一种基于同步并发分级聚类的时钟树综合方案

陈　阳，张树钢，俞泽文，肖　建
（南京邮电大学，南京 210023）

摘　要： 随着超大规模集成电路（Very Large Scale Integration Circuit，VLSI）制造工艺的快速发展以及其对应集成度的不断提高，数字集成电路的设计迎来了许多挑战。时钟树综合是数字后端设计的重要部分，现有的时钟树综合算法开始面临迭代效率变低和收敛速度变慢的问题。因此，提出了一种同步并发时钟树分级聚类算法（Synchronous Clock-tree Hierarchical Partitioning and Clustering，SC-HPC）。从系统优化的角度出发，SC-HPC 将原始的寄存器聚类过程转化为粗聚类和细聚类两步。粗聚类将布局完成的寄存器分为 N 大簇群，进一步把 N 个簇的细化任务分配给用户可调度的线程中进行加速处理。细聚类是根据缓冲器最大扇出的规则进行更加细致地划分寄存器。实验结果表明，相较于现有方法，SC-HPC 算法降低了缓冲器数量（30％以上）和程序运行时长（20％以上）。

关键词： 数字集成电路；时钟树综合；寄存器聚类；多线程加速
中图分类号：TN431.2　　**文献标志码：**A　　**文章编号：**1004-3365（2025）06-1049-07
DOI：10.13911/j.cnki.1004-3365.240473

A Clock-Tree Synthesis Scheme Based on Synchronous Concurrent Hierarchical Clustering

CHEN Yang, ZHANG Shugang, YU Zewen, XIAO Jian
（*Nanjing University of Posts and Telecommunications*, *Nanjing* 210023, *P. R. China*）

Abstract: Digital integrated circuit design faces several challenges due to the rapid development of VLSI manufacturing and increasing integration density. Clock-tree synthesis, a key part of digital back-end design, faces the issues of reduced iterative efficiency and slower convergence. In this study, we proposed the synchronous clock-tree hierarchical partitioning and clustering (SC-HPC) algorithm to overcome these challenges. The SC-HPC algorithm divides the original register clustering process into two steps: coarse clustering and fine clustering. The coarse clustering phase partitions physically placed the registers into N macro-clusters, with subsequent refinement tasks for these clusters being allocated to a user-configurable thread pool for parallel processing acceleration. Fine clustering further partitions the registers based on the buffer maximum fan-out rule. The experimental results indicate that the SC-HPC algorithm reduces the number of buffers by over 30% and runtime by over 20% when compared with the existing methods.

Key words: digital integrated circuits; clock tree synthesis; register clustering; multithreaded acceleration

收稿日期：2024-12-25；**定稿日期：**2025-03-18
基金项目：国家自然科学基金资助项目（U22B2024，62371256，U23B2042）
作者简介：陈　阳（2000－），男（汉族），四川成都人，硕士研究生，从事集成电路工程研究工作。
张树钢（2001－），男（汉族），山东滨州人，硕士研究生，从事集成电路工程研究工作。
俞泽文（2001－），男（汉族），浙江舟山人，硕士研究生，从事集成电路工程研究工作。
肖　建（1976－），男（汉族），江苏常州人，教授，硕士生导师，从事集成电路系统工程技术研究工作。通信作者。

0 引 言

数字集成电路设计分为前端设计和后端设计,前端设计是在逻辑层面实现电路设计并验证,后端设计是按照逻辑电路在物理层面实现电路设计。数字集成电路后端设计面临三大挑战:设计规模的持续扩展、设计目标与约束条件的日益复杂化以及设计流程的显著延长,这些挑战增加了芯片设计的时间投入和人力资源成本[1]。

在数字集成电路后端设计中,为保证时钟信号满足时序要求,需要通过插入缓冲器(Buffer)来构建时钟分布网。时钟分布网络的构建过程被称为时钟树综合(Clock Tree Synthesis,CTS)[2]。构建完成的时钟树会消耗大量功率,时钟树消耗的功率通常占比整个芯片功率预算的40%。时钟树的功耗大部分都在时钟树的叶级,时钟树的叶级包含寄存器、导线和缓冲器[3,4]。因此,合理的缓冲器插入位置以及较少的缓冲器数量可以有效提高设计效率和减少时钟树的功耗。

