第 5 章

直角结构总体布线算法

5.1 引言

5.1.1 绪论

集成电路产业是当今最重要的行业之一，其发展水平已成为衡量国家工业发展水平的重要指标。集成电路产业提供的核心技术能促进各产业体系的创新，推动国民经济的增长，并可以为全球数字化提供强大的动力。另外，由于当前美国对华的技术壁垒，集成电路产业逐渐成为中美科技战的主战场。由此可见，如今正是发展我国高质量集成电路产业的黄金时期。

集成电路产业的发展催生了电子设计自动化(Electronic Design Automation，EDA)软件，而 EDA 软件作为工业设计必备的工具，进一步促进着集成电路产业的发展。在过去的 60 多年里，基于摩尔定律的预言，EDA 软件有力地支撑起集成电路产业，并有效地带动了集成电路产业的发展。随着集成电路工艺的发展，现代芯片的设计变得越来越复杂。作为芯片设计的第一环，技术人员必须借助 EDA 软件来完成芯片设计与优化。

EDA 软件贯穿整个集成电路的设计流程，从高级系统设计到制造。随着制程技术进入纳米时代，芯片密度不断增大，这对 EDA 软件的研究提出了巨大挑战。EDA 软件随着制造工艺的发展不断地更新。EDA 软件基本算法更新改进操作会有效改善集成电路设计的整体效益。因此，布线是 EDA 算法研究中一个极其重要的课题。

在整个超大规模集成电路(Very Large Scale Integration，VLSI)设计过程中，物理设计是与产品研制和生产最紧密相关的一个设计过程，直接关系到芯片设计的周期、生产成本和产品质量。布线是整个 VLSI 物理设计中最为耗时和关键的环节之一，它受工业影响很大，面临的机遇和挑战很多。

为解决复杂的布线问题，传统上将电路布线分成两大阶段进行处理，分别是总

体布线和详细布线。作为拥塞估计器,总体布线算法需要快速且准确地评估布局的质量,及时将拥塞信息反馈给布局器,让布局器能设计出更容易布线的布局方案。作为布线引导器,总体布线算法需要生成一个充分考虑设计规则的布线方案,给详细布线提供准确的指导,从而减少后续详细布线负担和时间。由此可见,总体布线是整个 VLSI 设计中极其重要的阶段,其结果将决定整个电路设计的质量。

总体布线应考虑多种布线指标。在现代 VLSI 设计中,总线广泛用于功能单元之间并行传递信号。随着制造工艺的发展和系统级芯片(System On a Chip,SOC)设计概念的出现,知识产权(Intellectual Property,IP)模块的广泛应用已成为芯片设计的主流,这使得总线在芯片设计中的数量急剧增加。尤其对于多核的 SOC 芯片而言,总线是决定时序和功耗的决定性因素,这对功能模块之间总线的互连提出了更高的要求。目前,总体布线算法的研究没有充分考虑总线的设计规则问题,这导致总线时序紊乱,严重影响芯片的性能。因此,亟需一种考虑总线的总体布线算法来有效地解决总线的时序匹配问题。

另外,随着多金属层的布线工艺技术的普及,层分配广泛地应用于总体布线中,以实现片上互联系统的精密布通。而目前的层分配算法大多都聚焦于通孔数和时延的优化,忽略了总线的特殊性。因此,对于考虑总线的总体布线问题,现有的层分配算法对总线偏差等布线指标的优化能力有限。

综上所述,为了减轻后续详细布线的设计负担,需要对总线进行正确且有效的早期规划,有效降低总线出现设计违规的可能性。为了提升总体布线方案的准确性,需要在总体布线阶段更加全面地考虑各种布线指标,尽可能地满足工业设计需求。由此可见,在总体布线阶段考虑总线的设计规则将是一个充满挑战且有意义的课题。

5.1.2 国内外研究现状

制造工艺的不断发展,给 EDA 软件带来了新的挑战。为了促进对总体布线算法的研究,国际物理设计研讨会(International Symposium on Physical Design,ISPD)分别在 2007 年和 2008 年举办了两次总体布线竞赛。另外,为了使总体布线更好地引导详细布线解决各种布线设计规则,计算机辅助设计国际会议(International Conference on Computer Aided Design,ICCAD)在 2019 年举办了基于 DEF/LEF 工业文档的开源总体布线竞赛。同时,为了有效解决工业界中总线布线出现的新问题,ICCAD 在 2018 年举办了考虑障碍的总线布线竞赛,以促进新一代芯片的产出。在 ISPD 和 ICCAD 竞赛的共同促进下,涌现出了许多优秀的总体布线算法以及总线布线算法。本节将从总体布线算法和总线布线算法两个方面分别进行介绍。

1. 总体布线算法

在 VLSI 布线结构中,布线区域通常是多层的。因此,总体布线问题通常抽象

成一个 3D 网格寻路问题。按照其布线方式不同，总体布线算法一般分成两类，分别是 2.5D 总体布线和 3D 总体布线。

1）2.5D 总体布线

2.5D 总体布线是指首先将多层的布线资源和引脚等信息映射到 2D 平面上，然后在 2D 平面上完成布线，最后使用层分配将所有的边指定到各个层上。因为 2.5D 总体布线是在 2D 平面上布线，与 3D 总体布线相比，在时间复杂度上有较大的优势，所以大多数的总体布线算法都使用该方法。文献[13]提出了一种高效的总体布线算法，采用一种拥塞驱动的 Steiner 树算法来构建线网拓扑结构，并使用一种边转移技术来改善拓扑。为了提高可布线性，文献[14]提出了单调布线和多起点、多终点迷宫布线算法，以时间代价换取高质量布线方案。文献[15]提出的算法是在文献[14]所提算法的基础上引入了“虚拟容量”的概念，即在预先估计通道容量的使用情况后，使用“虚拟容量”引导后面的迷宫布线，不仅有效地减少了溢出数，还提高了整个算法的运行效率。文献[16]提出的算法以最小化通孔数为目标，在文献[15]所提算法的基础上，提出了考虑通孔的 Steiner 树构造算法和三拐弯布线算法，实现了通孔数的优化。文献[17]提出了两种新颖的动态模式布线算法，在动态调整的布线区域中，以较少的时间代价取得了更好的总体布线方案。文献[18]提出了一种基于禁止布线区域的拆线重布算法。该算法在迭代过程中，通过限制布线区域，可不断降低布线拥塞。文献[19]提出了一种单调布线和自适应的多起点、多终点迷宫布线相结合的布线算法。文献[20]提出的算法通过修改文献[19]所提算法的代价函数和拆线重布的顺序，进一步提升了算法的有效性。文献[21]提出的算法以热点区域减少为目标，将对温度的考量加入到文献[20]所提出的算法中，使用了两种基于温度的代价函数，有效地减少了热点区域并解决了其内存浪费的问题。文献[22]提出的算法使用伪随机法决定线网布线顺序，并且使用改进的拆线重布方法来消除布线拥挤。文献[24]提出了的算法采用点移动和基于协商的 A^* 算法，可得到一个高质量的布线解。文献[25]提出了一种多阶段的总体布线算法，在不同阶段针对不同布线指标，确定不同的拆线重布顺序，最终实现了布线质量的提高。此外，在 2019 年 ICCAD 总体布线竞赛的激励下，文献[26]提出了一种详细布线驱动的总体布线算法，通过 L 形布线与 A^* 算法的有效结合，可生成一个满足详细布线设计规范的总体布线方案。

2.5D 总体布线需要通过层分配算法实现最终的 3D 布线。现有层分配问题的研究主要集中在通孔数和时延的优化上。最小化通孔在层分配中是 NP-难问题。文献[24]提出的算法采用整数线性规划模型求解通孔最小化问题。文献[28]提出的算法先对准备进行层分配的线网集合进行排序，再按照顺序采用动态规划方法为每个线网寻找最优的层分配方案。为了降低文献[28]所提算法陷入局部最优解的可能性，文献[29]提出的算法先对线网进行分解，然后再对线网进行排序，最后采用一种基于协商机制的重分配算法以跳出局部最优情况，从而获得一个更好的层

分配方案。针对现实工业中不同金属层上线宽和通孔尺寸不同的问题，文献[30]提出一种基于动态规划和线性规划的两阶段层分配算法。为了降低总体布线中通孔设计约束对后续详细布线的影响，在文献[30]所提算法的基础上，文献[31]中的通孔溢出和边溢出模型代替了原本模型。由于互连线的延迟对 VLSI 设计的影响越来越大，在减少通孔数的同时，研究人员也尝试优化层分配方案的时延问题。以时延为优化目标，文献[32]提出了一种时延驱动的动态规划算法，是对文献[29]所提出的算法进行拓展。该算法可以有效解决层分配阶段的时延优化问题。文献[33]提出的基于拉格朗日松弛法的层分配算法，迭代地优化所有线网的时延。在文献[34]提出的一种运用双轨线与宽线对关键线网的时延进行优化的算法基础上，文献[35]提出的算法通过调整线网的层分配顺序以及代价函数，进一步优化了关键线网的最大时延和总时延。

2）3D 总体布线

3D 总体布线是指直接在 3D 网格图是进行布线。由于 3D 总体布线在寻路过程中直接考虑了通孔，在溢出和线长上，它比 2.5D 总体布线有明显的优势。但随着布线问题规模的扩大，3D 总体布线的缺陷也越来越明显。文献[36]提出的算法将总体布线问题细分为矩形区域上规模较小的子问题，并使用预先定义的候选布线集，通过整数线性规划求解这些子问题。文献[36]提出的算法在最大限度地减少总溢出和总线长方面发挥了出色的作用，但付出了巨大的时间代价。随着制程技术的发展，总体布线问题规模也不断增大。为了提高算法的效率，同时保留 3D 总体布线方法在减少通孔数的优势，文献[37]提出的算法通过简化 3D 网格图，并使用高效的多层框架在 3D 网格图上拥挤的区域重新布线，使其在减少时间复杂度的同时布线结果接近原有的 3D 布线结果。

总的来看，上述总体布线算法主要侧重于溢出、线长和时间等布线指标的优化。由于没有充分考虑总线约束的布线，这些算法对于考虑总线的总体布线问题很难保持总线的各信号位的时序同步。同时，作为总体布线中的后阶段，已有的层分配算法也没有充分考虑总线信号位之间的时序同步问题。

