清 华 大 学

综 合 论 文 训 练

题目：无网格布线算法优化及实现

系 别：计算机科学与技术系

专 业：计算机科学与技术专业

姓 名：赵锦明

指导教师：姚海龙

2013年 06月 11日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签 名： 导师签名： 日 期：

中文摘要

集成电路的发展是当今社会很多方面发展的基础，但随着集成电路的规模不断增大，手工设计已经无法满足要求，集成电路自动化设计工具的需求十分紧迫。由于数字集成电路的设计规则比较简单，数字集成电路自动化设计工具目前已经逐渐成熟，而模拟集成电路设计比较复杂，在自动化设计方面还很落后，这也成为了限制集成电路发展的瓶颈。布线模型及算法作为集成电路自动化设计工具中的核心，对于它的研究也就显得尤为重要。

目前在模拟集成电路布线模型和算法方面的研究很多，也有了很多不错的研究成果，但是现有的布线模型和算法在布线速度和布线质量平衡方面还存在不足，或者偏向于布线速度，或者偏向于布线质量。本文是基于实验室提出的分裂图布线模型[[1](#_ENREF_1)]来对无网格布线算法的优化及实现进行研究。主要内容是在原有模型基础上，提出并实现了新的分裂块构造算法DTC(Direct Tile Construction，直接分裂块构造)算法，同时加入了Corner Stitching(角勾链)[[2](#_ENREF_2)]结构并优化算法，最后通过与原有模型进行测试比较，对优化的效果进行了分析和验证。

关键词：模拟电路布线算法；分裂图模型；分裂块构造算法；角勾链结构

ABSTRACT

The development of integrated circuits in many ways is the basis for development of today's society. But with the increasing size of integrated circuits, the manual design has been unable to meet the requirements. The demand for integrated circuit design automation tools is very urgent. As digital integrated circuit design rules is relatively simple, digital integrated circuit design automation tool currently has gradually matured. While the analog integrated circuit design is more complex, the design is still lagging behind in automation, which has also become the bottleneck that limits IC development. Routing model and algorithm is the core for integrated circuit design automation tools, thus the research about it is particularly important.

Currently a lot of research have been done in analog IC layout model and algorithm aspect, and have a lot of good achievement. But for the existing routing model and algorithm, routing speed and cabling are in the mass balance, there are still insufficient, or biased in favor of routing speed, or bias the quality of the routing. This paper is to research the optimization and implementation of gridless routing algorithm based on the Splitting-Graph-Based model proposed by my laboratory. The main contents of this paper are based on the original model, propose and implement a new splitting tile construction algorithm ——DTC (Direct Tile Construction) algorithm, and the Corner Stitching structure is used. Finally, this paper proves the effect of optimization by instance testing compared to the original model.

Keywords：analog circuit routing algorithms; Splitting-Graph-Based model; splitting tile construction algorithm; Corner Stitching structure

主要符号对照表

如果论文中使用了大量的物理量符号、标志、缩略词、专门计量单位、自定义名词和术语等，应编写成注释说明汇集表。若上述符号和缩略词使用数量不多，可以不设专门的汇集表，但必须在论文中出现时加以说明。

DTC 直接块构造(Direct Tile Construct)

EDA 电子设计自动化(Electronic Design Automation)

SoC 片上系统(System on Chip)

PTR 多边形转化成矩形(Polygon-to-Rectangle)

MHS 最大水平条(Max Horizontal Strips)

MVS 最大竖直条(Max Vertical Strips)

目 录

第1章 引言 1

1.1 选题背景与意义 1

1.2 研究内容 2

1.3 论文结构 2

第2章 布线模型 3

2.1 布线模型概述 3

2.2 无网格布线模型 3

2.2.1 基于隐式连接图的布线模型 3

2.2.2 基于网块的布线模型 4

2.3 实验室前期工作 5

2.3.1 现有模型的不足 5

2.3.2 分裂图模型 5

2.3.3 分裂图模型中的问题 6

第3章 整体算法流程 7

3.1 算法流程 7

3.2 算法重要部分详细介绍 8

3.2.1 布局初始化 8

3.2.2 搜索布线 8

第4章 PTR算法和角勾链结构简介 9

4.1 PTR算法介绍 9

4.2 角勾链结构介绍 10

第5章 DTC算法详细及修改的角勾链结构介绍 11

5.1 DTC算法详情 11

5.1.1 算法思想 11

5.1.2 详细过程 13

5.1.3 算法伪代码 15

5.2 修改后角勾链结构介绍 16

5.2.1 角勾链结构的修改 16

5.2.2 修改的角勾链结构中实现的算法 17

5.3 算法分析 17

第6章 实验结果 19

6.1 实验测试结果 19

6.1.1 部分测例 19

6.1.2 测试结果 20

6.2 实验结果分析 20

第7章 总结与展望 21

7.1 毕设总结 21

7.2 工作展望 21

插图索引 23

表格索引 25

参考文献 27

致 谢 29

声 明 31

附录A 书面翻译 33

# 引言

## 选题背景与意义

目前社会已经进入信息时代，各种信息已经成为人们生活中密不可分的部分，而支撑着信息社会不断发展的一个重要基础是集成电路，社会中信息的存储、处理、传播等等都离不开集成电路这一基础，所以可以说是集成电路的发展决定和推动了社会的进一步发展。

在集成电路发展初期，主要依靠单纯的手工设计，工作量大、耗时长。然而随着集成电路逐步发展，目前已经进入深亚微米级别，SoC（片上系统）已经成为主流，并且电路的规模越来越大，工艺约束和要求越来越复杂，人工再也无法满足集成电路设计的需求，集成电路的设计必须借助计算机的辅助，也就是EDA（电子设计自动化）工具。

集成电路按照功能和结构可以分为模拟集成电路和数字集成电路两大类。模拟集成电路是用来处理模拟信号，例如进行信号的放大、过滤等；而数字集成电路是用来处理数字信号，主要用来进行复杂的信号计算。目前数字集成电路由于设计规则比较简单已经能够利用EDA工具完成自动化设计工作，而模拟集成电路设计还无法做到，同时由于自然中的信号都是模拟信号，必须通过模拟集成电路处理和模数转换操作后才能利用数字集成电路进行处理，所以模拟集成电路也是无法被取代的，因此模拟集成电路的自动化设计也成为了限制集成电路发展的瓶颈。

布线算法是集成电路自动化设计工具的核心，而布线算法又与布线模型的选择紧密相关，目前布线模型主要分为有网格布线模型和无网格布线模型，具体会在第二章进行介绍。鉴于模拟集成电路设计约束复杂的特点，在模拟集成电路设计布线模型中主要使用无网格布线模型，然而现有的无网格布线模型或者偏向于布线速度，但布线质量不够，或者布线质量好，布线速度太慢，所以对于模拟集成电路的无网格布线模型研究是有重要价值和现实意义的。

## 研究内容

基于以上背景和研究现状，本论文对无网格布线算法的优化进行研究与实现，在实验室提出的分裂图模型[[1](#_ENREF_1)]的基础之上进行优化，主要是对分裂块的构造和组织部分进行优化，使得模型在保证布线质量的同时进一步提高布线速度。

## 论文结构

本文基于实验室提出的分裂图模型进行研究，提出了一种新的分裂块构造算法，并修改了分裂块的组织方法，从而替换掉原来暴力遍历的方法，在保证布线质量的同时提高布线的速度。全文共分为7章，本章介绍了选题的背景与意义以及研究内容和论文结构；第2章主要介绍现有的一些无网格布线模型，以及实验室提出的分裂图模型；第3章介绍修改后算法的算法流程；第4章对修改的算法中使用到的重要思想和算法进行介绍；第5章是对提出的DTC算法的思想、流程和实现进行详细介绍，并对修改的角勾链结构进行介绍；第6章是实验测试的结果以及对结果的分析；第7章是对本次综合论文训练的总结与展望。

# 布线模型

## 布线模型概述

在集成电路设计布线算法中，布线模型起着重要的作用，也决定着布线速度的快慢和布线质量的好坏。目前按照布线区域的表示方法来划分，布线模型分为有网格布线模型和无网格布线模型。

有网格布线模型是将布线区域划分为具有相同大小的网格再进行布线，因此需要的数据结构以及相应的搜索算法比较简单，这种模型对于数字集成电路中线宽、线间距一致的情况比较适用；而无网格布线模型中布线区域被划分成不同大小的网格，适用于变线宽、变线间距的情况。

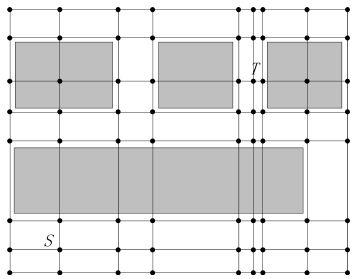
对于模拟集成电路而言，它的布线规则和布线约束不像数字集成电路那样简单，如存在变线宽、变线间距问题，所以在模拟集成电路中主要采用的是无网格布线模型。目前国内外也主要在研究这一方面，现在也已经有了不少的研究成果。下面介绍一下当前主要采用的一些无网格布线模型。