聚类算法是针对缓冲器进行优化的方法,将所有的寄存器聚成多个组群,并为每个组群分配一个时钟缓冲器。在时钟树综合过程中,结合机器学习的优势可以有效提高设计效率。因此,本文提出了SC-HPC算法,将输入总寄存器先进行粗聚类,再对粗聚类的每一簇进行细聚类,通过分级聚类来进行组群分配,提出的方法结合多线程并发处理可以有效减少缓冲器数量和时间成本。

本文的主要贡献包括:

1)数据保护的同步并发线程框架:面向时钟树综合,本文提出了一种同步并发处理的方法。为避免多线程处理遇到共享数据的访问与修改冲突问题,本文只设定一个共享数据—返回值(细聚类后的组群结果),并对返回值进行加锁处理。

2)粗细分级的聚类策略:针对大规模的寄存器建立时钟树,本文提出了一种分层级的聚类策略。该策略先进行粗颗粒级别聚类再进行细颗粒级别聚类。

3)真实工业场景数据集实验验证:本文采用2024中国研究生创芯EDA精英挑战赛提供的工业数据集进行了实验,验证了SC-HPC的优越性能。与现有方法相比,SC-HPC在保证通信质量的情况下,缓冲器的数量下降了30%以上,运行时间减少了20%以上。

1 时钟树综合

时钟树根据其在芯片内的分布特征,可分为多种结构,主要有H树、X树、平衡树等等[5]。H树通过在互连路径2个叶节点的几何中心点插入缓冲器,因此信号到达叶节点的时间理论上相等。X树也是一种实现等长互连线的方法,在4个叶节点的中心插入缓冲器[6]。上述2种结构的不足是主要适用于较小规模的设计场景,扇出难以协调。相较于这2种时钟树结构而言,平衡树比较适用于分层次的时钟树设计以及较大规模的时钟树综合。平衡树的构建过程分为2个阶段,聚类和平衡。在聚类阶段,将布局后的寄存器聚集成多个组群,形成叶级节点,汇聚到对应的缓冲器上,缓冲器再聚类最终汇聚到根节点(即时钟源)。平衡阶段需在满足缓冲器所带负载数目不超过最大扇出和负载电容不超过最大负载电容情况下,尽可能地减少时钟到各子节点的时钟偏移(skewness)和延迟(latency)。其中,时钟树的延迟由缓冲器、线长和寄存器延迟共同组成。时钟偏移是指时钟信号到达不同寄存器的时序差异。时钟树综合原理图如图1所示。

网络延迟和网络偏移的计算方式为:

$$latency_N=(N-1)\times buf_{\text{delay}}+\sum_{k=0}^{N-1} net\_delay_{k,k+1} \tag{1}$$

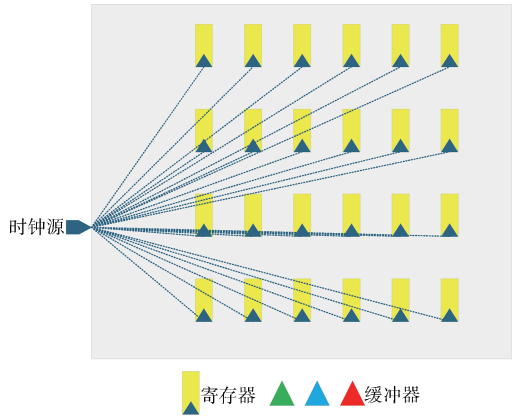
式中,N是时钟网络中单条网线中的器件数量,单条网线包含$(N-1)$个缓冲器和1个寄存器,buf_{delay}是缓冲器的延迟值,$(net_delay_{k,k+1})$是相连的器件k和器件$(k+1)$的线长延迟。

网络偏移的计算方式为:

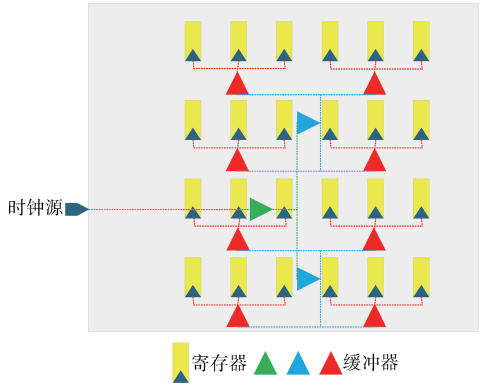
$$skew=\max_{k,1} skew_{k,1}=\max_{k,1}\left|latency_k-latency_1\right| \tag{2}$$