2. 总线布线算法

对于总线布线的不同优化目标，很多机构已经进行了一系列的相关研究。针对时序优化，利用文献[38]提出的带有布线延迟估计的布局模型检测潜在的违规情况。文献[39]提出的算法是一种在控制芯片面积和减少布线线长的同时最小化布线通孔数的算法。文献[40]提出的算法是一种既可以最大限度地减少总线偏差，又可优化线长的有效算法。文献[41]提出的算法是一种用于高性能印制电路板的总线长度匹配算法，以确保满足最大和最小长度约束。文献[42]提出的方法是一种称为“虚拟线网拓扑”的方法，来复用布线拓扑。文献[43]提出的算法是一种用于通用布线拓扑的长度匹配的无网格布线算法，有效地处理不受拓扑限制的实际设计。针对考虑障碍物的总线布线问题，文献[44]提出的算法是一种采用最

长公共子序列和多商品流的启发式算法。此外，针对 2018 年 ICCAD 总线布线竞赛，文献[46]提出的算法是一种基于有向无环图的总线布线算法，满足总线的特定拓扑。文献[47]提出的算法是一种同时对总线的所有信号位进行布线，来实现拓扑一致构建的总线布线算法。在 A* 算法的基础上，文献[48]提出的算法是基于布线长度限制的总线布线算法。

综上所述，现有的研究工作很少考虑总线和非总线的综合布线。因此，本章针对总体布线中存在的问题展开研究，期望能提供更全面的总体布线工具，即不仅可以有效提高可布线性，还拥有综合考虑各种布线约束的能力。

5.1.3　本章主要工作

本章基于工业上的现实发展和需求，通过 2.5D 总体布线方法实现工业上的多金属层布线工艺，重点开展总体布线前期中拥塞估计与解决的问题、总体布线中总线和非总线线网的布线问题以及总体布线中层分配阶段总线偏差的优化问题等方面的算法研究工作，来弥补现有总体布线算法在总线和非总线线网布线方面存在的不足。本章提出了一种新颖的总体布线框架——考虑总线的偏差驱动总体布线算法框架，如图 5.1 所示。

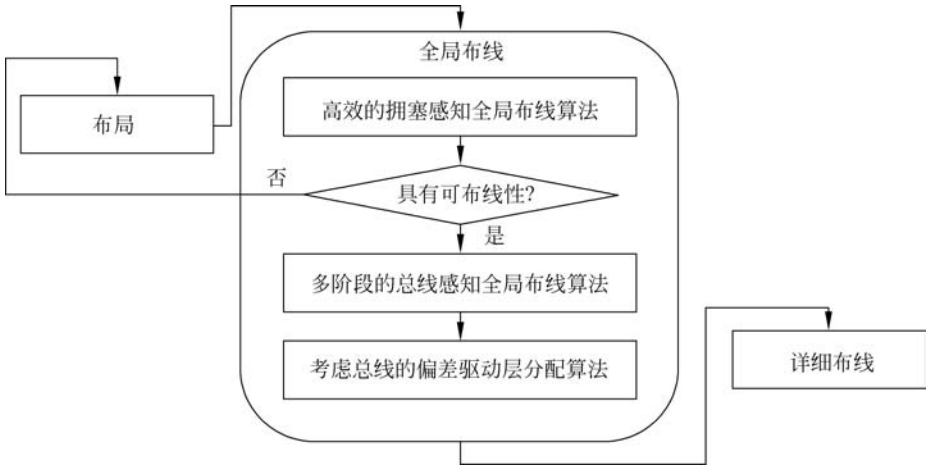

图 5.1　考虑总线的偏差驱动总体布线算法框架

针对不同的设计阶段，总体布线算法有不同的目标，起着不同的作用。与传统的总体布线框架相比，该框架不仅可以在布线早期预估存在的拥塞情况，还可以全面考虑总线的设计约束，给详细布线提供了高可布线性和高准确性的总体布线方案。总的来说，本研究在满足芯片的各种布线约束下，不仅有效地解决时序匹配问题，而且优化了总线和非总线线网的溢出数和线长，进一步提升了总体布线方案的质量。

本章具体的研究内容和主要贡献如下。

当总体布线和布局器协作时，总体布线如何快速且准确地反馈拥塞信息给布局器是一个重要的研究目标。另外，总体布线作为详细布线的引导者时，总体布线如何减少拥塞程度以有效减轻详细布线的负担是另一个难题。

因此，本章提出了一种高效的拥塞感知总体布线算法(C-GR)，以溢出、时间和线长为优化目标，适用于详细布局和总体布线之间的拥塞估计和解决。该算法引入了一种混合拓扑优化策略，来降低线网的拥塞程度；设计了一种基于区间划分的拥塞区域识别方法，来确定线网拆线重布的顺序；构建了一种基于拥塞松弛的启发式搜索算法，来实现线网的拆线重布。实验结果表明，所提算法能“高效”且准确地识别出拥塞区域，布线结果不仅具有较少的拥塞，还具有较短的线长。

5.2 问题描述

5.2.1 物理设计概述

VLSI 物理设计是将电路设计转化成芯片上的精确几何布局，直接影响电路的性能、功耗和功率。物理设计介于电路设计与制造之间，具有高复杂性。因此，物理设计通常分为 5 个关键的步骤，如图 5.2 所示。

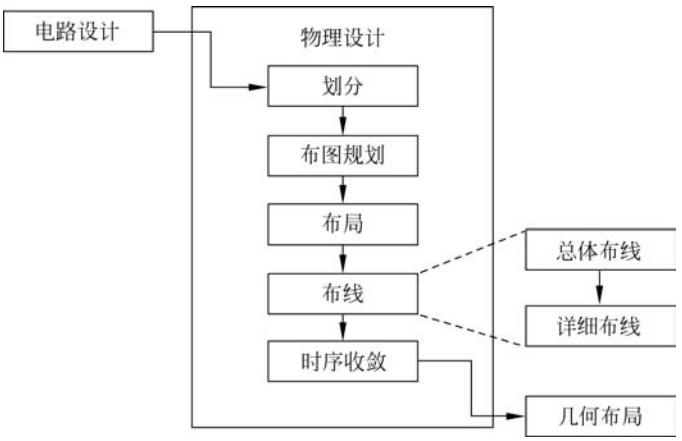

图 5.2 物理设计流程图

下面对各个阶段分别进行简要介绍。

(1) 划分：将电路分解成多个子电路或者模块，并且同时考虑不同子电路或者模块之间的连接数目最小化，使之能够单独设计或者分析。

(2) 布图规划：将分解后的子电路或模块根据芯片的大小做相应位置的摆放，即决定子电路或模块的形状和位置，使之能使整体利用芯片的面积最佳。

(3) 布局：根据布图规划，决定各个子电路或者模块中元件的最佳空间位置，同时考虑可布线性、时序、功耗和线长等指标。

(4) 布线：根据布局方案，在有限的布线资源下，完整地连接各个子电路或者模块之间所有的信号引脚。

(5) 时序收敛：利用布局和布线来优化电路性能。

物理设计直接影响芯片设计的生产周期、生产成本以及质量。随着制程技术的改良，物理设计受到的影响是最大的。

布线决定整个物理设计的成败，影响整个物理设计的质量。一个电路设计经过了划分、布图规划和布局后，才开始进行布线。这意味着该布局的现有资源已经固定，需要在有限的布线资源下，完成高质量的布线。布线分成两个阶段：总体布线和详细布线。在总体布线阶段，将每个线网的各部分合理地分配到各个布线单元中去，实现不同布线单元之间的布线，得到每个线网的大致布线路径。在详细布线阶段，根据总体布线的结果，完成每个线网在每个布线通道中的具体路径，得到最终的布线方案。

5.2.2　术语和定义

下面简要列出 VLSI 布线阶段所涉及的常用术语。

定义 5.1　引脚　一个电子终端，用于连接给定的构件到它的外部环境。在任意的金属层上，用 3 个坐标值 x、y、z 表示，代表该引脚在芯片的位置。

定义 5.2　线网　在相同电势下的需要连接的引脚集合。若一个线网有 3 个或者 3 个以上的引脚，则称该线网为多引脚线网。只有两个引脚的线网称为两引脚线网。

定义 5.3　金属层　芯片的布线层，通常指制造工艺等级。每个金属层都有一个布线偏好方向，要么水平，要么垂直。

定义 5.4　轨道　每个金属层中一条可用的水平或者垂直连线通路。轨道方向和布线偏好方向是一致的。

定义 5.5　通孔　指金属层之间的连线，用于连通不同金属层的布线结构。该连线具有相同的 x、y 坐标，但 z 坐标不同。

定义 5.6　布线区域　指包含了布线轨道的区域。

定义 5.7　通道容量　指两个布线单元之间存在的布线轨道数量。

5.2.3　总体布线模型图

总体布线问题是一个经典的图论问题。下面介绍总体布线模型图。

在 VLSI 布线中，布线区域分布在多个金属层，总体布线一般将每层划分为大小相同的单元，每个单元称为布线单元。图 5.3(a)给定 4 层金属层且每层被分成 3×3 个布线单元的总体布线结构图。在总体布线阶段，z 层的布线区域通常用网格图 $G^z=(V^z,E^z)$ 表示，节点 $v^z\in V^z$ 代表布线单元，边 $e^z\in E^z$ 代表相邻布线单元对 (v_i^z,v_j^z) 的连接边。边 E^z 由两部分组成，分别是布线边 E_e^z 和通孔边 E_v^z。其

中，布线边 $e^z \in E_e^z$ 表示其连接的布线单元对 (v_i^z, v_j^z) 位于同一层。通孔边 $e^z \in E_v^z$ 代表其连接布线单元对 $(v_i^z, v_i^{z'})$ 位于不同层。而通道容量 $c(e^z)$ 代表布线单元对 (v_i^z, v_j^z) 之间可用的布线轨道资源数，$d(e^z)$ 则是边 e^z 实际使用的轨道数。图 5.3(b)是图 5.3(a)对应的总体布线网格图。