## 无网格布线模型

无网格布线模型根据对要布线区域的不同模型化方法来划分，可以分为两种：基于隐式连接图的布线模型和基于网块的布线模型。下面大致介绍一下这两种布线模型。

### 基于隐式连接图的布线模型

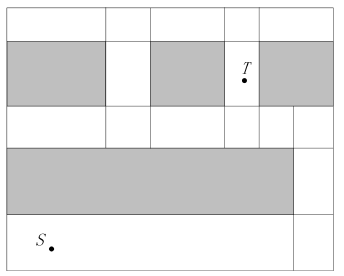
基于隐式连接图的布线模型使用的是零线宽模型，它将导线看作没有宽度的线条，然后将布线区域中的全部障碍根据“半线宽+线间距”的宽度进行扩展，并沿着扩展的障碍边界进行延伸直到与布线区域边界相交，这些经过特殊距离扩展后产生的延伸线提供了可能的待布线网，同时也构成了一个不均匀的网格图，如图2.1所示。由于没有显示地构造网格点，而是保存横向和纵向的延伸线，通过它们得到网格点的坐标，所以称为隐式连接图。在基于隐式连接图的布线模型中，常采用迷宫算法进行路径搜索。



* + - 1. 基于隐式连接图的布线模型

### 基于网块的布线模型

基于网块的布线模型将空白区域和障碍区域区分开来，各自单独形成一个个矩形区域，也称作“网块”。这样就形成了全部由空白区域或者全部由障碍构成的两种网块，如图2.2所示。在基于网块的布线模型中就是要在源点和目的点间找到一条由空白网块构成的路径，再通过回找操作确定具体的路径。



* + - 1. 基于网块的布线模型

## 实验室前期工作

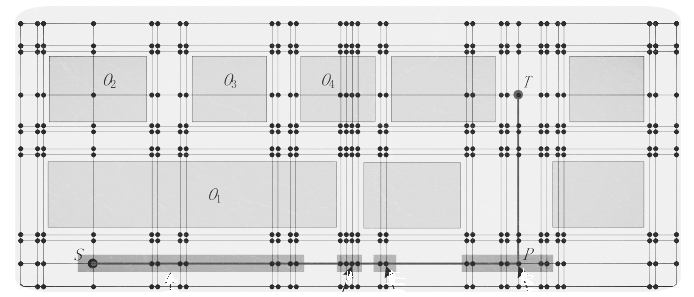
### 现有模型的不足

基于隐式连接图和基于网块的布线模型都能够满足部分布线的要求，但是都还存在着一定的缺陷和不足。基于网块的模拟集成电路布线模型数据结构复杂，同时在回找过程中由于不对齐等原因会导致拐点的质量无法保证。基于隐式连接图的布线模型是对障碍按照“半线宽+线间距”进行扩展，然后延长扩展后的障碍边界直到与布线区域的边界相交，从而构造出横竖相交的连接图，随着布线数量的逐渐增多，已布线网都被当做障碍进行处理，布线区域中的横竖线会大量增加，从而影响布线搜索的速度。但是基于隐式连接图布线模型采用零线宽模型，数据结构简单，回找过程也更加准确。模拟集成电路对布线的质量和准确性要求较高，所以主要采用的是基于隐式连接图的布线模型。

### 分裂图模型

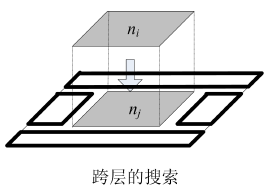
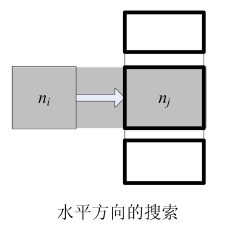
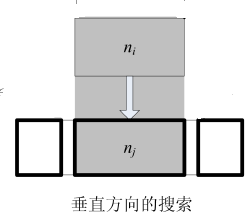
在基于隐式连接图布线模型的基础上，实验室提出了分裂图布线模型，这个模型以隐式连接图模型为基础的，如图2.3所示，考虑到搜索过程中采用的迷宫搜索算法或者A\*算法都是通过逐个网格点遍历搜索的方法，那么如果能够减少要搜索的点的数目也就能够加快布线的速度了。

分裂图布线模型正是出于这样一种考虑，图2.3中点如果具有相同的可布线方向，称它们具有相同的属性，将它们合并起来形成一个整体对于走线不会产生影响，同时也能减少需要遍历的次数，从而加快搜索速度，提高布线效率。



* + - 1. 分裂图模型

另一方面为了避免出现基于网块布线模型中拐点质量无法保证的问题，分裂图模型在搜索过程中增加了一个分裂操作，如图2.4所示，在两个搜索单元不对齐的情况下，通过分裂操作来进行处理，具体的分裂方式有如图所示的3种情况：水平方向分裂，竖直方向分裂，相邻层之间分裂。这就是分裂图模型的主要算法思想。考虑整个算法，可以发现主要的操作集中在分裂块的构造即将相同属性的点进行合并成块，分裂块的组织，分裂块的分裂以及搜索过程几个方面。



* + - 1. 分裂块分裂方式

### 分裂图模型中的问题

实验室分裂图模型现有算法在进行相同属性点合并构成分裂块操作中，采用的方法是记录每一个点的属性然后使用遍历操作来完成合并操作，同时记录每个点所属的分裂块。这样的操作构造速度比较慢。本论文主要针对这一部分进行优化和实现。

# 整体算法流程

## 算法流程

根据对实验室现有分裂图模型的了解及分析之后，本论文提出并实现了新的分裂块构造算法DTC，整个分裂图布线模型的算法基本流程图如图3.1所示：



* + - 1. 修改后算法流程

算法的基本流程是：

1. 首先从命令行读入运行参数，通过这些参数对布线器进行选择并设置一些布线环境；
2. 通过命令行参数选择STRouter布线器类进行布线。
3. 通过STRouter类调用STGridless无网格布线算法类进行具体的布线操作。
4. 其中在STGridless类中的gridless函数中，首先调用STLayout的initiate函数进行布局初始化，即完成布线参数设置及分裂块的构造、组织等操作。修改的DTC算法主要用在这一部分中。
5. 完成前面部分后再调用STMazeRouter类中的maze函数进行具体的搜索布线操作，返回得到的布线结果后，将布线结果输出。

从流程图中可以看到算法的主要步骤在选择布线器之后的具体算法、布线函数进行布线、布线开始的初始化以及搜索等部分。

## 算法重要部分详细介绍

### 布局初始化

布局初始化是整个算法比较重要的一个部分。在这一部分首先会根据前面的信息对布线类型进行设置；然后读入处理后的障碍信息；根据障碍信息对障碍进行扩展；然后将构成的点按照属性进行合并形成分裂块并设置分裂块的信息。

其中对障碍进行扩展部分是按照隐式连接图的模式，对障碍边界按照“半线宽+线间距”的距离进行扩展，然后对扩展的边界进行延伸直到与布线区域边界相交；同时从源点和目标点分别构造水平和垂直的线。这些线一起构成了待布线网。

由于线网上面的点具有的属性是不一样的，基于分裂图模型提出的原则是把相同属性的点合并成在一起构成块，在这一部分不再使用原来遍历的方法，而是采用本论文提出并实现的DTC算法对分裂块进行构造，在DTC算法中借助了PTR(Polygon-to-Rectangle)算法来将多边形转化为矩形，分裂块构造后使用修改的角勾链结构对构造的块进行组织。

### 搜索布线

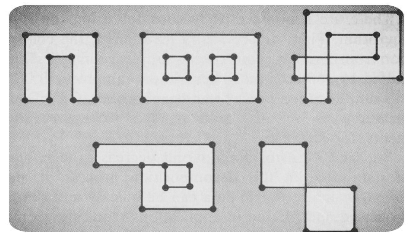
在初始化完成后就会进入到具体的搜索布线操作中，在这一部分采用的是A\*算法进行搜索。

1. 从源点所在的块开始，找到相邻的块，这一操作是通过修改后加入的角勾链结构的邻居查找算法完成的。
2. 对每个相邻的块都会有一个代价评估值，将这些相邻的块压入一个最小堆中，优先对代价评估值较小的块进行搜索。
3. 在对相邻块进行搜索时，如果存在不对齐的情况就会进行分裂操作，这一过程是在修改的角勾链结构中完成的。
4. 按照这样的方式递归的进行下去，直到找到目标点所在的块，或者遍历完成无法找到目标点所在的块。
5. 最后返回布线结果。

# **PTR算法和角勾链结构简介**

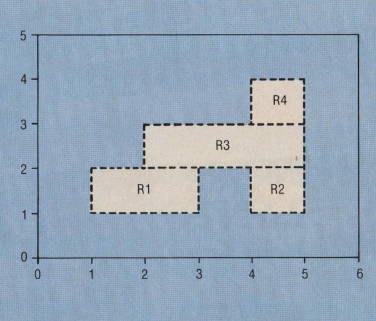
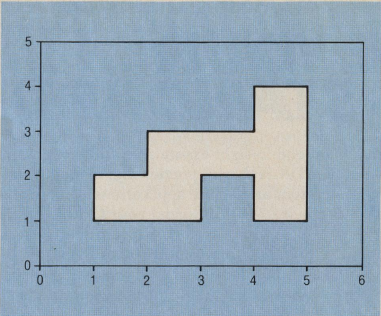
## PTR算法介绍