式中,$skew$定义为全局网络偏移,即最大延迟网络k减去最小的延迟网络l,得到网络偏移。

如式(1)所示,在保证信号完整性的条件下,通过优化缓冲器配置的数量可有降低传输路径的传播延迟,同时因缓冲单元数量减少带来的容性负载降低,能显著改善时钟网络的动态功耗特性。文献[3]通过设置最大距离和限制组群半径进行K-means聚类,实现了低功耗的时钟网络。文献[7]提出了一种自动寄存器放置技术,该技术能够为低功耗IC(Integrated Circuit,集成电路)合成低功耗时钟树。



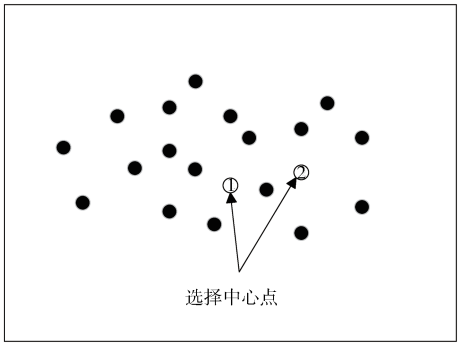
(a) 建树前



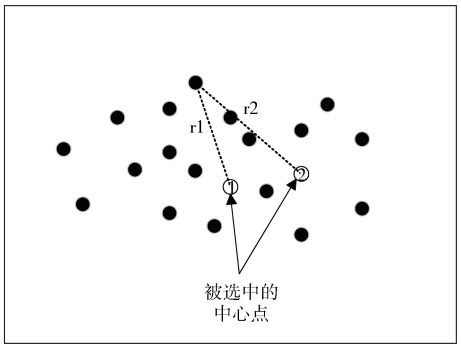
(b) 建树后

图 1　时钟树综合原理图

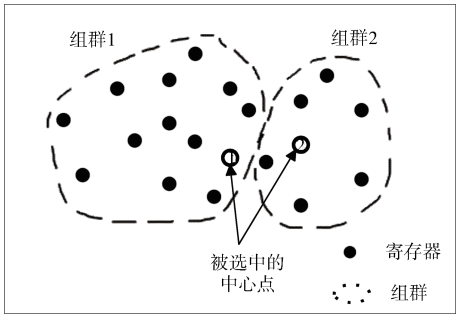
文献[8]提出广义分支H树,利用动态规划寻找最优的分支情况。文献[4]基于曼哈顿距离确定合理的组群和插入缓冲器的位置,通过K-means聚类构建最小线长的时钟树。K-means聚类过程如图2所示,当选中中心点后,根据待归纳的点距离(例如图2(b)中寄存器距离r1<r2,该寄存器归入中心点①分配组群。K-means聚类的缺陷是中心点的选择将决定组群,如果初始中心选择不当,可能会导致算法陷入局部最优解,而不是全局最优解[9]。



(a) 随机选择中心点



(b) 归纳近点



(c) 构建组群

图 2　K-means聚类示意图

在时钟树综合的层次化聚类阶段,传统聚类处理架构为串行处理,该架构下的时间开销会随着设计规模的扩展而增加。因此,多线程提速技术正受到广泛关注。多线程方案有两种:静态线程池方案和动态线程池方案。静态线程池在应用启动时预先创建固定数量的线程,通过重用减少创建和销毁开销,提升性能。动态线程池则根据任务负载、系统资源和执行情况动态调整线程数量,优化资源利用和任务处理效率[10]。文献[11]提出了一种兼具高效性和稳定性的改进算法。该算法利用静态多线程并行计算技术,高效地迭代计算图像中每个像素点与聚类中心之间的相似距离,并据此精确地确定出新的聚类中心,最终完成图像分类的任务。文献[12]在基于图的分割(Graph-Based Segmentation,GBS)算法的基础上,引入了静态多线程并行数据处理技术,从而显著加快了传统层次聚类算法的执行速度,实现了算法性能的优化。

针对以上问题,本文提出一种同步并发时钟树分级聚类算法(SC-HPC算法),通过带有数据保护的多线程并发框架和基于KmeansPlus算法对中心点智能选择的分级聚类,实现了对时钟树综合的优化。