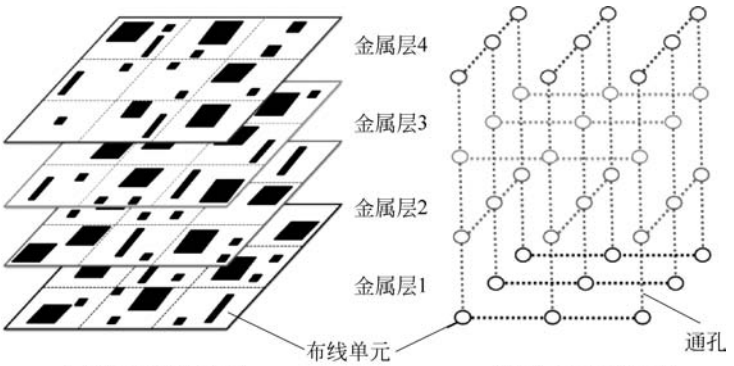

图 5.3 总体布线模型图

5.2.4 总体布线方法

在总体布线问题中，有两种方法可以解决 3D 总体布线问题，分别是 3D 总体布线和 2.5 总体布线。3D 总体布线算法是在 3D 布线网格图上直接进行寻路。虽然该方法可能会获得更好的布线结果，但是十分耗时。2.5D 总体布线是将 3D 布线网格图压缩为 2D 布线网格图，完成 2D 布线后，执行层分配算法得到 3D 布线方案。由于布线规模不断增大，2.5D 总体布线是主流的总体布线方法。下面通过图 5.4 来解释 2.5D 总体布线方法。

给定一个有 3 个金属层的总体布线图以及一个有 3 个引脚的线网，如图 5.4(a)所示。首先，将该 3 层的布线图投影到 2D 平面上，得到一个单层的布线图，并将线网的所有引脚也映射到对应的布线单元中，如图 5.4(b)所示。然后，执行 2D 总体布线算法，以获得 2D 总体布线方案，如图 5.4(c)所示。最后，执行层分配算法，将该线网的每条布线边分配到合适的金属层，并通过通孔保持连通，以得到最终的 3D 总体布线方案，如图 5.4(d)所示。

1. 2D 布线方法

本节将介绍本章所使用到的 2D 布线方法。

1) L 形布线

L 形布线是一种快速的布线算法。它只会搜索边界框所在的路径，因此，路径只有一个拐弯。L 形布线总共有两种搜索路径，分别是上 L 形和下 L 形，布线时一般选择代价小的路径。

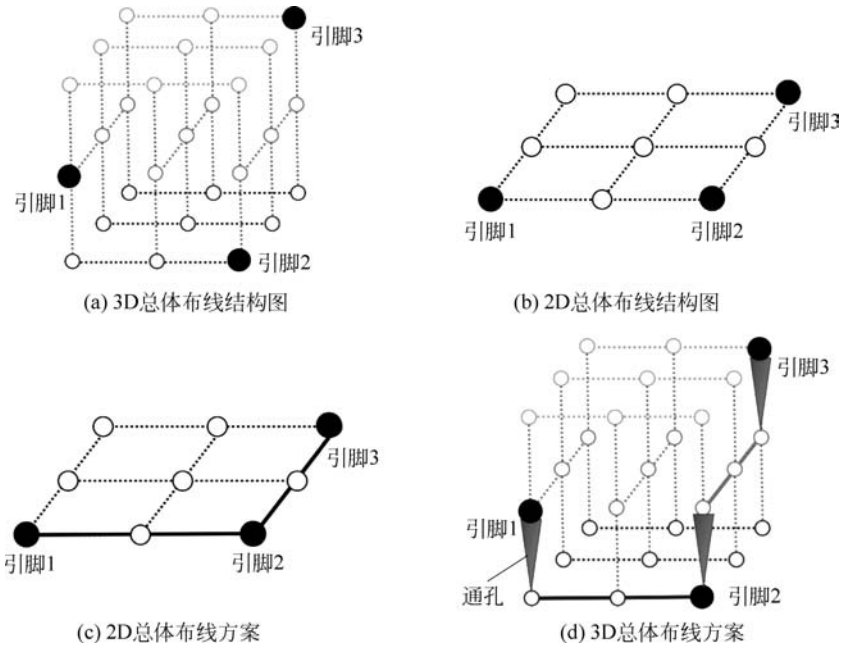

图 5.4　一种总体布线算法

2）混合单向单调布线

文献[50]提出的布线算法是混合单向单调布线(Hybrid Unilateral Monotonic Routing,HUMR)。给定一个重布边界框,HUMR 的路径由重布边界框中的两条路径组成。HUMR 将重布边界框中每个点都看成“中间点”,分别计算引脚对中两个引脚到“中间点”的代价,最终选择两条代价最小的路径组成最终路径。每条路径限定为只有在水平或者垂直其中一个方向不能产生绕行且代价最小。

3）自适应的多起点多终点迷宫布线

在拆完有溢出的引脚对的路径后,整棵布线树被分成了两棵子树。自适应的多起点多终点迷宫布线(Adaptive Multi-source Multi-sink Maze Routing,AMMMR)分别将每棵子树上所有的引脚看成起点和终点。然后,遍历起点和终点组成了所有路径。最后,选择代价最小的路径作为新路径。

4）A^* 迷宫布线

A^* 算法是一种高效的迷宫布线搜索算法。在搜索路径的过程中,A^* 算法会提前估计当前状态点到终点的各条路径的信息,并结合起点到当前状态点的权重,选择可能产生最佳解的路径进行处理。

5）协商布线

在总体布线中,通常会将每一个线网的线长以及拥塞情况等信息转化成一个代价值赋予布线边,作为线网经过该边时需承担的代价,此代价值即为布线的权值。在进行布线的过程中,会对每一条连线寻找其布线权重最小的路径。而协商

布线则是将先前的拥塞信息一起计入代价数值之中，以突显出总体布线中各布线边的使用需求情况，并进行最佳连线路径分配的方法。

6）拆线重布

如果在所有线网进行完一轮的总体布线后，总体布线的边上产生溢出，那么通常会将通过该边的连线路径全部拆除，再重新决定这些连线的路径，此方法称为拆线重布。通常使用了协商布线的总体布线算法会先将有溢出的路径拆线后，先依据布线重布代价函数进行布线代价的更新，再进行重新布线。

7）FLUTE 算法

FLUTE 算法是一个基于查找表的直角 Steiner 最小树（RSMT）构建算法。对于每个线网，FLUTE 算法会根据引脚的位置预先计算潜在的最优 Steiner 树和最优线长矢量，形成一个准确的查找表。给定一个线网，通过计算该线网的最优线长，返回相应的最优 Steiner 树，即可快速地得到相应的 RSMT。FLUTE 算法的时间复杂度为 $O(n\log n)$。对于 9 个及以下引脚的线网，FLUTE 算法能为其找到最优的 RSMT。对于 9 个以上引脚的线网，需要对线网进行切分，递归处理各个子线网。

2. 层分配

层分配是 2.5D 总体布线方法中重要的阶段。在层分配阶段，需要在不更改 2D 布线拓扑且不增加任何溢出的前提下，将每个线网的每条布线边分配到合适的金属层上，得到最终的 3D 总体布线方案。作为总体布线的后阶段，层分配应避免拥塞，同时实现时延、通孔数和串扰等性能指标的优化。现有实现层分配算法的策略有两种：顺序分配策略和并行分配策略。顺序分配策略按照特定的顺序依次对每个线网进行层分配，一个线网完成层分配后，更新布线空间的资源使用情况，再对下一个线网进行层分配。顺序分配策略所用的时间短，但是对线网的层分配顺序有依赖性，容易产生局部最优解。并行分配策略主要是基于整数线性规划，可以对所有线网进行并行分配。并行分配策略可以有效避免层分配对分配顺序的依赖性问题，理论上可以得到最优解。但是由于如今布线规模庞大，并行分配策略十分耗时，甚至可能会存在无法求解的情况。因此，大多数层分配算法都采用顺序分配策略。

5.2.5 总体布线的优化目标

按照引脚在布线单元中的位置，将其映射到总体布线网格图 $G^z=(V^z,E^z)$ 中相应的顶点上。总体布线的目标是为每个线网在 $G^z=(V^z,E^z)$ 上找到能将其中所有引脚连接起来的一条正确的路径。

总体布线的质量在不同阶段由不同目标衡量，溢出和线长是最常见的优化指标。在完成详细布局后，为了快速地判断布局的有效性，需要执行总体布线算法来评估拥塞并解决拥塞。因此，针对拥塞估计问题，总体布线算法的目标是可布线

性、运行时间以及线长。为了给详细布线一个准确的布线引导方案，需要总体布线算法综合考虑多种布线指标。在当今的 VLSI 设计中，总线的地位越来越高，在总体布线中无法忽略。因此，针对总体布线和详细布线的匹配度问题以及总线布线的设计约束问题，总体布线算法的优化目标是溢出、线长和总线偏差。作为总体布线的一部分，层分配算法将决定最终的 3D 布线方案。在考虑总线的层分配问题中，需要线长优化和总线偏差等布线指标。

5.3　C-GR：高效的拥塞驱动总体布线算法

5.3.1　引言

在纳米制程下，可布线性是 VLSI 设计中最重要的问题之一。总体布线作为布线的重要组成部分，上承详细布局，下启详细布线。对于详细布局而言，总体布线算法可以及时快速地反馈拥塞信息给布局器，以优化详细布局的结果，从而避免其生成无效的布局方案。对于详细布线而言，总体布线算法有效地解决了布线拥塞的问题，详细布线的负担大幅度减轻且所需的时间有效降低。

一般存在两种策略来识别拥塞区域，以提升可布线性。第一种策略是通过引脚的密度、线网的边界框或 Steiner 树算法来估计布线区域的潜在拥塞。虽然这种拥塞估计方法速度很快，但是通常无法得到详细布线中真实的布线路径。因此，这种拥塞估计策略准确性十分低。第二种策略是直接执行总体布线算法来分析布线拥塞。这种拥塞估计策略可以精确地识别拥塞信息，但是，这种方法的效率明显比第一种方法低。

综上，为了获得高质量的总体布线方案，同时兼顾算法的运行效率，本节基于第二种拥塞估计策略，提出了一种高效的拥塞驱动总体布线算法，称为 C-GR。C-GR 不仅能快速且准确地识别出拥塞区域，而且能有效地解决拥塞。本节工作的贡献如下。

(1) 针对线网拓扑构建问题，基于 Prim 算法和分治法，本节算法引入了一种混合拓扑优化(Hybrid Topology Optimization，HTO)策略。该策略可以根据线网的特征，构建不同的布线拓扑结构，从而达到降低布线区域的拥塞的目的。

(2) 为了有效地识别拥塞区域，本节算法设计了一种基于区间划分的拥塞区域识别(Congestion Area Identification，CAI)方法。该方法可以准确地识别出拥塞区域内的溢出线网，并确定其进行拆线重布的顺序。