为了不采用遍历的算法而是直接根据障碍边界的限制得到分裂块，在实际实现过程中使用了PTR算法进行处理。PTR算法全称是A Polygon-to-Rectangle Conversion Algorithm即将多边形转化为矩形的算法，通过将描述多边形的数组传给该算法，能够得到一个个划分的矩形。该算法是一个迭代算法，每次通过该算法改变或者减少描述多边形的数组中的点，同时生成一个多边形。如图4.1所示，对于图中所示的情况该算法都可以进行处理。



* + - 1. 合法的待处理多边形

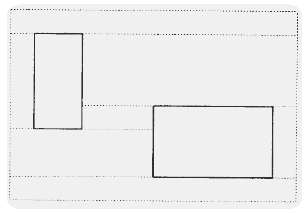
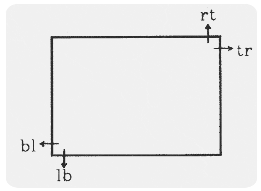
图4.2是经过该算法处理后的一个具体例子：



* + - 1. PTR算法处理前后

## 角勾链结构介绍

由于不再存储每个网格点的信息而是将每个分裂块当作一个处理的单元，所以需要一个相应的数据结构来对它们进行组织和处理，本论文借助了Corner-Stitching structure(角勾链结构)这种数据结构并根据实际问题进行修改。角勾链结构的主要思想是将区域内的障碍和空闲部分按照最大水平条(MHS)或者最大竖直条(MVS)的原则划分成一个个的块，每个块记录左下角的点的坐标同时在左下角和右上角各有两个指针，左下角的两个指针沿着左下角的点指向左边和下边，右上角的两个指针沿着右上角的点指向右边和上边，如图4.3所示。通过这样的一种形式整个区域都能被表示出来，同时基于这种结果能够实现很多比较高效的算法，如相邻块的查找、点的查找等，这也是采用这种数据结构的重要原因。



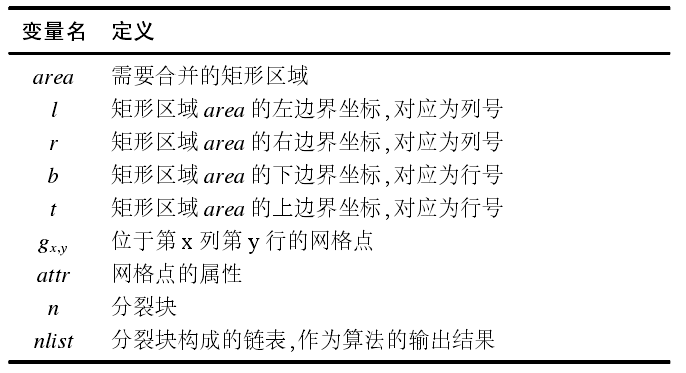
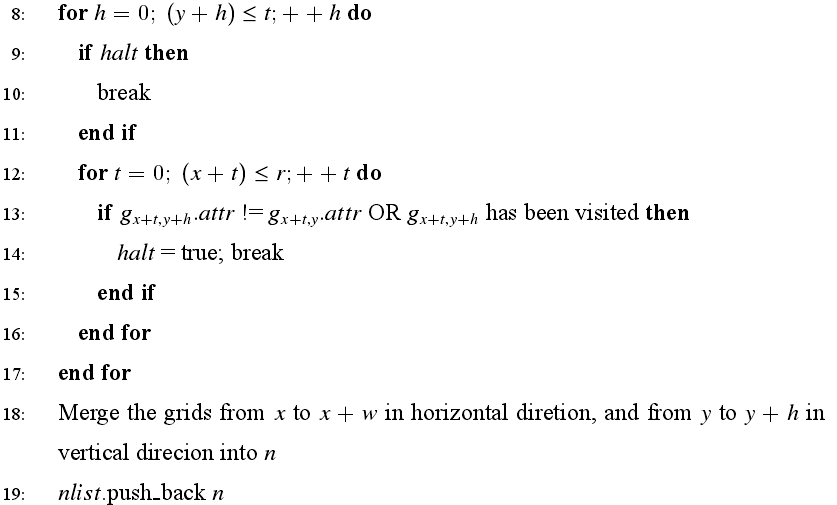
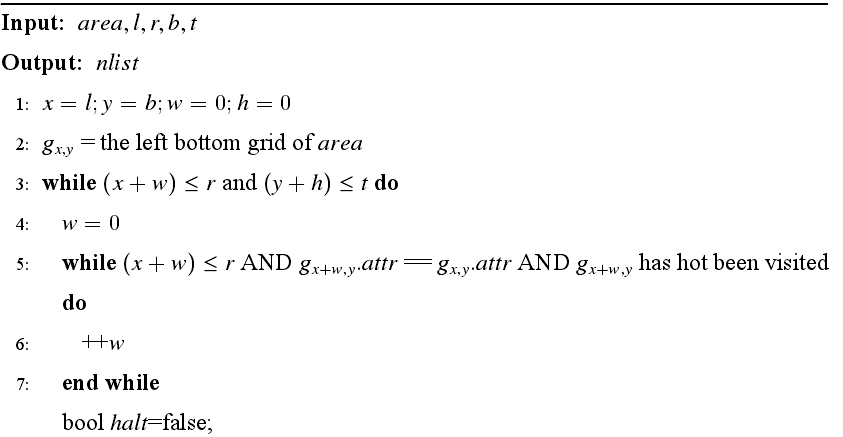
* + - 1. 角勾链结构及平面表示

# **DTC算法详细及修改的角勾链结构介绍**

## DTC算法详情

### 算法思想

在分裂块构造部分，原始程序采用的方法是给每个点赋予一个属性并记录下来，然后再对这些点进行遍历与合并，原始算法的伪代码如下：



从伪代码可以看出这个构造算法需要记录每个网格点的属性，而布线区域中网格点的数目是非常多的，因此该算法占用的内存空间会较大，同时对每个点进行多次遍历花费的时间也较多。

基于这一方面的考虑，本文提出了一个新的分裂块的构造算法——直接分裂块构造算法DTC(Direct Tile Construct)，算法基于对布线环境与点属性的以下几点分析：

1. 布局边界与障碍扩展后的边界上的点才存在布线方向限制；
2. 障碍扩展后的内部点不允许走线；
3. 障碍的数目相对于网格点来说是很少的；

综合以上三点考虑，新算法大致思想如下，图5.1是部分处理结果：

1. 首先得到障碍的原始坐标；
2. 根据布线约束对障碍进行扩展，得到扩展后障碍集合；
3. 在障碍的每条边上构造一些辅助点；
4. 根据辅助点对属性不同的点分别进行处理；
5. 根据处理结果构造分裂块；
6. 将分裂块按照角勾链结构进行组织。



* + - 1. DTC算法处理

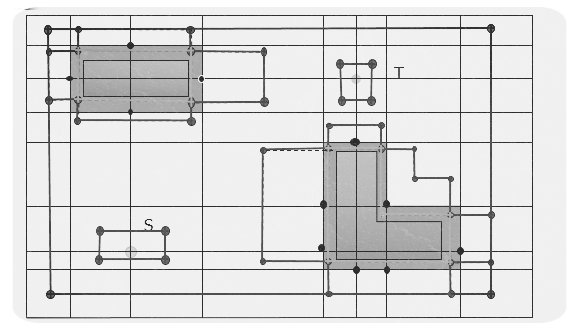
### 详细过程

基于上一部分中的思想，在原有分裂图模型程序的基础上进行了修改，详细步骤如下图所示，分为6个主要步骤：

* + - 1. 原始障碍点及障碍扩展

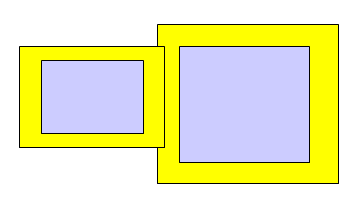
1. 按照逆时针方向取得原始障碍(未经过处理程序)的顶点集合；
2. 对获得的障碍进行线宽、线间距要求的扩展；
3. 判断障碍经过扩展后是否存在相交的情况,如果两个障碍扩展后相交，将相交的障碍合并为一个障碍;
4. 分别对布线边界和障碍扩展边界进行处理，得到需要的关键点，如图5.3所示;
5. 调用PTR程序对得到的图中关键点进行处理，从而得到障碍外部满足要求的分裂块;
6. 再对每个障碍扩展后的内部单独进行处理，得到一个个矩形分裂块;
7. 按照类似角勾链的方式对得到分裂块进行组织，并更新它们之间的相互关系。



* + - 1. PTR算法处理前

算法中还涉及到障碍扩展后相交的检测与合并，关键点的设置等问题。

首先是障碍扩展后相交的检测与合并，基于后面关键点设置的需要，对于扩展后相交的障碍需要合并成一个多边形。在对障碍进行扩展之后，本论文通过对障碍多边形的每条边逐一比较判断边是否相交，从而判断障碍是否相交。对于相交的障碍，通过程序将它们合并为一个多边形，并更新障碍集合，如图5.4障碍相交的检测与合并。



