的解决方案基本与OpenROAD和GH-Tree持平；在缓冲器数量方面，本文的解决方案与OpenROAD相比减少了30.2%，与GH-Tree相比减少了29.2%. 在

功耗方面,本文的解决方案与OpenROAD相比平均减少了29.9%,与GH-Tree相比平均减少了28.9%.

相较于先前工作,本文的时钟偏差、缓冲器数量和功耗方面有了显著提升. 时钟偏差的降低主要源于本文在自底向上过程中就通过GBC贪婪聚类算法基本控制了聚类内时钟偏差,并通过底层缓冲器重选位算法进一步降低了时钟偏差. 而在自顶向下过程中,二分区四分支类H树的构建过程将缓冲器插入与合并算法相结合,严格控制低时钟偏差的同时大幅度降低了缓冲器数量和功耗.

总之,本文的时钟树综合算法在所有关键指标上都表现出了卓越的性能,适用于大规模集成电路的高效时钟树综合.

5　结论

本文提出了一种基于二分区四分支的类H树结构,在自底向上过程中以贪婪思想迭代形成触发器聚类,并通过缓冲器重选位算法实现了底层时钟偏移和缓冲器数量的减少. 在自顶向下过程中推导得到了长路径中缓冲器按特定数量均匀插入能够得到当前路径下时钟延迟最小化的缓冲器插入策略,同时在类H树构建过程中将对称路径的缓冲器合并,实现了缓冲器数量的优化. 总体上看,本文构建的二分区四分支类H树结构在严格控制时钟偏差的前提下,尽最大程度优化了全局时钟延迟和缓冲器数量. 实验结果证明了本文方法对时钟树综合关键指标的优化与权衡. 本文的研究仍存在一些可以深入探索的方面. 在时钟延迟优化方面,虽然本文提出的缓冲器插入策略在给定路径下实现了延迟最小化,但其延迟结果与理论下界相比仍有进一步优化的空间. 未来的研究方向将主要集中在以下两个方面:首先,我们将继续探索更先进的缓冲器插入与拓扑优化算法,力求在维持甚至降低现有低时钟偏差水平的前提下,使全局时钟延迟能够更逼近其理论极限值. 其次,我们将致力于研究在多种工艺下均能保持稳健性的时钟树综合方法,以提升设计的可靠性,为先进工艺VLSI时钟树综合提供性能更出色的算法.

参考文献

[1] TENG S K, SOIN N. Low power clock gates optimization for clock tree distribution[C]//2010 11th International Symposium on Quality Electronic Design. Piscataway: IEEE, 2010: 488-492.

[2] 千路, 林平分. ASIC后端设计中的时钟偏移以及时钟树综合[J]. 半导体技术, 2008, 33(6): 527-529.
QIAN L, LIN P F. Clock skew and clock tree synthesis in ASIC backend design[J]. Semiconductor Technology, 2008, 33(6): 527-529. (in Chinese)

[3] 张祥, 赵启林. 基于缓冲器的ASIC芯片时序优化设计[J]. 集成电路与嵌入式系统, 2024, 24(12): 33-37.
ZHANG X, ZHAO Q L. Timing optimization design of ASIC chip based on buffer[J]. Integrated Circuits and Embedded Systems, 2024, 24(12): 33-37. (in Chinese)

[4] TSAI J L, ZHANG L Z, CHEN C C. Statistical timing analysis driven post-silicon-tunable clock-tree synthesis[C]// IEEE/ACM International Conference on Computer-Aided Design. Piscataway: IEEE, 2005: 575-581.

[5] PULLELA S, MENEZES N, OMAR J, et al. Skew and delay optimization for reliable buffered clock trees[C]//Proceedings of International Conference on Computer Aided Design. Piscataway: IEEE, 2002: 556-562.

[6] KUZMIN P A. Comparative analysis of the characteristics of clock network structures buffer tree, H-tree, clock mesh for technological nodes 28nm and 90nm[C]//2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials. Piscataway: IEEE, 2024: 270-279.

[7] ABDELHADI A, GINOSAR R, KOLODNY A, et al. Timing-driven variation-aware synthesis of hybrid mesh/tree clock distribution networks[J]. Integration, 2013, 46(4): 382-391.

[8] CONG J, KAHNG A B, KOH C K, et al. Bounded-skew clock and steiner routing[J]. ACM Transactions on Design Automation of Electronic Systems, 1998, 3(3): 341-388.

[9] PAPA D, ALPERT C, SZE C, et al. Physical synthesis with clock-network optimization for large systems on chips[J]. IEEE Micro, 2011, 31(4): 51-62.