(3) 为了提高可布线性，本节算法构建了一种基于拥塞松弛的启发式搜索(Congestion-Relaxed based Heuristic Search，CRHS)算法，用来实现线网的拆线重布。该算法通过设计重布区域和代价函数，既能高效地解决溢出，又可以优化线网的线长。

实验结果表明，本节所提出的高效的拥塞驱动总体布线算法 C-GR 能准确

且快速地识别出拥塞区域，并能有效地提高可布线性。与文献[45]中的算法相比，C-GR 在总的溢出上平均减少了约 16.3%，并且优化了约 23.7%的平均运行时间。此外，C-GR 还取得了约 1.2%的线长优化。

5.3.2 问题描述

正如 5.2 节所述，总体布线问题是一个典型的图论问题，布线区域被建模成化为网格图 $G^z=(V^z,E^z)$。其中，V^z 表示布线单元集合，E^z 表示连接布线单元的边集。边 e^k 的通道容量 $c(e^k)$代表第 k 层上相邻布线单元之间可用的布线轨道数量，而 $d(e^k)$则表示实际已使用的布线轨道数量。当实际已占用的轨道数量大于可用的轨道数时，则出现边溢出 $o(e^k)$。$o(e^k)$的计算方式如式(5.1)所示。

值得注意的是，与 ISPD07 和 ISPD08 竞赛规定一样，所有的通孔边 E_v^z 的容量是不受限制的。因此，通孔边 E_v^z 的溢出不做计算。另外，布线边 E_e^z 的拥塞程度 $\mathrm{cong}(e^z)$侧面体现出设计的可布线性。$\mathrm{cong}(e^z)$的计算方式如式(5.2)所示。

$$o(e^k)=\begin{cases}d(e^k)-c(e^k), & \text{如果 } d(e^k)>c(e^k)\\ 0, & \text{否则}\end{cases} \tag{5.1}$$

$$\mathrm{cong}(e^z)=\frac{d(e^z)}{c(e^z)} \tag{5.2}$$

一个优质的总体布线算法致力于为每个线网生成无溢出路径。对于拥塞驱动的总体布线算法而言，溢出数的最小化比其他指标有更高的优先级。除了溢出数这个主要目标，平均运行时间(CPU)和总的线长(TWL)也是本节的优化目标。

因此，拥塞驱动的总体布线问题可以描述为：给定一个 z 层的总体布线图 $G^z=(V^z,E^z)$，每条布线边的通道容量 $d(e^z)$，以及线网集合 $N=\{N_1,N_2,\cdots,N_m\}$。对于每个线网 $N_j\in N$，$1\leqslant j\leqslant m$，都有一组引脚 $P=\{p_1,p_2,\cdots,p_k\}$。由引脚所在布线单元中的位置，把其映射到 $G^z=(V^z,E^z)$中相应的顶点上。拥塞驱动的总体布线的目标是为 N_j 在 $G^z=(V^z,E^z)$上找能将 N_j 的引脚连接起来的一条正确路径，使得总的溢出数(TWO)和总的线长(TWL)最小化。TWO 和 TWL 的计算方式如下所示：

$$\mathrm{TWO}=\sum_{e^z\in E_e^z}o(e^z) \tag{5.3}$$

$$\mathrm{TWL}=\sum_{e^z\in E_v^z}d(e^z)\times V_{\mathrm{cost}}+\sum_{e^z\in E_e^z}d(e^z) \tag{5.4}$$

其中，z 是金属层数；V_{cost} 是通孔代价。

5.3.3 C-GR 算法设计与实现

1. 算法总体设计流程

C-GR 算法的流程图如图 5.5 所示。该算法主要由 3 个阶段组成：初始阶段、

主阶段和层分配阶段。在初始阶段，先将 3D 布线信息投影到 2D 布线平面上，并采用一种混合拓扑优化策略，以减少布线区域的初始拥塞程度，从而得到一个较好的初始布线方案。在主阶段，首先，为了确定线网拆线重布的区域和顺序，提出了一种基于区间划分的拥塞区域识别方法；然后，使用一种基于拥塞松弛的启发式搜索算法，可以有效优化溢出，并防止线长增加过多；最后，更改代价函数，在保证溢出不增加的前提下，为有溢出的线网找到一条没有溢出的路径，可以更好地提升可布线性。在层分配阶段，使用一种最小化通孔数的层分配算法将 2D 总体布线方案还原为 3D 总体布线方案。下面详细介绍 C-GR 算法的各个阶段以及所使用的策略。

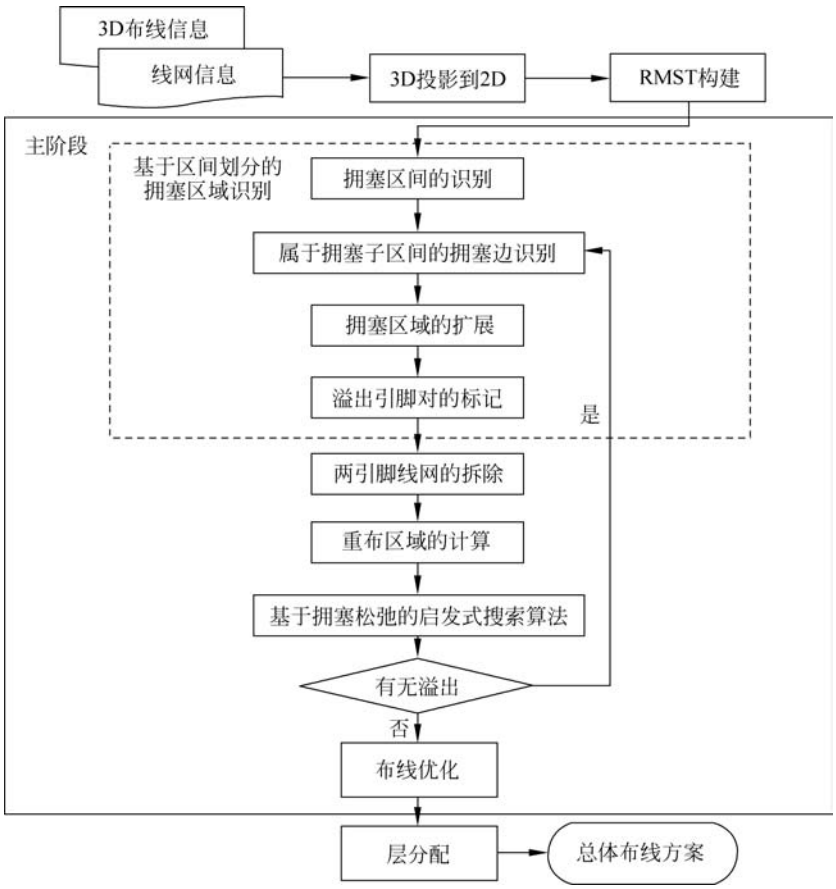

图 5.5　C-GR 算法的设计流程

2. 初始阶段

一个优质的初始总体布线方案，不仅可以减少拥塞程度，还可以减少后续拆线重布的时间代价。初始布线阶段的目的是快速地得到一个较好的总体布线初始解。正如 5.2.4 节所述，2.5D 总体布线方法比直接 3D 总体布线方法有明显的时间优势。因此，对于 C-GR 而言，2.5D 总体布线方法是一个最佳的选择。

在初始阶段，C-GR 需要将 3D 布线网格图和线网信息投影到 2D 平面上，并依次处理每个线网，构建其拓扑结构，来得到一个优质的初始布线方案。

对于拓扑结构的构建，直角最小生成树（Rectilinear Minimal Spanning Tree，RMST）和 RSMT 都是线网连接的常用模型。下面用图 5.6 展示两者各自的优缺点。虽然 RSMT 可以生成比 RMST 线长更短的树结构，但是 RSMT 会生成 Steiner 点，并且有很多的直接相连的两引脚线网，使得布线灵活性降低，增加了不必要的拥塞，最终影响到芯片的功率，如图 5.6（a）所示。除此之外，初始布线方案的拥塞较多，会导致后续拆线重布需要付出更多时间代价去减少溢出，严重影响算法的性能。相对比 RSMT，RMST 有更好的布线灵活性。图 5.6（b）是 RMST 的拓扑结构，其可以完全避开拥塞区域，得到一个无溢出的布线方案。图 5.6 中的矩形区域代表布线拥塞。

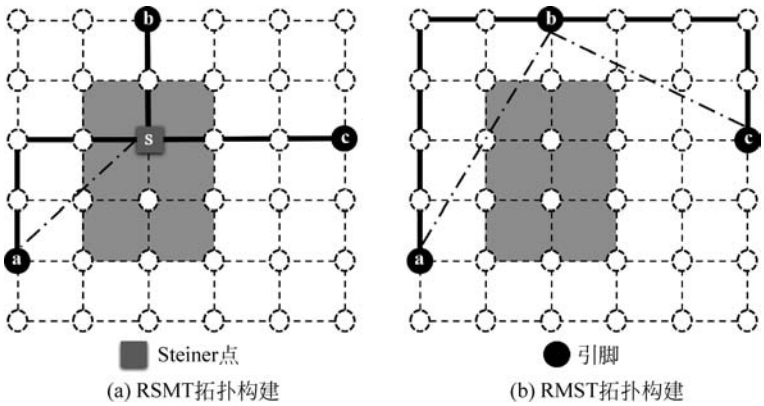

图 5.6　RSMT 和 RMST 拓扑对比

对于拥塞驱动的总体布线问题而言，平均拥塞比线长有更高的优先级。因此，基于 RMST 构建算法，C-GR 设计了一种混合拓扑优化策略，其可以为每个线网生成一个好的拓扑结构，从而保证有较低的拥塞程度。下面将介绍混合拓扑优化策略的原理和实现。

在总体布线结构图中，任意两个布线单元都是相互可达的，即对于给定的节点集 V，其对应的总体布线网格图是一个含 $|V|\times(|V|-1)/2$ 条边的无向完全图。Prim 算法和 Kruskal 算法都可在给定的加权连通图里筛选出权值总和最小的 RMST。Prim 算法的时间复杂度为 $O(n^2)$，其运行效率只与网格图上的节点数相关，适用于稠密图。而 Kruskal 算法的时间复杂度为 $O(e\log e)$，其运行效率与边数相关，适用于稀疏图。鉴于总体布线的网格图十分稠密，为了高效地构建 RMST，C-GR 使用基于节点选择的 Prim 算法。具体步骤如下：对于一个给定的 2D 总体布线网格图 $G^1=(V^1,E^1)$，首先，Prim 算法将 G^1 上的节点分成两类，一类是已包含在生成树中的节点（集合 A），剩下的节点则为另一类（集合 B），初始状态时，所