* + - 1. 障碍相交的检测与合并

在对障碍进行扩展及相交问题处理之后需要对每个障碍及源点、目标点、布线区域边界设置关键点，所以具体的设置方法分为3中，如图5.5 关键点设置所示：

* + - 1. 关键点设置

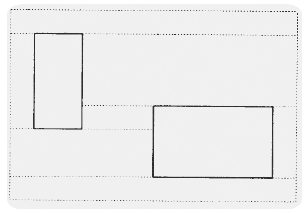
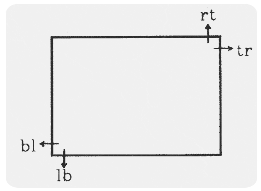
1. 对于布线区域边界，由于布线边界线上的属性与内部属性不同，所以需要单独划分出来进行处理，具体的是取边界与最近一条相邻边靠近边界的1/10的位置，横纵坐标都按相同方式处理。
2. 对于源点和目标点，由于需要单独划分为一个块，所以在它们四周取4靠近它们的点，距离是与它们最近边距离的1/10。
3. 对于障碍部分，障碍的边界与障碍的内部同障碍外部都是不一样的，所以需要进行处理。对障碍的每个顶点，判断它前后边上是否存在交点，如果存在就在顶点外部靠近顶点方向与外部最近边距离1/10的位置设置关键点，如果不存在就在内部靠近顶点方向与内部最近边距离1/10的位置设置关键点。
4. 由于障碍可能为多边形(非矩形)，所以障碍的内部还需要划分，因此需要为每个障碍添加内部关键点，即在障碍内部靠近顶点方向与内部最近边距离1/10的位置设置内部关键点。

### 算法伪代码

|  |
| --- |
| GetOriObstacles(); //获取原始障碍点  for obstacle m in obstacles{  for vertex n in obstacle m{  expand(n)  }  } //按照线宽线间距的要求扩展障碍  for obstacle m in obstacles{  for expand vertex n in obstacle m{  get\_outer\_point(n);  get\_inner\_point(n);  }  } //对每个扩展顶点得到外部和内部辅助点  PTR outer\_polygon; //根据外部辅助点调用PTR算法  for obstacle m in obstacles{  PTR inner\_polygon; //根据每个障碍内部辅助点处理每个障碍  }  for rectangle m in PTR\_results{  construct split tile;  } |

## 修改后角勾链结构介绍

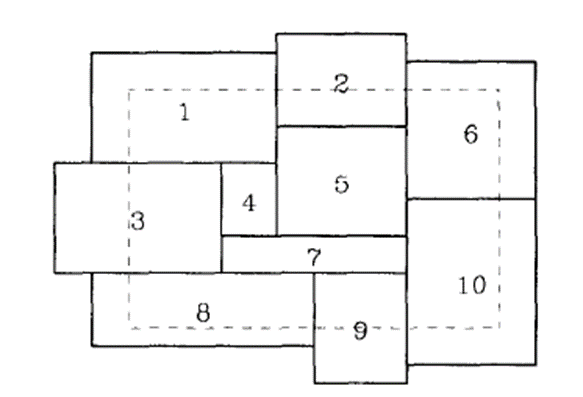
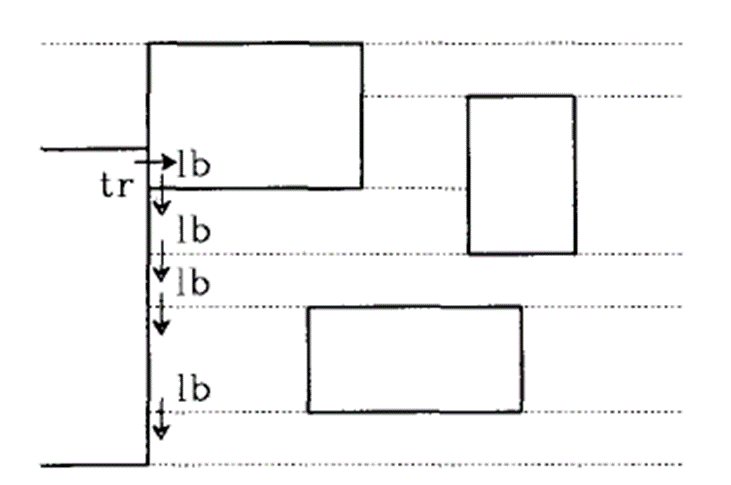
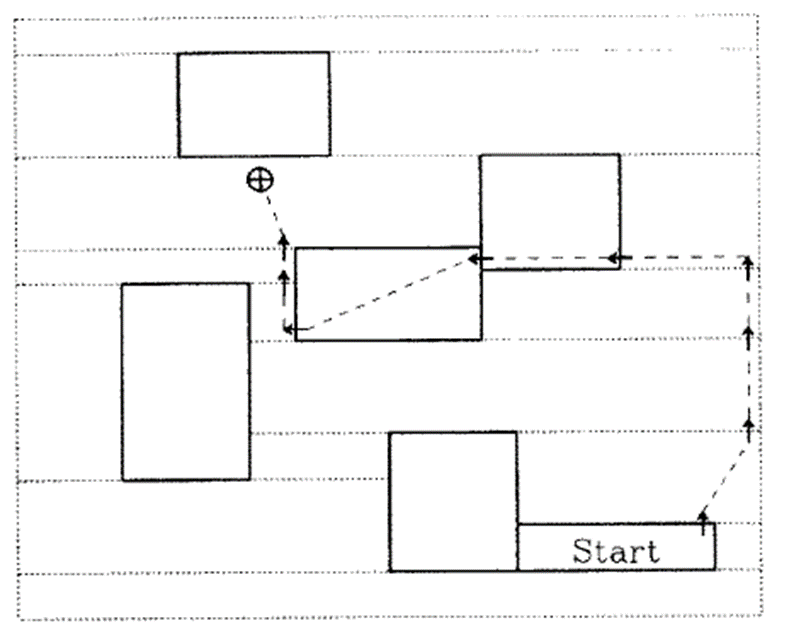
### 角勾链结构的修改



* + - 1. 角勾链基本结构

1. 使用DTC算法之后，不需要存储每个网格点的信息
2. 无法使用原有的每个点记录信息来判断点属于哪个块
3. 以每个块作为基本单元处理
4. 根据具体情况进行角勾链形式的修改和实现，由整个图覆盖到网格点覆盖

### 修改的角勾链结构中实现的算法



* + - 1. 修改的角勾链结构实现的部分算法

主要需要用到的是相邻块的查找和块的分裂操作，由于网格点不再存储信息，所以无法通过网格点进行相邻块的查找操作，因此在使用角勾链结构对分裂块进行组织后需要针对这一结构实现相邻块的查找操作，具体的算法如下。

以查找与右边界相邻的块为例：

1. 首先使用点查找算法确定块右上角点的右边一个点属于哪个块
2. 将找到的块加入到相邻块列表中，然后取这个块左下角勾链得到一个块
3. 判断这个块的左边是否和原块的右边相交，如果是加入到相邻块列表，并重复步骤2)和3），如果不是算法结束

此外，由于分裂图模型中还存在着一个重要的分裂操作，所以也需要在修改的角勾链结构中进行实现，具体的操作过程为：

1. 找到要分裂的位置
2. 构造一个新的分裂块
3. 更新原分裂块、新分裂块以及与原分裂块存在勾链关系的块，使得它们符合角勾链结构
4. 返回新的分裂块

## 算法分析

通过对算法结构和流程进行分析，可以得出算法的时间复杂度：

经过对DTC算法流程和伪代码的分析，可以看到DTC算法需要对每个障碍的顶点进行遍历处理，这也是算法的主要操作步骤，故算法的时间复杂度为O(m\*n)，其中m为障碍数目，n为障碍的平均顶点数。