[10] 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 II: Analog and Digital Signal Processing, 1992, 39(11): 799-814.

[11] SHIH X W, CHANG Y W. Fast timing-model independent buffered clock-tree synthesis[C]//IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Piscataway: IEEE, 2012: 1393-1404.

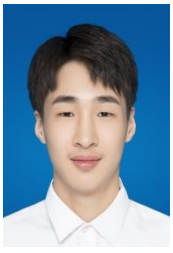
[12] 曾璇, 周丽丽, 黄晟, 等. 时钟延时及偏差最小化的缓冲器插入新算法[J]. 电子学报, 2001, 29(11): 1458-1462.
ZENG X, ZHOU L L, HUANG S, et al. A novel buffer insertion algorithm for clock delay and skew minimization[J]. Acta Electronica Sinica, 2001, 29(11): 1458-1462. (in Chinese)

[13] WANG H H, LEI Q Q, YANG Y F, et al. A clock tree synthesis scheme based on flexible H-tree[C]//2022 2nd International Conference on Electrical Engineering and

Mechatronics Technology. Piscataway: IEEE, 2022: 249-252.
[14] HAN K, KAHNG A B, LI J J. Optimal generalized H-tree topology and buffering for high-performance and low-power clock distribution[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(2): 478-491.
[15] LI W G, HUANG Z P, YU B, et al. iCTS: Iterative and hierarchical clock tree synthesis with skew-latency-load tree[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024, 44(10): 3948-3961.
[16] DING J H, WEN J Z, TANG H, et al. An effective and efficient cross-link insertion for non-tree clock network synthesis[C]//2025 Design, Automation & Test in Europe Conference. Piscataway: IEEE, 2025: 1-7.
[17] ISMAIL Y I, FRIEDMAN E G, NEVES J L. Equivalent elmore delay for RLC trees[C]//Proceedings of the 36th annual ACM/IEEE Design Automation Conference. New York: ACM, 1999: 715-720.
[18] EDAHIRO M. A clustering-based optimization algorithm in zero-skew routings[C]//Proceedings of the 30th international Design Automation Conference. New York: ACM, 1993: 612-616.
[19] NIU F F, ZHOU Q, YAO H L, et al. Obstacle-avoiding and slew-constrained buffered clock tree synthesis for skew optimization[C]//Proceedings of the 21st Edition of the Great Lakes Symposium on Great Lakes Symposium on VLSI. New York: ACM, 2011: 199-204.
[20] DENG C, CAI Y C, ZHOU Q. Register clustering methodology for low power clock tree synthesis[J]. Journal of Computer Science and Technology, 2015, 30(2): 391-403.
[21] AJAYI T, BLAAUW D. OPENROAD: Toward a self-driving, open-source digital layout implementation tool chain[C]//Proceedings of Government Microcircuit Applications and critical Technology Conference. Virginia: Defense Technical Information Center, 2019: 1105-1110.

作者简介

郭静静 女，1989 年 1 月出生于山西省大同市. 2011 年和 2014 年毕业于西安电子科技大学，获得微电子学学士学位和集成电路工程硕士学位. 2020 年毕业于东南大学，获得微电子学与固体电子学博士学位. 现任南京邮电大学讲师. 主要研究方向为统计时序分析.
E-mail: guo625jingjing@njupt.edu.cn



刘润衍 男，2004 年 8 月出生于浙江省宁波市. 2022 年进入南京邮电大学集成电路科学与工程学院，在读本科生. 主要研究方向为电子设计自动化.
E-mail: b22030712@njupt.edu.cn

杨君威 男，2004 年 6 月出生于江苏省常州市. 2022 年进入南京邮电大学集成电路科学与工程学院，在读本科生. 主要研究方向为电子设计自动化.
E-mail: b22030715@njupt.edu.cn

王嘉伟 男，2000 年出生于江苏省镇江市. 2022 年进入南京邮电大学集成电路科学与工程学院. 主要研究方向为静态时序分析.
E-mail: wjw756042721@163.com

王 冲 男，分别于 2012 年、2015 年和 2018 年在东南大学获得电子科学与技术学士、集成电路工程硕士和电子科学与技术博士学位. 现任南京理工大学副教授. 主要研究方向为数模混合集成电路设计、电子设计自动化.
E-mail: wangchong1111@njust.edu.cn

蔡志匡 男，1983 年出生于江苏省连云港市. 2014 年毕业于东南大学电子科学与工程学院. 现任南京邮电大学集成电路科学与工程学院教授、博士生导师. 主要研究方向为低功耗集成电路设计与测试. 中国电子学会会员编号：E1900019695S.
E-mail: whczk@njupt.edu.cn