有的节点都位于集合 B；然后，任意选择集合 B 中一个节点作为起点；紧接着，选择集合 B 到集合 A 中的权值最小的节点加入到集合 A，重复以上步骤，直到集合 B 中没有节点；最终，可以得到 RMST。

另外，为了进一步降低拥塞程度，除了使用 Prim 算法生成 RMST 以减少不必要节点的方法，还可以适当调整 RMST 的拓扑结构。因为当引脚数大于 3 时，其对应的 RMST 有可能并不是唯一的，此时应当选择各边较为分散的 RMST 作为总体布线初始方案，以减少潜在的拥塞。为此，可以先运用分治算法剔除一些边，将 $G^1=(V^1,E^1)$ 里的 $|V|\times(|V|-1)/2$ 条边减少到 $O(|V|)$ 条较为分散的边，然后调用最小生成树构建算法生成 RMST。该算法可在 $O(n\log n)$ 的时间效率下构建各条边较为分散的 RMST。

基于以上的理论分析可知，Prim 算法和分治法有各自的优势。Prim 算法在引脚数较少时，构建 RMST 的效率优于分治法。分治法在引脚数较多时，构建 RMST 结构灵活性和效率比 Prim 算法高。因此，为了能同时有效地发挥两种算法的性能，C-GR 引入了引导因子 λ，将这两种 RMST 拓扑构建算法有效混合。具体的实现方式如下：假设给定线网中的待布线引脚数为 n，当 $n\leqslant\lambda$ 时，该线网使用 Prim 算法进行 RMST 拓扑构建，否则使用分治法构建拓扑。

对于初始阶段代价函数的选择，应该尽可能体现出拥塞的情况，以引导线网生成的拓扑均衡分布。因此，在初始阶段，C-GR 使用逻辑斯谛方程作为布线边的基本代价函数 b_e。b_e 的计算方式如下所示。

$$b_e=1+\frac{h}{1+\mathrm{e}^{-k\times(d(e)-c(e))}} \tag{5.5}$$

其中，e 是 2D 总体布线网格图 $G^1=(V^1,E^1)$ 上的布线边；k 和 h 是用户自定义的参数，分别设为 0.8 和 2。

下面以图 5.7 为例，解释初始阶段使用该代价函数的原因。

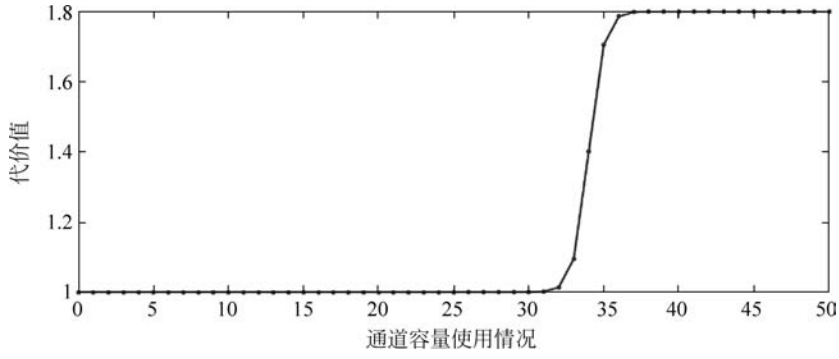

图 5.7　布线边的基本代价变化趋势

给定当前布线边 e 的通道容量为 35，即 $c(e)=35$。根据通道容量的使用情况，分成两种情况考虑。

(1) 当 $d(e)$接近 $c(e)$时,即通道容量即将耗尽,说明该布线边 e 的资源竞争激烈,期望后续的线网尽可能不使用该布线边的布线资源。因此,该布线边的代价将急剧增加。

(2) 当 $d(e)$远小于 $c(e)$时,即通道容量过剩,说明布线边 e 的资源剩余丰硕,鼓励后续的线网去使用该布线边的资源。因此,该布线边的代价几乎不增加。

综上所述,该代价函数可以根据当前的通道容量情况,显现出不同差异,达到有效平衡拥塞的目的。

3. 主阶段

提高可布线性是拥塞驱动的总体布线问题的主要目标。因此,主阶段的目的是拥塞区域的识别与溢出的减少。

1) 重布顺序的确定

在基于拆线重布的总体布线算法中,线网的拆线重布顺序对最终的总体布线方案有极大的影响。通常而言,倾向于先对布线边界框较小的线网进行拆线重布,这是因为该线网的布线灵活性比布线边界框大的线网差。但是,由于缺乏对布线区域拥塞程度的考虑,此布线顺序往往无法得到一个较好的布线方案。

因此,为了得到一个优质的两引脚线网的拆线重布顺序,C-GR 设计并使用一种基于区间划分的拥塞区域识别方法。通过计算拥塞值,定义拥塞区间,该方法可以快速且准确地得到拥塞区域以及两引脚线网的拆线重布顺序,从而为减少溢出做好准备。

基于区间划分的拥塞区域识别方法具体做法如下。

(1) 拥塞区间的定义。

首先,计算所有布线边的拥塞程度。在最大拥塞值与 1 之间平均划分为 10 个不同区间 $I_1, I_2, \cdots, I_{10}$。如图 5.8(a)所示,整个布线图中所有布线边的最大拥塞值为 2,所以它的子区间为{[2,1.9),[1.9,1.8),[1.8,1.7),…,[1.1,1)}。

(2) 属于拥塞子区间的拥塞边识别。

其次,根据拥塞边的拥塞值,将其分配给相应拥塞子区间。如图 5.8(a)所示,对于拥塞边 e 来说,它属于子区间[1.5,1.4)。属于拥塞子区间 I_i 的拥塞边 e 的定义如下:

$$e \in I_i, \quad \text{如果 } I_i.\min < \text{cong}(e) \leqslant I_i.\max$$

其中,$I_i.\max$ 和 $I_i.\min$ 分别是拥塞子区间 I_i 的上界和下界。

(3) 拥塞区域的扩展。

然后,从拥塞值最小的拥塞子区间 I_{10} 开始,为在这个拥塞区间内每一条拥塞边 e 生成一个拥塞区域 CR(e)。由于存在大量的布线边溢出,若从拥塞值大的拥塞子区间 I_1 开始,则会造成后续的迷宫布线花费大量的时间代价以及线长代价去生成无溢出路径。换言之,先解决拥塞值较小的拥塞子区间,可以让拥塞值较大的拥塞子区间有更多的布线选择,有利于加速算法。

拥塞区域的大小是由该拥塞边邻近的拥塞程度决定的，不断扩大直到该区域内所有边的平均拥塞程度不大于拥塞区间的 $I_i.\min$。如图 5.8(a)所示，虚线所围成的拥塞区域的平均拥塞值 $\mathrm{Avg}_{\mathrm{cong}}(\mathrm{CR}(e))=(3/2+2/2+3/2+4/2)/4=3/2$，其值大于 1.4。因此，拥塞区域进行 4 个方向的扩展，如图 5.8(b)所示。虚线所围成的拥塞区域的平均拥塞值 $\mathrm{Avg}_{\mathrm{cong}}(\mathrm{CR}(e))=1.1$，小于当前最小拥塞值 1.4，停止扩展。$\mathrm{Avg}_{\mathrm{cong}}(\mathrm{CR}(e))$的计算方式如下所示：

$$\mathrm{Avg}_{\mathrm{cong}}(\mathrm{CR}(e))=\frac{\sum_{e\in \mathrm{CR}}\mathrm{cong}(e)}{n} \tag{5.6}$$

其中，n 是拥塞区域 $\mathrm{CR}(e)$内布线边的边数。

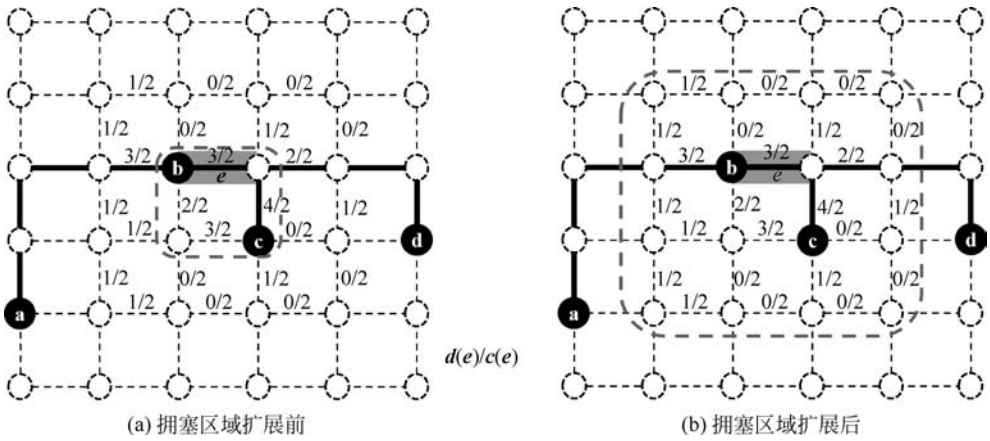

图 5.8　拥塞边 e 的拥塞区域扩展

(4) 两引脚线网的标记。

最后，标记拥塞区间中对应拥塞区域内的所有两引脚线网。值得注意的是，一旦两引脚线网中任一个引脚在拥塞区域内，那么它会被标记。

2) 拆线重布

为了提高电路可布线性，需要对已经标记好的两引脚线网进行拆线重布来优化溢出。

C-GR 的具体做法如下：假设线网 N 的拓扑 T 中一条边 e 发生了溢出。首先，将从 T 中拆除包含边 e 的两引脚线网的路径，这样 T 就拆分成两棵独立的子树 T_1 和 T_2，将其中一棵子树 T_1 上的所有引脚当成起点，另一棵子树 T_2 上所有的引脚当成终点；然后，计算进行重新布线的区域大小，以确定路径搜索的空间；最后，使用基于拥塞松弛的启发式搜索算法进行拆线重布。

通常情况下，溢出和线长无法同时优化。在拆线重布时，为了有效地优化溢出，通常需要进行绕行，这会导致线长的增加。因此，在优化溢出的前提，为了有效防止线长过多增加，C-GR 提出了一种基于拥塞松弛的启发式搜索算法。该算法结合了