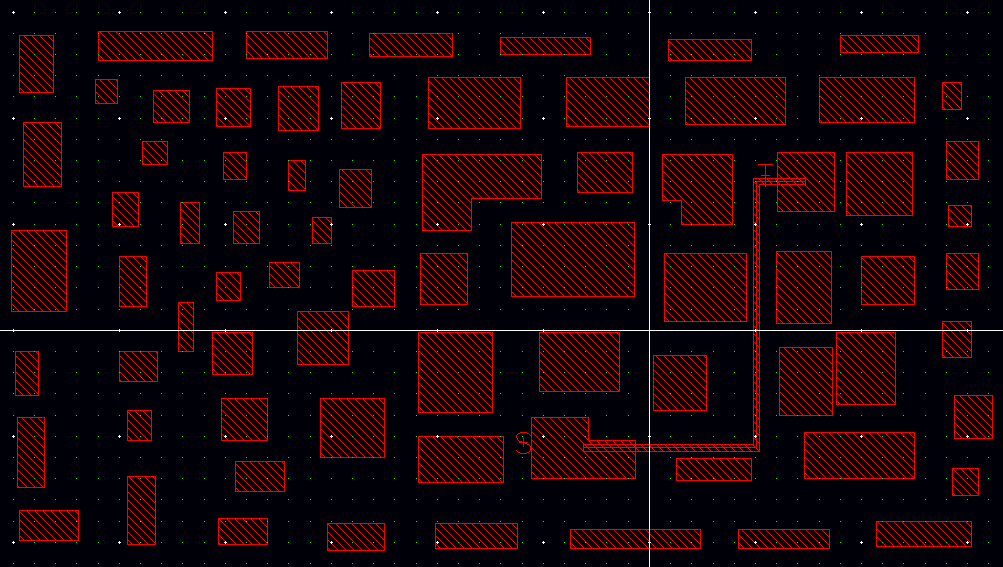
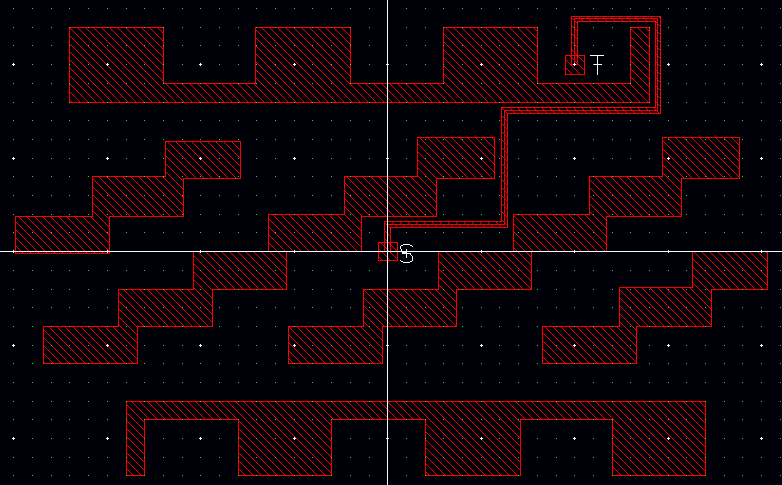
而原算法是通过遍历网格点的方法来构造分裂块，算法复杂度为O(n\*n)，其中n为网格点数目。

# **实验结果**

## 实验测试结果

为了证明修改后算法的有效性，本论文对原算法和修改后的算法在运行时间和内存占用方面进行了测试，部分测例如图6.1 所示。测试结果如表6.1所示。

### 部分测例



* + - 1. 部分测例

### 测试结果

* + - * 1. 测试结果

|  |  |  |
| --- | --- | --- |
|  | 时间(s) | |
| 修改前 | 修改后 |
| Test1 | 0.055992 | 0.052992 |
| Test2 | 0.265959 | 0.088986 |
| Test3 | 0.294955 | 0.111983 |
| Test4 | 0.560914 | 0.145978 |
| Test5 | 0.693894 | 0.271959 |

## 实验结果分析

从测试结果我们可以看到改进后的算法在运行时间都有了较大的改善。

从5.3算法分析部分，可以看出DTC算法的复杂度大致为O(m\*n)，其中m为障碍数目，n为障碍的平均顶点数，而原算法的复杂度为O(n\*n),其中n为网格点数目，而网格点的数目相对于障碍数目来说是很多的，所以DTC算法相对于原算法在障碍数和网格点数多的情况下时间方面是有优势的，表中实验结果也进一步证明了这一点。

# **总结与展望**

## 毕设总结

由于本科课程对于具体问题的接触比较少，对理论知的讲解学习相对较多，本人在电路设计方面了解的也比较少，导致开始阶段不太适应。通过导师的指导与对相关知识的学习，毕业设计相关的内容也顺利地进行。

通过本次毕业设计训练，了解了集成电路设计特别是模拟集成电路方面的知识，对当前模拟集成电路设计的现状和遇到的问题有了一定的认识。同时使自己对于科研工作的流程有了具体的了解，培养自己对具体问题的处理能力，也使自己的沟通交流能力有了一定的提高。

由于知识和能力的限制，虽然导师安排的内容基本完成，但是做得比较突出的地方较少，这也是在以后进入工作岗位自己应该提高的地方。

## 工作展望

本论文对提出的问题基本完成，但由于本人在集成电路设计方面没有基础，对于相关的知识了解不多，在考虑这一问题方面还不够深入，目前实现的算法还存在一些不完善的地方，如对布线的环境有所要求等，希望在以后有机会能够针对这些问题进行完善和改进。

插图索引

图2.1 基于隐式连接图的布线模型 4

图2.2 基于网块的布线模型 4

图2.3 分裂图模型 5

图2.4 分裂块分裂方式 6

图3.5 修改后算法流程 7

图4.6 合法的待处理多边形 9

图4.7 PTR算法处理前后 10

图4.8 角勾链结构及平面表示 10

图5.9 DTC算法处理 13

图5.10 原始障碍点及障碍扩展 13

图5.11 PTR算法处理前 14

图5.12 障碍相交的检测与合并 14

图5.13 关键点设置 15

图5.14 角勾链基本结构 16

图5.15 修改的角勾链结构实现的部分算法 17

图6.16 部分测例 19

表格索引

表6.1 测试结果 20

参考文献

[1] YANG F, YAO H, ZHOU Q, et al. SIAR: splitting-graph-based interactive analog router; proceedings of the Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI, F, 2011 [C]. ACM.

[2] DIVEKAR D A, DOWELL R I. Corner stitching: a data-structuring technique for VLSI layout tools [J]. IEEE Transactions on Computer-Aided Design, 1984, 3(1): 87.

[3] Dion J. Contour: A Tile-based Gridless Router, Mar. 1995, Digital Western Research Laboratory[R]. research Report 95-3.

[4] CONG J, FANG J, KHOO K-Y. DUNE: A multi-layer gridless routing system with wire planning; proceedings of the Proceedings of the 2000 international symposium on Physical design, F, 2000 [C]. ACM.

[5] LI Y-L, CHEN H-Y, LIN C-T. NEMO: A new implicit-connection-graph-based gridless router with multilayer planes and pseudo tile propagation [J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2007, 26(4): 705-18.

[6] YANG F, CAI Y, ZHOU Q, et al. SAT based multi-net rip-up-and-reroute for manufacturing hotspot removal; proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010, F, 2010 [C]. IEEE.

致 谢

衷心感谢姚海龙老师在本次毕业设计及综合论文训练中对我的帮助和悉心指导。姚海龙老师认真负责的研究态度以及严谨的作风都让我受益匪浅。

同时感谢实验室的师姐们在我毕设过程中对我的帮助，感谢实验室前期的工作。

最后感谢班里同学在我完成综合论文训练阶段给予我的帮助。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：\_\_\_\_\_\_\_\_\_\_\_\_\_ 日 期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

附录A 书面翻译

角勾链：一种用于VLSI布局工具中的数据结构技术

A1摘要——角勾链是一种用来表示二维矩形对象的技术。它尤其非常适合交互式的VLSI布局编辑系统。这种数据结构有两个重要的特点：第一，空白的空间被明确地表示出来；第二，矩形区域的角被勾连起来就像一张拼成的被子。这种组织方式带来了搜索、新建、删除、扩展和压实操作的快速算法。这些算法会在一个简单的VLSI电路模型中展示出来，同时也会讨论这个结构的存储需求。角勾链已经在一个正在工作的布局编辑器中实现。初步测试表明它需要的存储空间是可能的最简单表示需要空间的三倍。

A2引言

用于集成电路的交互式布局工具对它内部的数据结构有特殊的要求。这个数据结构必须能够处理大量的信息（在当前的布局中一百五十万或者更多的几何元素），同时要给设计者提供快速的响应。随着设计复杂性的增加，布局工具必须在像布线和验证等方面给设计者提供更加有力的辅助。为了支持这些交互式工具，相关的数据结构必须提供快速的几何操作，比如为扩展和压实操作找到相邻对象，为布线找到空闲空间。这种数据结构也必须允许快速增量式修改以便它们能够用于交互系统中。

角勾链就是一个能满足这些要求的数据结构技术。正像这里描述的，它设计时就被限制带有曼哈顿特征（只有水平和垂直边）；但是在那样的框架下，它提供了许多有效的操作，例如邻居查找、扩展、压实和路线查找。这些操作的算法只依赖局部信息（操作紧邻的对象）。它们期望的运行时间通常是线性于相邻对象的数目；在最坏的情况下（实际布局中不太可能出现）运行时间可能正比于整体的设计大小或者正比于附近对象的数目和设计大小。角勾链在处理有差不多相同大小的对象时特别有效。然而，在有不同的特征尺寸时它也能很好的工作，例如在一个单元包含许多子单元并且许多小的线把他们连在一起的层次式布局中。

角勾链允许对数据库的快速修改，因为更新时仅仅用到了局部的信息。大多数现存能够提高有效的布线、压实等操作的系统都无法提供代价不高的更新：对数据库的很小的改动可能导致大量的重新计算。角勾链的有效操作和简单更新的结合意味着许多以前只在批量处理模式有效的工具，现在能够嵌入到交互式系统中了。

A3角勾链

角勾链在考虑了其他机制缺点之后被提出，并且有两个区别于其他机制的特点。第一个重要的特点是所有的空间，空闲和被占用的都在这个数据库中明确的表示出来。第二个特点是通过一种新颖的方式把对象的角连在一起。这些角勾链允许数据库简单的修改，并且带来了各种操作的有效实现。