2　SC-HPC 算法的设计

2.1　算法概述

本文提出的SC-HPC算法是在多线程架构下分级聚类。具体表现为首先预处理布局后的寄存器进行粗聚类,粗聚类的结果根据最大扇出数量进行细聚类,并且将细聚类任务放入线程池中进行多线程处理。聚类方式为KmeansPlus聚类,核心思想是设置距离较远的寄存器为中心点递归聚类,该方法更加智能地选择初始聚类中心。图3和算法1对SC-HPC算法进行了描述。

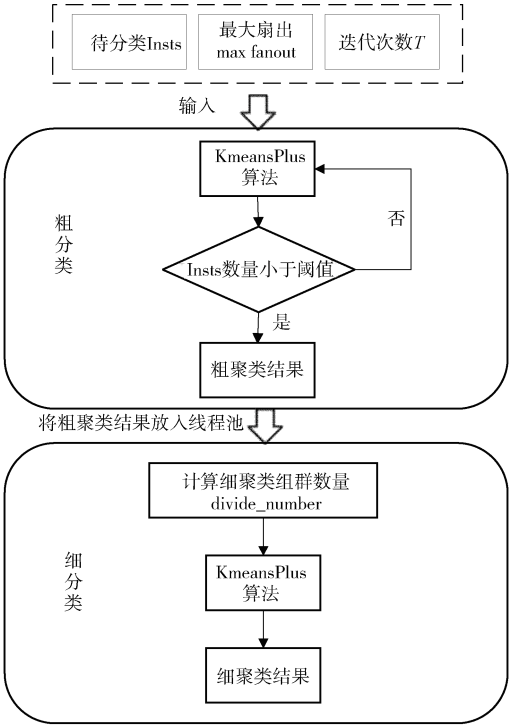


图3　SC-HPC算法结构框图

算法1(SC-HPC算法):

输入:待分类的器件(insts),最大扇出数量(maxfanout),迭代次数(T)

输出:聚类好的组群(clusters)(预处理,进行粗聚类)

1)粗聚类结果=KmeansPlus算法(所有器件insts,组群数k=6,随机数种子0,迭代次数T);

2)对粗聚类结果单组器件数量进行判断,大于再进行KmeansPlus聚类,直到单组器件数量小于阈值;

3)返回粗颗粒聚类下的组群结果(将粗聚类结果放入线程池,对每组粗聚类的组群进行细聚类)

4)对每组进行以下操作:

① 计算单组的细聚类的组群数量divide_number=单组器件数量/maxfanout;

② 细分类结果=KmeansPlus算法(单组器件insts,组群数k=divide_number,随机数种子0,迭代次数T);

5)将各单组聚类结果进行加锁处理;

6)待所有线程中的细分类处理完成,合并各单组聚类结果并返回最终值;

2.2　多线程架构设计

SC-HPC算法采用平衡树建树方式。在聚类过程中,本算法基于层次聚类构建了多线程架构,将分级聚类的过程融入到多线程架构当中,便于设备更加快速地构建时钟树。

分级聚类包含2个步骤:粗聚类和细聚类。基于粗聚类产生的组群结果具有数据独立性特征,同时考虑到细聚类阶段任务复杂度增加,算法采用线程池技术对细聚类任务进行并行化处理,从而实现计算效率的显著提升。由于设备可调度的最大线程数量固定,SC-HPC算法采用静态线程池作为并行计算框架。将所有细聚类任务分批次进入线程池,一批次的任务数量按照可用线程数设定。这里,本文可调用线程数为6,即一批次最大6个任务群如图5所示。

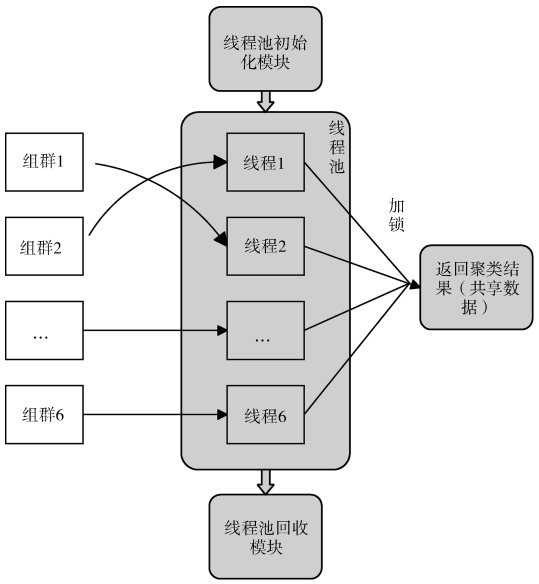


图4　线程池方案示意图

多线程处理过程中,各线程对共享数据访问和修改会引发冲突问题。为避免这种情况产生,SC-HPC算法通过加锁来控制对数据的访问与修改。该算法将共享数据尽可能减少,最后只使用一个共享数据返回细聚类的结果,并对其进行加锁处理。实验结果表明,数据完整,未存在遗漏或重复的情况。