A* 算法和自适应的多起点、多终点迷宫算法的优点，利用自适应的多起点、多终点迷宫算法来扩充路径搜索的解空间大小，并引入 A* 算法的启发式函数来对路径搜索的节点进行评估。基于拥塞松弛的启发式搜索算法的伪代码如算法 5.1 所示。假设 dis(v)是从 T_1 到节点 v 路径上所有布线边的总代价，parent(v)是记录 T_1 到节点 v 的最短路径上节点 v 的前一个节点，visit(v)用来标记节点 v 的访问状态。第 4～24 行是基于贪心选择的路径扩展。第 9～22 行是遍历当前节点 cur_v 的邻居节点 nei_v。当 dis(cur_v)＋cost(cur_v，nei_v)＜dis(nei_v)时，更新路径代价和线索数组。第 23 行是选择 Q 中代价最小的节点进行操作。

算法 5.1 基于拥塞松弛的启发式搜索算法

输入：子树 T_1，T_2，优先队列 Q，标记数组 visit(v)，线索数组 parent(v)，代价 dis(v)

输出：连接 T_1 和 T_2 的新路径 Path

```
1.   初始化优化队列 Q，标记数组 visit(v)、线索数组 parent(v)和路径代价 dis(v)
2.   令 cur_v = Q.Top();
3.   WHILE 队列 Q 不为空 DO
4.      IF cur_v 属于 T2 子树 THEN
5.         根据 parent(v)回溯以找到从 T1 到 T2 的最小代价路径 Path
6.      END IF
7.      Q.Pop();
8.      令 visit(cur_v) = 1;
9.      FOR 遍历当前节点 cur_v 的邻居节点 nei_v DO
10.        IF visit(cur_v) THEN
11.           遍历下一个邻近节点
12.        END IF
13.        IF dis(nei_v) > dis(cur_v) + cost(cur_v, nei_v) THEN
14.           dis(nei_v) = dis(cur_v) + cost(cur_v, nei_v);
15.           更新 parent(nei_v)节点为 cur_v;
16.           IF nei_v 存在于 Q 队列中 THEN
17.              对队列 Q 更新节点 nei_v;
18.           ELSE
19.              向队列 Q 插入节点 nei_v;
20.           END IF
21.        END IF
22.     END FOR
23.     令 cur_v = Q.Top();
24.  END WHILE
```

3）代价函数的选择

在拆线重布过程中，代价函数的选择对布线的结果起决定性的作用。为了有效减少溢出且避免线长过多增加，基于拥塞松弛的启发式算法引入了类似 A* 算法的代价函数 cost(N)。该代价函数计算方式如下所示：

$$\mathrm{cost}(N)=H(v)+P(v)\times\rho \tag{5.7}$$

其中，ρ 是权重因子；$H(v)$是当前节点 v 到终点的估计路径代价；$P(v)$是起点到当前节点 v 的实际路径代价。

启发函数 $H(v)$的选取很关键。为了避免在路径搜索时线长的过多增加，基于拥塞松弛的启发式搜索算法启发函数为

$$H(v)=|x_v-x_{\text{sink}}|+|y_v-y_{\text{sink}}| \tag{5.8}$$

其中，x_v、y_v 分别是当前节点 v 的横、纵坐标；x_{sink}、y_{sink} 分别是在曼哈顿距离下离 v 最近接收引脚的横、纵坐标。

为了有效权衡拥塞程度和线长，基于拥塞松弛的启发式搜索算法设计了一种有效的代价函数 $P(v)$来评估实际路径代价，$P(v)$的计算方式如下所示：

$$P(v)=\sum_{e\in P}(b_e+h_e+p_e+\text{vc}_e) \tag{5.9}$$

其中，P 表示该路径；b_e 是边 e 的基础代价；h_e 是历史代价；p_e 是拥塞惩罚项；vc_e 是通孔代价。

b_e 和 vc_e 是自适应的代价函数，它们的计算方式为

$$b_e=1-\text{e}^{-\alpha\times\text{e}^{-\beta\times i}} \tag{5.10}$$

$$\text{vc}_e=\lfloor 4\times b_e\rfloor\times v_{\text{g}} \tag{5.11}$$

其中，α 和 β 是用户自定义的参数；i 是迭代的次数；v_{g} 是估计通孔数。

下面通过图 5.9 来解释设计 b_e 的原因。图中颜色的深浅表示布线区域的拥塞程度。颜色越深，表明拥塞程度越严重。给定一个两引脚线网，路径 1 是只考虑线长得到的路径。路径 3 是过度重视拥塞而忽略线长得到的路径。路径 2 是同时考虑线长和拥塞得到的路径。通常，相比路径 1 和路径 3，路径 2 是最佳的路径。它比路径 1 经过更少的拥塞区域，比路径 3 占用更少的布线资源。但是，在布线过程中，3 种布线路径都是需要的。在布线初期，由于布线资源丰富，倾向于选择类似路径 1 的路径，使得线长最小化。在布线中期，为了避免布线资源过度使用，造成后续布线的线网发生溢出，倾向于选择路径 2。为了尽可能避免溢出，路径 3 是布线末期的最佳布线选择。

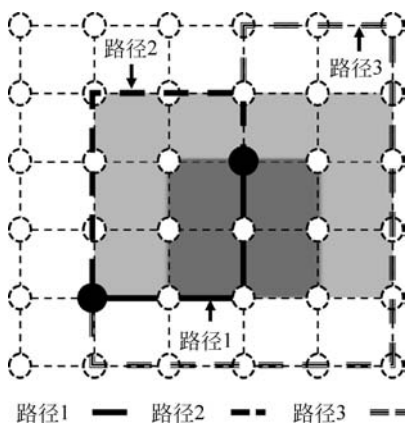

图 5.9　不同路径的选择

另外，h_e 和 p_e 的值会随着迭代次数的增加而变大。h_e 和 p_e 的计算方式为：

$$h_e^{i+1}=h_e^i+1,\quad \text{如果 } o(e)>0 \tag{5.12}$$

$$p_e=\left[\left(\text{cong}(e)+\frac{1}{c(e)}\right)\times f(h_e,i)\right]^k \tag{5.13}$$

其中，i 是迭代的次数；k 是用户自定义的参数；f 是与 h_e 和 i 相关的函数。

当所有布线边的溢出数达到用户预设值时，基于拥塞松弛的启发式搜索算法不再使用基于历史代价的代价函数式(5.7)来评价路径。为了尽可能地优化溢出这一最重要的布线指标，布线边将使用新的代价函数 cost(e)。cost(e)的计算方式如下所示：

$$\text{cost}(e)=\begin{cases}C, & \text{如果 } o(e)>0\\ 0, & \text{否则}\end{cases} \tag{5.14}$$

其中，C 是用户自定义的参数，设为 100。

为了提高 C-GR 的运行效率，在使用基于拥塞松弛的启发式搜索算法前，会设计一个重布区域来限制路径搜索的区域。这个重布边界框的初始大小等于进行重布的两引脚线网所围成的边界框大小。值得注意的是，在当前重布区域内无法找到一条无溢出的路径，重布区域会在下一次迭代时向四周扩大一个布线单元。换句话说，重布区域的大小会随着迭代次数的增加而变大，以增大路径搜索的空间，从而实现溢出的减少。

本阶段结束条件是所有布线边的溢出数为 0 或者迭代次数达到用户预设值。

4. 层分配阶段

在求解完 2D 总体布线问题后，C-GR 使用一种最小化通孔的层分配算法来得到最终的 3D 总体布线方案。该层分配算法首先根据线长和引脚数确定线网的层分配顺序，然后使用动态规划算法，依次将每个线网分配到合适的金属层上。

由于拥塞的识别与溢出的减少是拥塞驱动的总体布线问题的主要目标。如果 2D 布线无法有效解决溢出问题，则说明布局的结果并不理想。为了提高算法的效率，可能选择不运行层分配阶段，直接将拥塞信息反馈给布局器，让布局器重新布局，达到提高可布线性的目的。

5.3.4 实验结果与分析

本节所提算法 C-GR 由 C/C++语言编程实现，并在 CPU Intel Xeon 2.0GHz，RAM 96GB 的 Linux 服务器上运行。本节给出了所有基准电路集的详细描述。为了验证本节算法 C-GR 所提出 HTO 以及 CRHS 的有效性，对总的溢出(TWO)、总的线长(TWL)和平均运行时间(CPU)3 个指标进行实验对比。

1. 实验数据集

C-GR 所使用的基准电路集均来自 2008 年 ISPD 总体布线算法竞赛，表 5.1 展示了它们对应的基准电路名称(Benchmark)、布线单元数目(G-Cell)、金属层数(Layer)和线网数量(Net)。其中，所有的基准电路都是多金属层结构，并且每个金属层上的布线单元的数目都是几十万。最重要的是，基准电路中线网的规模从 219 794 增加到 2 228 930，从而显示出布线的规模越来越大，复杂度越来越高，这也充分说明了布线问题在 VLSI 物理设计中的地位愈发重要。

表 5.1　ISPD08 基准电路

Benchmark	G-Cell	Layer	Net
adaptec1	324×324	6	219 794
adaptec2	424×424	6	260 159
adaptec3	774×779	6	466 295
adaptec4	774×779	6	515 304
adaptec5	465×468	6	867 441
bigblue3	555×557	8	1 122 340
bigblue4	403×405	8	2 228 930
newblue2	557×463	6	463 213
newblue4	455×458	6	636 195
newblue5	637×640	6	1 257 555
newblue6	463×463	6	1 286 452

2. 混合拓扑优化策略的有效性验证

为了验证本节所提出的 HTO 的有效性。首先，使用 FLUTE 算法构建 RSMT 的布线方案与 Prim 算法、分治法构建 RMST 的布线方案进行对比，实验结果如表 5.2 所示。由实验结果可知，两种 RMST 构建算法对实验结果都存在一定的提升，但是也存在相应的缺点。相比于 FLUTE 算法，Prim 算法和分治法在 CPU 上有明显的优势。但是就 TWL 而言，Prim 算法和分治法都付出了约 0.4% 的代价。另外，对于基准电路 newblue4，分治法在 TWO 上发生了恶化，说明其在 TWO 的优化上存在局限性。