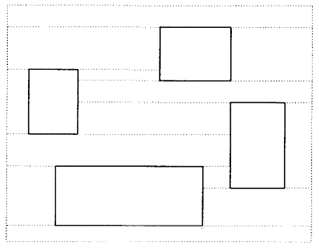


图3

图3 展示了四个对象用角勾链的方式表示出来。这张照片类似于用两种类型的矩形块，空的和被占用的，构成的马赛克。这些块必须是边与坐标轴平行的矩形。块包含它们的左边和下边，不包含上边和右边。这样平面上每一个点都被一个块明确的表示了。整个平面覆盖了x和y方向的负无穷到正无穷（实际中，最大的可表示的正负数用来表示无穷大）。覆盖到无穷大是通过扩展最外层的空闲

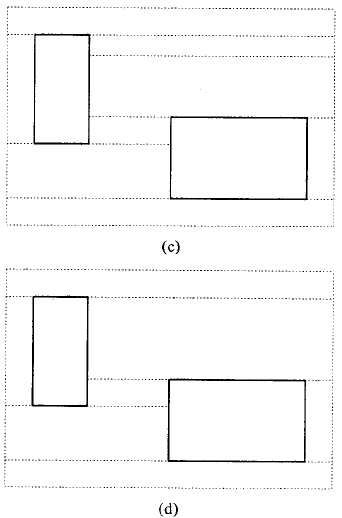
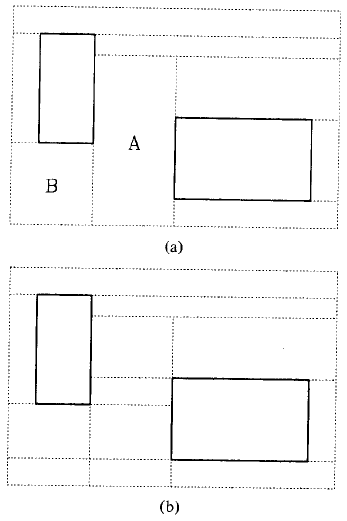


图4

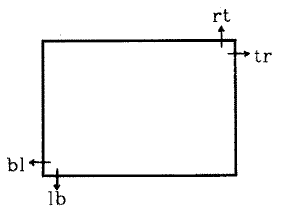
块达到的，而不需要额外的块。

空闲的块按照最大水平条的方式组织。这就意味着没有空闲块在与它紧邻的左边或者右边有其他的空闲块。在修改数据库时，水平相邻的空闲块必须被分

割成更短的块然后合并成最大的条，如图4 所示。在保证空闲块已经尽可能宽之后，有相同的水平宽度的垂直相邻块合并到一起。这种空闲块的表示对于VLSI布局和设计者来说没有意义，在实际系统中甚至都看不到。然而，最大水平条的表示是这些工具时间和空间有效性的关键，我们在第5和第6部分会看到。在它其他特性中，最大水平条的表示是独一无二的：每种不同摆放的被占用块中的空间有且只有一种分解方法。

块被它们角上称为角勾链的指针集合连接起来。每一个块包含四个勾链，左下角和右上角各两个，如插图5 。由于在四个方向的每一个上只有一个指针，勾链提供了一种等价于邻居指针的一种排序形式。开始的时候，使用了8个勾链，四个角每个角两个，但是四个被证明对于这里展示的算法是足够的。这些特殊的四个勾链的选择是重要的。

图5



角勾链形式表示有几个吸引人的特点，在下面的部分会进行说明。首先，这种机制将水平和垂直指针综合到一个结构中。空闲块提供了一种在垂直和水平信息之间定位的形式，并且在电路被修改后能够很容易的保持指针的更新。由于空闲块可能在尺寸上不一样（与固定大小的块相反），这种结构能够很自然地适应块在尺寸上的变化。空间的最大水平条表示使得空闲块的数量和算法的复杂度上限很简单。所有的块拥有同样数量的指针指向其他快，因此他们占用相同字节的存储空间；这简化了数据库的管理并且减少了算法中的常数因子。

A4算法

这个部分展示了操作块和角勾链的算法。所有算法中最重要的属性是它们的局部性：每个算法仅仅只依赖于操作附近的信息。没有一个算法的期望运行时间比线性于被影响区域中块的数量还差。最差的情况下能看到算法需要线性甚至平方于整体布局的大小，但是实际情况下（特别是对于密密麻麻的VLSI布局）它们的运行时间小于或者独立于布局大小。

在讨论算法的性能时，角勾链提供了一个很好的评价单位。算法的复杂度可以依据必须遍历的角勾链数目（或者选择必须访问的块的数目）和/或必须被修改的角勾链数目。

A4.1点查找

角勾链促进了几种不同的搜索。其中最常见的一种操作是查找给出的点(x, y)位于哪个块。图 6 说明了用角勾链怎么实现它。这个算法从数据库中任意给定的块开始在x和y方向迭代：

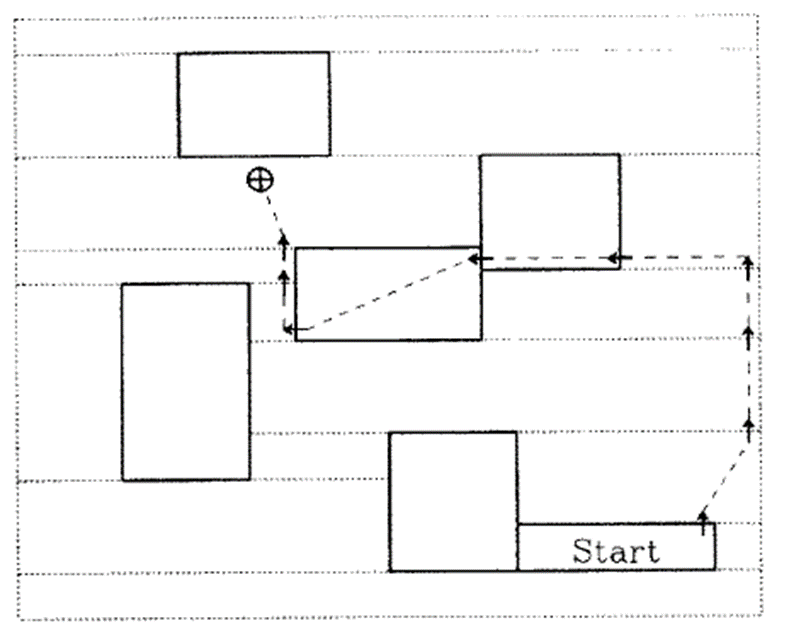


图6 点查找算法

1） 首先向上或者下移动，使用右上(rt)和左下(lb)勾链，直到找到一个块，它的垂直范围包含期望的点；

2） 然后向左或者右移动，使用tr和lb勾链，直到找到一个块，它的水平范围包含期望的点；

3）由于水平的动作可能引入了垂直方向的错位，所以步骤1)和2)可能需要迭代几次来找到包含这个点的块。网块的凸性保证了算法的收敛性。

在最坏的情况下，这个算法可能需要搜索整个结构中的每一个块（例如如果结构中所有的块都在一行或者一列这种情况就发生了）。幸运的是，平均情况下的表现比最坏情况好得多。如果有总数为N并且有相对均匀的尺寸时，平均情况下N0.5个块会被遍历。对于包含上百万网块的布局(典型的当前VLSI布局的完全扩展的掩码集)，这意味着几千个块会被遍历。

在交互系统中，有一个简单的方式来减少点查找的时间花费：用一个指针指向设计者工作的近似区域中的任意一个块。在编辑一个大的设计时，设计者的注意力通常集中在设计中的一小片上(例如在一个图像设备上能够看得比较舒服的一小片)。如果这个区域中的一个块被记下来作为参考，那么搜索时间仅仅依赖于屏幕中的块数目，而不是整个设计有多大。

点查找算法说明了一个大多数算法中常见的一个特点：错位。在水平搜索时，很可能丢失垂直方向的定位，所以算法必须迭代水平和垂直动作。图6 就是一个例子。通常情况下，大网块会导致算法在离它们感兴趣的区域很远的地方任意徜徉，当这种情况发生时，算法必须遍历勾链回到感兴趣的区域。极端的错位导致许多算法的行为极差。幸运的是，严重的错位在密密麻麻的设计中不太可能发生。

A4.2邻居查找

另一个常见的搜索操作是邻居查找：找到与给定块的一条边接触的所有块。邻居查找对于设计规则检查、压实，电路提取和描绘网络连接都很有用。图7 说明了如何找到与给定块右边相接触的所有块：