2.3　分级聚类

在超大规模集成电路设计中,寄存器规模通常

超过100k量级时,本算法采用两阶段层次化聚类策略。预处理阶段基于空间分布密度分析,通过下式确定初始聚类数C_{init}:

$$C_{init}=\left\lceil\frac{N_a}{\alpha\bullet\rho}\right\rceil \tag{3}$$

式中,N_a为所有寄存器总数,α为经验系数(默认值20k,可配置),ρ为约束因子(0.8~1.2)。默认配置下,ρ取0.9,根据式(3)得出$C_{init}=6$。根据上述配置,实验结果表明提速效果为最佳,分得的组群数等于可调用的线程数。

粗聚类后,6个大组群进行了更加精细的聚类——细聚类,它的主要作用是限制单个组内的器件数量不超过最大扇出数量,这可以有效规避负载过大导致时钟信号传递失效的情况产生。根据上述多线程处理方法,6个线程分别处理6个组群,将每个组群按照不超过最大扇出的原则,使用KmeansPlus算法进行细聚类,会产生$\lceil N_s/maxfanout\rceil$个组群,$N_s$为单组寄存器总数。

在中心点的选择上,SC-HPC算法中采用的KmeansPlus比K-means更加智能,KmeansPlus算法的详解如算法2和图5所示,图5与图2相比,KmeansPlus的聚类显得更加合理。

算法2(KmeansPlus算法):

输入:待分类的器件insts,组群数k,随机种子seed,迭代次数T。

输出:聚类后的组群。

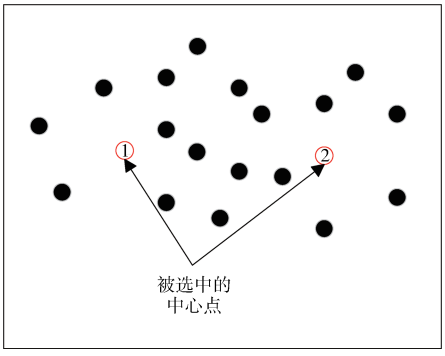
1)建立随机生成器gen;

2)从insts中随机选择一个inst做为中心点;

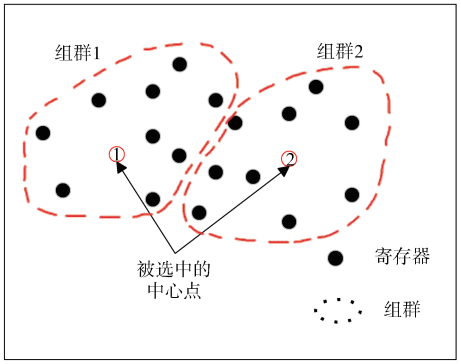
3)循环操作,计算其余的insts的曼哈顿距离,并找到较远的inst,做为新的中心点,直到找到k个中心点;

4)遍历剩余insts,将其归纳入最近的中心点,组成一个群;

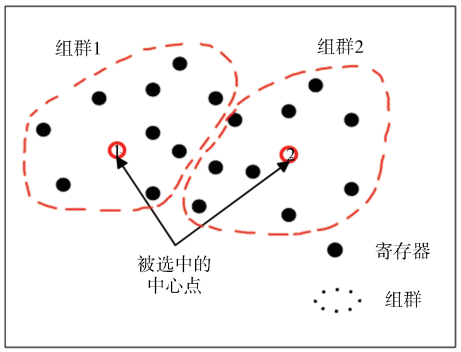
5)返回组建好的组群;



(a)选择相距较远的中心点



(b)归纳近点



(c)构建组群

图5　KmeansPlus聚类示意图

KmeansPlus的具体方法为:

1)从待聚类的器件中随机选择一个器件作为中心点,其余的器件计算与该中心点的曼哈顿距离:

$$D_{1,c}=|X_1-X_c|+|Y_1-Y_c|, \tag{4}$$

式中,$D_{1,c}$是选择的中心点1与其他未被选中的器件c沿X坐标和Y坐标方向的距离之和。

2)将每个寄存器与中心点的曼哈顿距离放入集合$\mathcal{D}$中:

$$\mathcal{D}=\{D_{1,2},D_{1,3},\ldots,D_{1,n}\}, \tag{5}$$

式中,n表示待聚类的器件总数,$D_{1,2}$表示中心器件1与器件2的曼哈顿距离,其他同理。

3)找出较大的曼哈顿距离,用式(5)将计算好的各个曼哈顿距离除以所有距离总数,得到一个概率值,距离越远越容易被选中做为新的中心点,从$\mathbb{D}_p$中按照概率选取新的中心点:

$$\mathbb{D}_{sum}=\sum_{k=1}^{n-1}D_{1,k+1},D'_{1,k+1}=\frac{D_{1,k+1}}{\mathbb{D}_{sum}}, \tag{6}$$

$$\mathbb{D}_p=\{D'_{1,2},D'_{1,3},\ldots,D'_{1,n}\} \tag{7}$$

4)重复1)~3)的操作,直到找出k个中心,k为需要聚类的组群数。

5)找出k个中心点后,遍历所有剩余器件并利

用式(7)计算其两点距离,将距离较近的器件归纳进对应的中心点的组群内:

$$distance_{k,l}=\sqrt{(x_k-x_l)^2+(y_k-y_l)^2}, \quad (8)$$

式中,$distance_{k,l}$表示中心点器件k与未被选中的器件l之间的距离。

3　实验结果与分析

本实验采用了2024中国研究生创芯EDA精英挑战赛的数据集。该数据集为工业仿真数据,对算法验证具有重要意义。数据集中一共有10个案例,每个案例的数据规模都不同,即具有不同的寄存器数量。每个案例包含:单位线长的电阻值、单位线长的电容值、最大限制的容阻值、最大扇出的数量、寄存器尺寸、缓冲器尺寸、时钟位置和大量寄存器及其位置。

本研究构建了多维度对比实验体系,将传统K-means算法作为测试基准,文献[8]的广义H-Tree采用基于K-means算法迭代聚类,将其作为对比实验。为确保聚类算法的有效性,实验将3个对比内容中的优化平衡部分设置一致。采用的优化平衡是调整缓冲器将缓冲器下的负载容阻值限制在最大限制之内,以及调整缓冲器的位置避免与其他器件发生重叠。

实验对10个案例进行了测试,实验结果图如图6所示。可以看出SC-HPC算法插入的缓冲器数量比K-means算法少了大约30%,比H-Tree算法减少了大约25%。寄存器规模越大,缓冲器数量减少的效果越明显。

实验挑选了3个规模跨度较大的案例进行运行时间分析。其中,Case2拥有115 077个寄存器,Case5拥有192 290个寄存器,Case9拥有170 629个寄存器。3种算法在3个案例中的结果对比结果如表1所示。

如表1所示,相比于K-means算法,SC-HPC算法插入的缓冲器数量分别减少35.5%、38.4%、36.4%,得益于缓冲器数量减少叠加多线程的处理,程序运行速度分别提升了24.2%、24.2%、25.0%。相比于H-Tree算法,SC-HPC算法插入的缓冲器数量分别减少了28.8%、31.1%、29.9%,程序运行速度分别提升了19.4%、32.4%、31.9%。

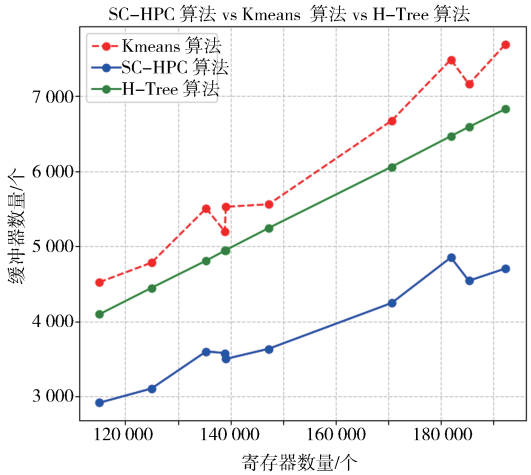


图6　十个案例的测试结果

由以上结果分析可得,H-Tree算法中多次迭代的聚类方式可以有效规避效果不佳的聚类结果。SC-HPC算法在迭代的基础上增加了分层级的聚类方式,实验结果表明聚类效果显著增强。SC-HPC算法产生的聚类组群大部分都能满足信号的传输完整,不需要在平衡优化阶段插入新的缓冲器。而K-means随机选择的中心点较近,有较多不满足的net_rc的组群,也就不能保证信号传输的完整性,所以需要设置新的中心点来解决net_rc问题,其中还遇到Case9的极端情况,K-means选择的2个中心点位置紧靠,导致缓冲器数量增加到9 486个,运行时长更是达到了10.5 min。