表 5.2　不同算法构建拓扑结构的比较

Benchmark	FLUTE			Prim			分治法		
	TWO	TWL (e5)	CPU (min)	TWO	TWL (e5)	CPU (min)	TWO	TWL (e5)	CPU (min)
adaptec1	0	62.7	11.8	0	62.4	9.5	0	62.3	9.4
adaptec2	0	63.4	2.7	0	63.7	2.1	0	63.7	2.2
adaptec3	0	146.9	4.7	0	148.2	4.3	0	148.1	4.3
adaptec4	0	135.7	2.1	0	137.4	2.1	0	137.3	2.1
adaptec5	0	167.0	16.7	0	167.2	12.6	0	167.0	12.7
bigblue3	0	142.3	3.6	0	143.6	3.4	0	143.3	3.4
bigblue4	138	241.6	14.2	118	242.6	13.5	114	242.6	14.1
newblue2	0	85.2	0.7	0	86.2	0.6	0	86.1	0.6
newblue4	190	140.3	23.9	182	140.5	19.8	212	141.2	30.6
newblue5	0	246.0	15.1	0	248.2	13.1	0	247.9	13.8
newblue6	0	197.5	32.4	0	195.9	23.4	0	195.7	23.4
RATIO	**1**	**1**	**1**	**0.915**	**1.004**	**0.816**	**1.006**	**1.004**	**0.912**

为了进一步提升性能，C-GR 有效地结合了这两种 RMST 拓扑优化算法，提出了 HTO 策略。根据线网的特征，自主决定线网生成 RMST 拓扑结构的方式。

在 HTO 策略中，引导因子 λ 是确定线网拓扑生成方式的阈值参数。如果 λ 设置太小，则线网的拓扑结构生成都由分治法生成，容易造成拓扑过于分散。如果 λ 设置过大，则线网的拓扑都因 Prim 算法生成，又很容易引起拥塞。考虑到基准用例中线网的引脚数量一般较小，此处 λ 只取 2～10 的整数进行参数实验，实验结果如表 5.2 和表 5.3 所示。

表 5.3 λ 取不同值时的效果比较

λ	MWO	TWL(e5)	CPU/min
2	212	148.7	10.6
3	**184**	**148.2**	9.1
4	192	148.6	9.6
5	228	148.2	9.6
6	186	148.6	9.3
7	192	148.6	9.6
8	202	148.2	10.1
9	202	148.6	9.5
10	200	148.2	9.3

从图 5.10 的趋势中可以发现，随着引导因子 λ 的变化，最大溢出数(MWO)、TWL 和 CPU 也发生相应的变化。由此可知，引导因子 λ 的取值对实验结果影响巨大。从表 5.3 的数据中也可以看出，当引导因子 λ＝5 时，MWO 的结果最差。另外，λ＝2 时，C-GR 的运行效率最差。当 λ＝3 时，实验得到的 TWL、MWO 和 CPU 三者都是最优的，故最终引导因子 λ 取 3。

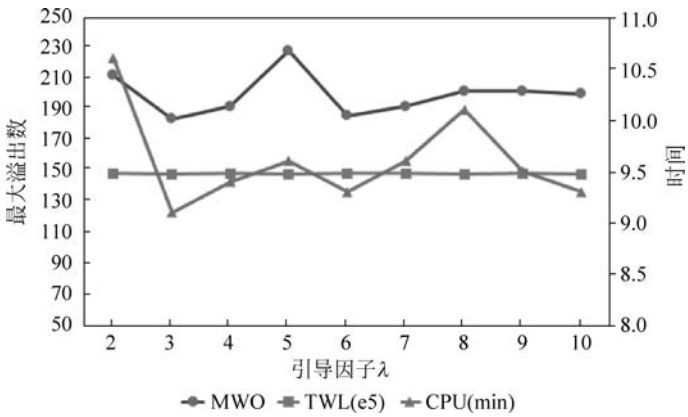

图 5.10 λ 取值的变化趋势

表 5.4 给出了使用 HTO 与使用 FLUTE 算法构建 RSMT 的实验数据对比。分析实验结果可知，对于 TWO 而言，所提出的 HTO 策略比 FLUTE 算法优化了

约 23.4%。对于每个基准电路，HTO 策略的运行效率明显比 FLUTE 高。这是因为 HTO 策略构建的线网拓扑结构质量高，初始布线方案的溢出数少，使得后续拆线重布执行迷宫布线算法的次数变少，节约了大量的时间。但是，由于 RSMT 布线结构在线长的优势，所提策略在 TWL 上出现了约 0.1%的恶化。总体上看，所提出的 HTO 策略能有效地解决 RSMT 构建拓扑存在的拥塞过于集中的问题，并在一定程度上缩短布线算法的运行时间。

表 5.4　混合拓扑优化策略的优化效果

Benchmark	FLUTE			HTO		
	TWO	TWL(e5)	CPU/min	TWO	TWL(e5)	CPU/min
adaptec1	0	62.7	11.8	0	62.3	9.5
adaptec2	0	63.4	2.7	0	63.7	2.2
adaptec3	0	146.9	4.7	0	148.1	4.3
adaptec4	0	135.7	2.1	0	137.3	2.2
adaptec5	0	167.0	16.7	0	167.0	12.8
bigblue3	0	142.3	3.6	0	143.3	3.4
bigblue4	138	241.6	14.2	100	242.5	14.0
newblue2	0	85.2	0.7	0	86.1	0.6
newblue4	190	140.3	23.9	184	140.7	18.6
newblue5	0	246.0	15.1	0	248.0	13.2
newblue6	0	197.5	32.4	0	190.9	19.4
RATIO	**1**	**1**	**1**	**0.866**	**1.001**	**0.783**

3. 基于拥塞松弛的启发式搜索算法的有效性验证

为了验证本节所提出的 CRHS 的有效性，将该算法与 AMMMR 进行了实验对比，实验结果如表 5.5 所示。从表 5.5 的实验数据可以看出，CRHS 比 AMMMR 在 TWL 上优化约 1.3%。这是因为所提出的 CRHS 能够在拆线重布的迭代过程中同时考虑布线图拥塞程度和线长因素的影响，从而避免了线长在拆线重布阶段过多的增加。但是，由于前期没有构建一个优质的拓扑结构，而且后期限制了线长的增加，使得拆线重布的次数增加，最终导致了 CPU 的恶化。由此可见，该算法需要搭配一个优质的初始布线方案。

表 5.5　基于拥塞松弛的启发式搜索算法的优化效果

Benchmark	AMMMR			CRHS		
	TWO	TWL(e5)	CPU/min	TWO	TWL(e5)	CPU/min
adaptec1	0	62.7	11.8	0	61.8	12.6
adaptec2	0	63.4	2.7	0	62.5	2.8
adaptec3	0	146.9	4.7	0	144.8	5.1
adaptec4	0	135.7	2.1	0	133.8	2.4

续表

Benchmark	AMMMR			CRHS		
	TWO	TWL(e5)	CPU/min	TWO	TWL(e5)	CPU/min
adaptec5	0	167.0	16.7	0	164.7	17.2
bigblue3	0	142.3	3.6	0	140.3	3.9
bigblue4	138	241.6	14.2	110	238.2	20.1
newblue2	0	85.2	0.7	0	84.0	0.6
newblue4	190	140.3	23.9	208	140.1	19.7
newblue5	0	246.0	15.1	0	242.6	16.1
newblue6	0	197.5	32.4	0	194.7	40.0
RATIO	**1**	**1**	**1**	**0.970**	**0.987**	**1.099**

4. 与其他总体布线算法的对比

为了验证本节算法 C-GR 的有效性，与文献[45]提出的总体布线算法进行对比，实验结果如表 5.6 所示。

表 5.6 最终优化效果

Benchmark	[45]			C-GR		
	TWO	TWL(e5)	CPU/min	TWO	TWL(e5)	CPU/min
adaptec1	0	62.7	11.8	0	61.5	10.2
adaptec2	0	63.4	2.7	0	63.0	2.1
adaptec3	0	146.9	4.7	0	145.9	4.6
adaptec4	0	135.7	2.1	0	135.4	2.1
adaptec5	0	167.0	16.7	0	164.5	13.8
bigblue3	0	142.3	3.6	0	141.3	3.5
bigblue4	138	241.6	14.2	96	239.5	18.7
newblue2	0	85.2	0.7	0	85.0	0.6
newblue4	190	140.3	23.9	186	140.1	20.9
newblue5	0	246.0	15.1	0	244.3	13.4
newblue6	0	197.5	32.4	0	188.1	20.5
RATIO	**1**	**1**	**1**	**0.837**	**0.988**	**0.863**

从表 5.6 可以看出，相较于文献[45]中的总体布线算法，本节算法 C-GR 在 TWO、TWL 和 CPU 方面分别取得了约 16.3%、1.2%和 13.7%的优化效果。最重要的是，从表 5.6 中的数据可以看出，对于每个基准电路，C-GR 在 TWL 都取得了一定的优化。

可见，C-GR 所提出的混合拓扑优化策略通过减少初始布线拓扑结构中节点的数量，并合理分散初始拓扑结构中边的分布位置，实现了降低电路中初始布线方案拥塞的目的，从而减小 TWO。另外，该策略还能进一步减少后续拆线重布的迭

代次数，缩短了整体流程的 CPU。除此之外，C-GR 设计了一种基于区间划分的拥塞区域识别方法能快速且有效地标记溢出线网，得到一个优质的拆线重布顺序。基于该方法，C-GR 所提出的基于拥塞松弛的启发式搜索算法能够在迭代拆线重布过程中，不仅考虑当前线网对当前区域内布线图拥塞程度的影响，而且还结合线长因素的影响，最终有效减少 TWL。

综上所述，本节算法 C-GR 不仅能有效识别和解决电路的拥塞，还能对 VLSI 总体布线的拥塞预测以及布局的优化带来帮助。

5.3.5　小结

本节针对拥塞估计以及可布线性判断问题，提出了一种高效的拥塞驱动总体布线算法(C-GR)。首先，引入了一种混合拓扑优化策略，可根据线网的特征，构建多种不同的拓扑结构的 RSMT，从而能够有效降低布线区域的拥塞程度，得到一个高质量的初始布线方案。然后，设计了一种基于区间划分的拥塞区域识别方式，能快速且准确地识别出拥塞区域，从而为后续拆线重布得到一个优质的重布顺序。最后，构建了一种基于拥塞松弛的启发式搜索算法，以有效地平衡拥塞程度和线长对布线结果的影响。实验结果表明，本节所提出的 C-GR 算法能对 TWO、TWL 以及 CPU 这 3 个重要的评价指标均取得有效的优化。

5.4　本章总结

总体布线作为物理设计中重要的设计阶段，上承详细布局，下启详细布线。本章提出了一个新颖的总体布线框架，主要的研究内容及创新之处如下。

为了提高拥塞估计的准确性和布局的可布线性，本章针对总体布线下拥塞估计和可布线性提高问题，以溢出、时间以及线长为目标，提出了一种拥塞驱动的总体布线算法 C-GR。C-GR 由 3 个阶段组成，分别为初始阶段、主阶段和层分配阶段。第一阶段，C-GR 使用一种混合拓扑优化策略来构建线网拓扑结构，力求得到一个较好的初始布线方案。第二阶段，C-GR 设计了一种基于区间划分的拥塞区域识别方法，并采用一种基于拥塞松弛的启发式搜索算法，实现线网的拆线重布。第三阶段，C-GR 使用一种最小化通孔的算法得到最终的 3D 总体布线方案。实验数据表明，所提出的算法 C-GR 能快速地识别拥塞区域，拥有较强的拥塞解决能力。

参考文献

[1]　朱晶. 我国集成电路产业高端化突破面临的问题研究及有关建议[J]. 中国集成电路，2020，29(Z3)：14-19.