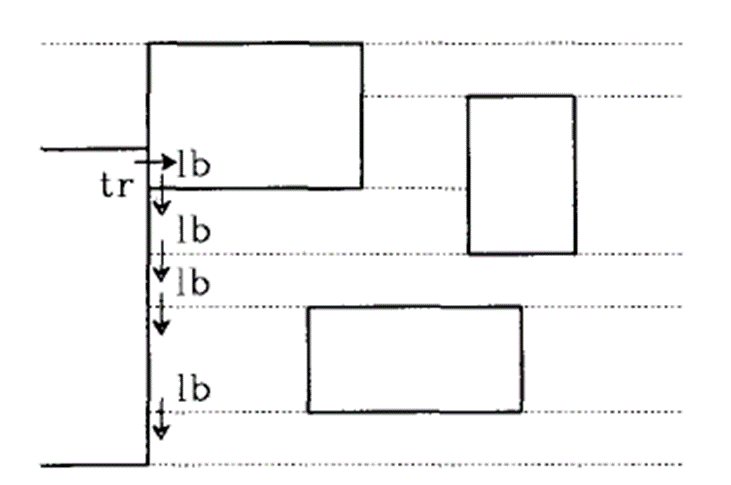


图 7

1. 随着起始块的tr角勾链找到它右边最上方的邻居；
2. 然后跟着lb角勾链向下直到找到所有邻居(最后一个邻居是首先遇到的较小的y坐标比起始块的较小y坐标小或者相等的块)

类似的算法可以设计出来搜索其他的每条边。搜索的时间线性于邻居的数目。

A4.3区域搜索

第三种形式的搜索是来看看在给定区域中是否有被占用的块。它可以借助于角勾链用下面的方法实现(见图 8)：

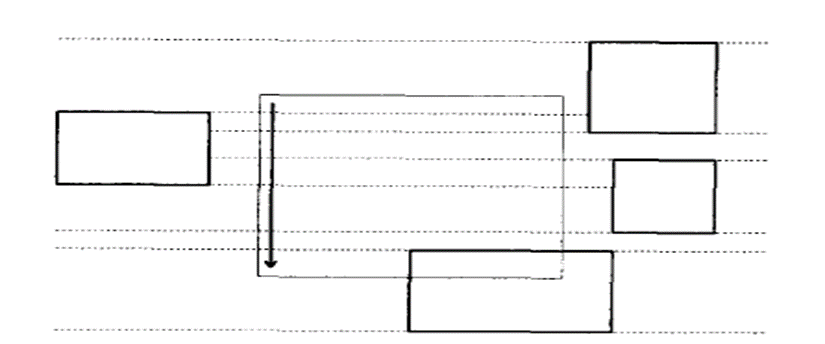


图 8

1. 使用点查找算法来找到包含感兴趣区域左上角的块
2. 检查这个块是否被占用。如果不是，它一定是个空块。检查它的右边是否在感兴趣的区域中。如果是，它就是被占用块的边。
3. 如果在步骤2)中找到了被占用的块，那么搜索就完成了。如果没有找到，就向下移动到与感兴趣区域左边接触的下一个块。这个操作可以通过调用点查找算法或者通过遍历lb角勾链向下然后访问tr角勾链向右直到找到期望的块的方法能够完成。
4. 重复步骤2)和3)直到找到一个被占用的块或者到达感兴趣区域的底部。

和其他操作一样，这个操作必须的时间仅仅依赖于局部特征：感兴趣区域内部和周围的块的数目。算法的代价可以通过计算必须访问的角勾链数目来评价。算法中迭代的次数与区域的高度成比例(和通常一样假设比较统一的尺寸分布)。在每次迭代中，步骤2)遍历一次角勾链是必须的。此外，步骤3)平均情况下会导致1/2个块的错位。因此总的运行时间线性于搜索区域的高度，并且完全不依赖于搜索区域的宽度。像图9)显示的一个最坏情况下，错位会导致运行时间与布局中所有的块数目成比例。

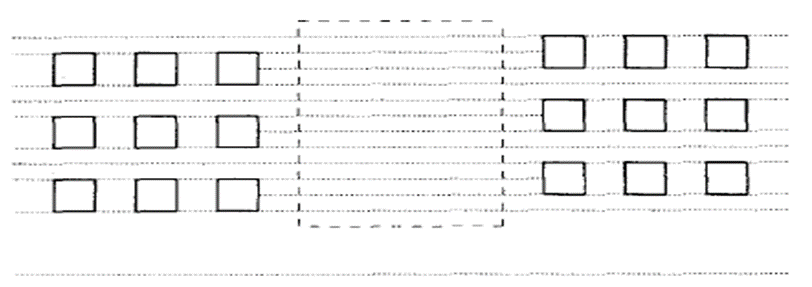


图 9

A4.4区域的定向枚举

在A4.3中的算法是判断在一个区域中是否有被占用的块。然而，对于很多应用如压实和布局规则检查，统计给定区域中所有的块数目是很有用的，也就是每个块访问一次。而且，在一个特定方向上进行这个操作通常很有用。例如，在一个从左到右的压实操作中，一个块左边的所有块处理完后再处理它本身是很重要的。这一部分展示了这样一个算法，每一个块只有在它上方和左边的所有块被访问了之后才能被访问。我称这种枚举为定向枚举。角勾链使它成为一个线性时间操作。图10 作为一个例子说明了枚举的顺序。

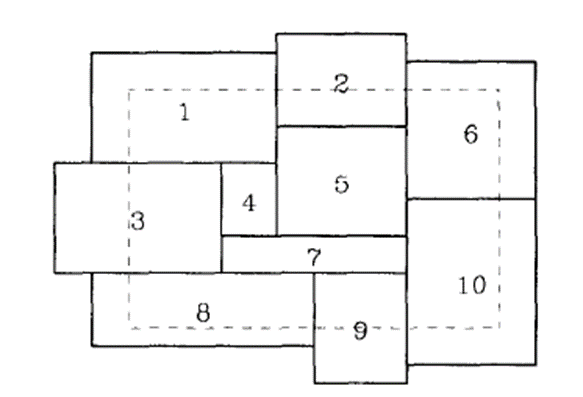


图 10

1. 和区域搜索算法一样，使用点查找算法来找到感兴趣区域左上角的点在哪个块中。然后使用区域搜索中相同的技术沿着左边向下找到所有的块。
2. 对步骤1)中找到的每个块，使用R1)到R5)行给出的R程序递归的枚举。

R1) 枚举块(这个操作通常会涉及到特定应用程序的处理)。

R2) 如果这个块的右边缘在搜索区域之外，就从R程序中返回。

R3)否则，使用邻居查找算法来找到与当前块右边缘接触并且与搜索区域相交的所有块。

R4)对于这些相邻块中的每一个，如果它的左下角与当前块接触，就递归的调用R来枚举邻居(例如图10 中，当前块是1时，块2是它的邻居)。

R5)或者，如果搜索区域的底边同时穿过了当前块和相邻块，就递归的调用R程序来枚举相邻块(在图10 中，当前块是8，相邻块是9就是这种情况)。

定向枚举算法的期望运行时间与搜索区域中的块数目成线性关系。下面的理由能够说明这一点。步骤R4)和R5)的检查保证了每个块都只被枚举一次。然而，为了满足步骤R4)或者R5)的检查，一个块可能会被检查几次：对于每个块，与它左边接触的块都会被检查一次。因此算法总的期望运行时间与搜索区域临近块的数目成比例。

最坏的情况下，区域定向枚举可能会要求检查电路中每一个块。在网块远远伸出被枚举区域的上边缘这种情况就发生了：在步骤R3)中它们所有的邻居都会被枚举，即使它们中的大多数不和感兴趣的区域相交。

定向枚举这个算法不依赖于空闲块是最大水平条的事实。实际上，它甚至不区分空闲块和被占用的块。能够设计一个类似的算法来颠倒枚举的方向(从右下到左上)。但是，从左下到右上或者从右上到左下的操作算法的重新编码可能非常困难(这是因为块没有从右下或者左上发出的角勾链)。

A4.5 新建块

创建一个新的被占用的块第一步是检查在希望新建块的区域内是否已经存在被占用的块。区域搜索算法能够检查这个问题。第二步是将新建的块插入到数据结构中去，像图11 中那样剪切合并空闲块并且更新角勾链。插入算法如下：

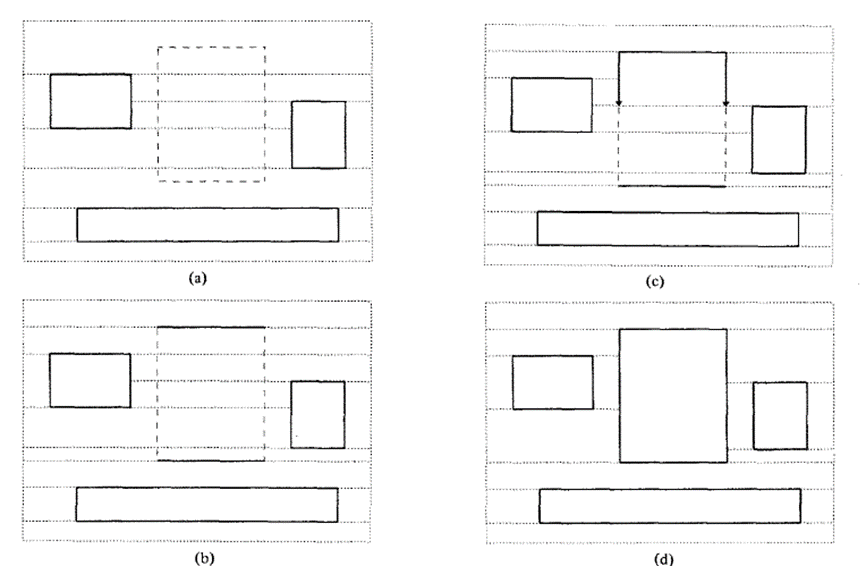


图 11

1. 找到包含新建块上边缘的空闲块(由于水平条的性质，一个空闲块就包含了整个上边缘)。
2. 沿着一条水平线将顶部的空闲块分成完全在新建块上方的一部分与和新建块交叠的一部分。更新角勾链链接新建块。
3. 找到包含新建块下边缘的空闲块，用同样的方式分割它，并更新它周围的角勾链。
4. 和区域搜素算法一样，沿着新建块区域的左边一直向下。沿着这个边缘的每个块肯定都是一个跨越了新建块整个宽度的空闲块。将空闲块分成新建块左边的一部分，右边的一部分，完全在新建块中的一部分。这种分割可能会使左右两边剩下的部分能够在垂直方向上与它们上方的块合并：合并一切可能的地方。最后，合并中间的空闲块构造被占用块。每次分裂与合并都需要更新附近块的角勾链。