SC-HPC算法将待分类的器件先粗略地分为六大块,再进行更加细致的聚类分组,这样,有效减少了不合理的缓冲器。在时钟树平衡阶段,关于处理缓冲器重叠的时间复杂度为O(n),n代表缓冲器数量加上寄存器数量,缓冲器的减少会有效缩短平衡处理时间。不仅如此,多线程的加入也能将进程中存在的空闲时间利用起来用作其他任务处理,处理速度显著提升。

表1　3种算法在3个案例中的结果对比

案例算法	Case2		Case5		Case9	
	缓冲器数量/个	运行时间/s	缓冲器数量/个	运行时间/s	缓冲器数量/个	运行时间/s
K-means算法	4 523	198.3	7 695	594.8	6 674	494.2
H-Tree算法	4 096	183.7	6 830	666.3	6 060	548.4
SC-HPC算法	2 918	150.3	4 706	450.1	4246	370.2

4　结　论

超大规模集成电路中,时钟树综合是数字集成电路芯片设计必不可缺的一部分,随着集成规模越来越大,所消耗的时间成本也在不断增加,设计者们需要准确快速的EDA工具来加速数字芯片的设计过程。本文提出了SC-HPC算法是一种基于同步并发时钟树分级聚类的时钟树综合方案,通过KmeansPlus进行粗聚类,其结果同步并发处理,结合最大扇出规则,将每组粗聚类结果使用KmeansPlus进行细聚类。本算法旨在保证基本时钟信号传输不受影响下减少缓冲器数量并提高时间树综合的速率。真实数据集上的对比测试结果证实了提出算法的先进性。下一步,团队将针对skew和latency其他指标进行更进一步的优化。

参 考 文 献:

[1] 林亦波, 高笑涵, 陈廷欢, 等. 机器学习辅助数字集成电路后端设计方法[J]. 微纳电子与智能制造, 2021, 3(2): 11-20.

[2] 祝雪菲, 张万荣, 万培元, 等. 一种有效实现IC时序收敛的方法[J]. 微电子学, 2015, 45(4): 474-478, 483.

[3] 唐俊龙, 卢英龙, 戴超雄, 等. 一种低功耗时钟树综合的寄存器聚类方法[J]. 湖南大学学报(自然科学版), 2023, 50(8): 147-152.

[4] 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.

[5] UYSAL N, EWETZ R. An OCV-aware clock tree synthesis methodology [C]//2021 IEEE/ACM International Conference on Computer Aided Design. Munich, Germany. 2021: 1-9.

[6] ANIRUDH S, RAMESH T K. An enhanced clock tree synthesis methodology for optimizing power in physical design [C]//2022 IEEE 3rd International Conference on VLSI Systems, Architecture, Technology and Applications. Bangalore, India. 2023: 1-9.

[7] HOU W T, LIU D, HO P H. Automatic register banking for low-power clock trees [C]//2009 10th International Symposium on Quality Electronic Design. San Jose, CA, USA. 2009: 647-652.

[8] 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 an Systems, 2020, 39(2): 478-491.

[9] 叶帅辰. 基于Canopy-Kmeans++算法的网络威胁流量检测[J]. 信息技术与信息化, 2024(7): 155-158.

[10] 王鑫, 扈月, 刘坤禹, 等. Linux下多线程的方案实现与对比[J]. 信息记录材料, 2024, 25(3): 246-248.

[11] 崔利娜, 侯立旺. 一种快速稳健的K均值聚类算法[J]. 产业科技创新, 2022, 4(4): 48-49.

[12] 郭昕刚, 王佳, 程超. 层次聚类算法和基于图的分割算法相融合的图像分割算法[J]. 国防科技大学学报, 2022, 44(3): 194-200.