[2]　于燮康. 我国集成电路产业面临的问题、挑战和发展途径[J]. 集成电路应用，2016，33(4)：

4-5.

[3] Lavagno L, Markov I L, Martin G, et al. *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*[M]. Boca Raton: CRC Press, 2016.

[4] Kahng A B, Lienig J, Markov I L, et al. *VLSI Physical Design: From Graph Partitioning to Timing Closure*[M]. Berlin: Springer Netherlands, 2011.

[5] 徐宁,洪先龙. 超大规模集成电路物理设计理论与方法[M]. 北京: 清华大学出版社, 2009.

[6] Wolf W. *Modern VLSI Design: IP-Based Design* [M]. Boston: Pearson Education Press, 2009

[7] Tang H, Liu G G, Chen X H, et al. A survey on steiner tree construction and global routing for VLSI design[J]. *IEEE Access*, 2020, 8: 68593-68622.

[8] Chen X H, Liu G G, Xiong N X, et al. A survey of swarm intelligence techniques in VLSI routing problems[J]. *IEEE Access*, 2020, 8: 26266-26292.

[9] Nam G J, Yildiz M, Pan D Z, et al. ISPD placement contest updates and ISPD 2007 global routing contest[C]//Proceedings of the International Symposium on Physical Design. 2007: 167-167.

[10] Nam G J, Sze C, Yildiz M. The ISPD global routing benchmark suite[C]//Proceedings of the International Symposium on Physical Design. 2008: 156-159.

[11] Dolgov S, Volkov A, Wang L T, et al. 2019 CAD Contest: LEF/DEF based global routing [C]//Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. 2019: 1-4.

[12] Cheng Y H, Yu T C, Fang S Y. Obstacle-avoiding length-matching bus routing considering nonuniform track resources[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2020, 28(8): 1881-1892.

[13] Pan M, Chu C. FastRoute: a step to integrate global routing into placement[C]// Proceedings of the IEEE/ACM International Conference on Computer Aided Design. 2006: 464-471.

[14] Pan M, Chu C. FastRoute 2.0: a high-quality and efficient global router[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2007: 250-255.

[15] Zhang Y H, Xu Y, Chu C. FastRoute 3.0: a fast and high quality global router based on virtual capacity[C]//Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. 2008: 344-349.

[16] Xu Y, Zhang Y H, Chu C. FastRoute 4.0: global router with efficient via minimization [C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2009: 576-581.

[17] Cao Z, Jing T T, Xiong J J, et al. Fashion: a fast and accurate solution to global routing problem[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008, 27(4): 726-737.

[18] Chen H Y, Hsu C H, Chang Y W. High-performance global routing with fast overflow reduction[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2009: 582-587.

[19] Gao J R, Wu P C, Wang T C. A new global router for modern designs[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2008: 232-237.

[20] Chang Y J, Lee Y T, Gao J R, et al. NTHU-Route 2.0: a robust global router for modern design[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2010, 29(12): 1931-1944.

[21] Lee Y T, Chang Y J, Wang T C. A temperature-aware global router[C]//Proceedings of the International Symposium on VLSI Design, Automation and Test. 2010: 279-282.

[22] Dai K R, Liu W H, Li Y L. NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2012, 20(3): 459-472.

[23] Liu W H, Kao W C, Li Y L, et al. NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013, 32(5): 709-722.

[24] Cho M, Lu K, Yuan K, et al. BoxRouter 2.0: a hybrid and robust global router with layer assignment for routability[J]. *ACM Transactions on Design Automation of Electronic Systems*, 2009, 14(2): 1-21.

[25] 朱自然，陈建利，朱文兴. 基于多阶段拆线重布的总体布线算法[J]. 计算机辅助设计与图形学学报，2016，28(11)：2000-2008.

[26] Liu J W, Pui C W, Wang F Z, et al. CUGR: detailed-routability-driven 3D global routing with probabilistic resource model[C]//Proceedings of the IEEE/ACM Design Automation Conference. 2020: 1-6.

[27] Naclerio N J, Masude S, Nakajima K. The via minimization problem is NP-complete[J]. *IEEE Transactions on Computers Society*, 1989, 38(11): 1604-1608.

[28] Lee T H, Wang T C. Congestion-constrained layer assignment for via minimization in global routing[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008, 27(9): 1643-1656.

[29] Liu W H, Li Y L. Negotiation-based layer assignment for via count and via overflow minimization[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2011: 539-544.

[30] Shi D H, Tashjian E, Davoodi A. Dynamic planning of local congestion from varying-size vias for global routing layer assignment[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2016: 372-377.

[31] Shi D H, Tashjian E, Davoodi A. Dynamic planning of local congestion from varying-size vias for global routing layer assignment[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017, 36(8): 1301-1312.

[32] Ao J C, Dong S Q, Chen S, et al. Delay-driven layer assignment in global routing under multi-tier interconnect structure[C]//Proceedings of the International Symposium on Physical Design. 2013: 101-107.

[33] Liu D R, Yu B, Chowdhury S, et al. TILA-S: timing-driven incremental layer assignment avoiding slew violations[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(1): 231-244.

[34] Han S Y, Liu W H, Ewetz R, et al. Delay-driven layer assignment for advanced technology nodes[C]//Proceedings of the Asia and South Pacific Design Automation Conference. 2017: 456-462.

[35] Zhang X H, Zhuang Z, Liu G G, et al. MiniDelay: multi-strategy timing-aware layer assignment for advanced technology nodes[C]//Proceedings of the Design, Automation & Test in Europe Conference & Exhibition. 2020: 586-591.

[36] Wu T H, Davoodi A, Linderoth J T. GRIP: scalable 3D global routing using integer programming[C]//Proceedings of the Annual Design Automation Conference. 2009: 320-325.

[37] Xu Y, Chu C. MGR: multi-level global router[C]//Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. 2011: 250-255.

[38] Pasricha S, Dutt N D, Bozorgzadeh E, et al. FABSYN: floorplan-aware bus architecture synthesis[J]. *IEEE Transaction on Very Large Scale Integration Systems*, 2006, 14(3): 241-253.

[39] He O, Dong S Q, Bian J N, et al. Bus via reduction based on floorplan revising[C]//Proceedings of the ACM Great Lakes Symposium on VLSI. 2010: 9-14.

[40] Wu P H, Ho T Y. Bus-driven floorplanning with bus pin assignment and deviation minimization[J]. *Transactions on Integration the VLSI Journal*, 2012, 45(4): 405-426.

[41] Ozdal M M, Wong M D F. A length-matching routing algorithm for high performance printed circuit boards[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006, 25(12): 2784-2794.

[42] Mo F, Brayton R K. A semi-detailed bus routing algorithm with variation reduction[C]//Proceedings of the International Symposium on Physical Design. 2007: 143-150.

[43] Yan T, Wong M D F. BSG-Route: a length-constrained routing scheme for general planar topology[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2009, 28(11): 1679-1690.

[44] Zhang R, Pan T Y, Zhu L, et al. A length matching routing method for disordered pins in PCB design[C]//Proceedings of the Asia and South Pacific Design Automation Conference, 2015: 402-407.

[45] Liao P X, Wang T C. A bus-aware global router[C]//Proceedings of Synthesis and System Integration of Mixed Information Technologies. 2018: 20-25.

[46] Hsu C H, Hung S C, Chen H, et al. A DAG-based algorithm for obstacle-aware topology-matching on-track bus routing[C]//Proceedings of the ACM/IEEE Design Automation Conference. 2019: 1-6.

[47] Chen J S, Liu J W, Chen G J, et al. MARCH: maze routing under a concurrent and hierarchical scheme for buses[C]//Proceedings of the ACM/IEEE Design Automation Conference. 2019: 1-6.

[48] Kim D, Do S, Lee S Y, et al. Compact topology-aware bus routing for design regularity[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2020, 39(8): 1744-1749.

[49] Kastner R, Bozorgzadeh E, Sarrafzadeh M. Pattern routing: use and theory for increasing predictability and avoiding coupling[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, 2002, 21(7): 777-790.

[50] Liu W H, Li Y L, Koh C K. A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing[C]//Proceedings of IEEE/ACM International Conference on Computer-Aided Design. 2012: 713-719.

[51] 青鸿阅. 基于并行 A～＊算法的 VLSI 线网布线研究[D]. 黑龙江：哈尔滨工业大学，2016.

[52] Chu C, Wong Y. FLUTE: fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, 2008, 27(1): 70-83.

[53] Lou J, Thakur S, Krishnamoorthy S, et al. Estimating routing congestion using probabilistic analysis[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, 21(1): 32-41.

[54] Westra J, Bartels C, Groeneveld P. Probabilistic congestion prediction[C]//Proceedings of the International Symposium on Physical Design. 2004: 204-209.

[55] 孟畅，蔡懿慈. 基于数据场的总体布线拥挤度计算模型[J]. 微电子学，2013，43(2)：296-300.

[56] Liu W H, Wei Y G, Sze C, et al. Routing congestion estimation with real design constraints [C]//Proceedings of ACM/EDAC/IEEE Design Automation Conference. 2013: 1-8.

[57] Zhou Z H, Chahal S, Ho T Y, et al. Supervised-learning congestion predictor for routability-driven global routing[C]//Proceedings of the International Symposium on VLSI Design, Automation and Test. 2019: 1-4.