新建块算法的速度取决于穿过新建块区域的空闲块分割与合并操作的花费。空闲块的数目取决于新建块左右阴影的被占用块数目。可以构造情况使得空闲块数目任意多，但是实际上，期望的数目是和新建块与周围块的相对高度成比例的。平均情况下块的分割与合并的花费是常数的，对于非常大的块，花费与块的周长成比例。这就意味着在最坏的可能情况下，新建一个块的花费可能与布局中所有块的数目成比例（见图 12）。平均情况下，如果新建块与它周围块的尺寸差不多，运行时间是常数；如果新建块比周围块大得多，运行时间与新建块的高度成比例而与它的宽度无关。

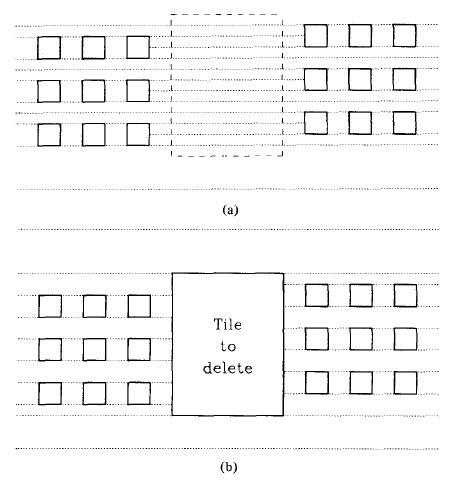


图 12

A4.6 删除块

块的删除操作是很复杂，因为需要分割和合并空闲块来维护水平条的表示。下面的算法对于绕着被删除块的顺时针方向是有用的，要被删除的块也成为死块。看图 13 的例子。

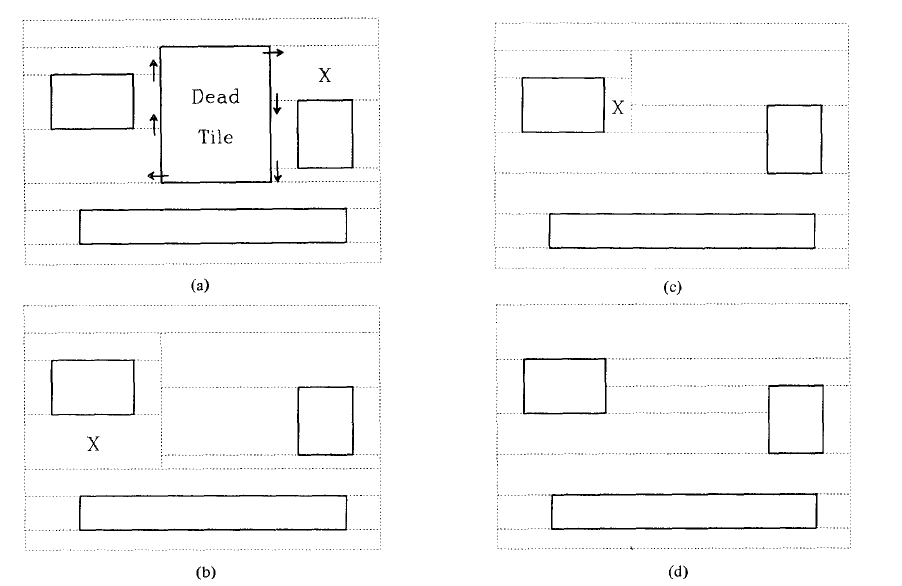


图 13

1. 将死块的类型改为空闲。
2. 使用邻居查找算法从顶部到底部搜索与死块右边接触的所有块。
3. 对于步骤2)中找到的所有空闲块，或者分割邻居、或者分割死块、或者两者都分割，使得两个块有相同的垂直跨度，然后在水平方向合并它们。
4. 当到达原始死块的底部时，沿着原始死块的左边缘向上扫描找到原始死块左边的所有邻居。
5. 对于步骤4)中找到的所有空闲块，合并空闲块与原始死块剩余部分。将当前空闲块看作步骤2)-3)中的死块，重复步骤2)-3)。
6. 在步骤5)中做垂直合并也是必须的。在步骤5)每次水平合并之后，检查合并结果的块能够和它上方与下方的块合并，如果可以就合并。

和其他算法一样，删除算法在不好的情况下也可能需要大量的时间，例如，图14 展示了一种情况，布局中每个块的角勾链都要检查和修改，所以运行

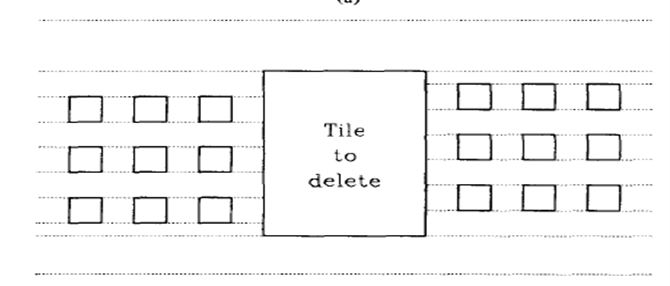
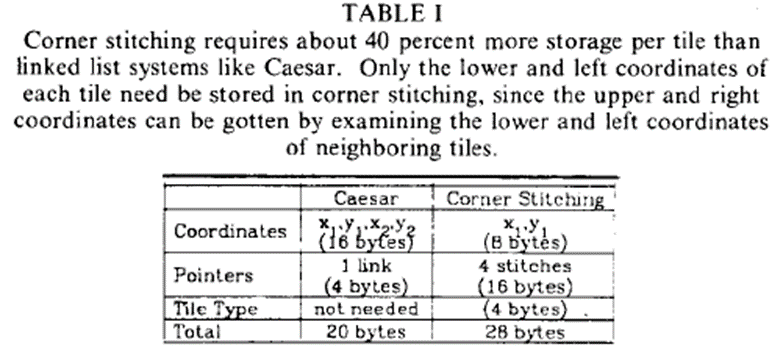


图 14

时间与整个布局尺寸成比例。然而，这种情况在集成电路中几乎不可能出现。如果要删除块和周围块的尺寸分布大致相同，需要分割合并的数目就是常数，运行时间也就是常数。当删除一个很大的块时，运行时间和这个块左右邻居的数目成比例，也与要删除块的高度成比例。

A5空间要求

由于VLSI设计巨大的规模，VLSI电脑辅助设计工具要想高效那么它使用的数据结构必须在空间上高效。例如，在Caesar中分层表示的45000个晶体管的芯片需要大约1.5\*106字节的主存。角勾链比Caesar系统要求在数据结构中存储更多信息。表I 比较了角勾链和链表形式的Caesar。角勾链比Caesar三倍多的指针，还有一个类型域(在链表系统中给定表中的所有块类型相同)。角勾链通过只存储每个块左下角的坐标来保存空间，而不是四个坐标点：右上坐标可以通过相邻块的左下坐标得到。所以，角勾链形式的块要比Caesar块大40%。此外，由于角勾链把空闲块也表示出来，所以它比其他系统有更多的块。如果有许多空闲块，那么角勾链实际需要更多的空间。而且，大多数算法依赖于一个区域内的所有块数目，包括空闲块和被占用块；如果有许多空闲块的话，算法将不高效。



表I

A6总结

角勾链对于表示几何数据是一个非常有力的技术。它的两个最重要的特点a)它把空闲块明确的表示出来，b)它把不同类型的块通过它们的角连接在一起。这两个特点使它能够完全局部性的实现多种重要操作。这些算法的效率仅仅依赖于局部的信息而不是整个电路的大小。这个结构的数据库能够被增量式的修改，所以能够在不影响任何其他部分设计的情况下修改设计中的一部分。角勾链对于密密麻麻的电路和疏松的电路一样有效。表III 总结了不同算法的复杂度。

这个机制的主要缺点是它需要接近三倍于简单机制的存储空间。幸运的是设计者大多数时候关注的都是布局中的一小部分，而角勾链仅仅用到局部信息，所以它在要求分页的环境中表现良好。