基于 OA 数据环境的集成电路布线形体处理算法及其应用

（申请清华大学工程硕士专业学位论文）

培 养 单 位：　计算机科学与技术系

工 程 领 域：　计算机技术

申　请　人：　汤启明

指 导 教 师：　喻文健　　副教授

二〇一二年五月

基于OA数据环境的集成电路布线形体处理算法及其应用　汤启明

The Algorithms and Applications of the Routing Geometry Processing in Integrated Circuits Based on OA database

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the professional degree of

Master of Engineering

by

Tang Qiming

(Computer Technology Engineering)

Thesis Supervisor Associate Professor Yu Wenjian

May, 2012

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：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

随着半导体工艺技术的进步，集成电路芯片特征尺寸不断减小，器件延迟已经不再占有主要地位，互连线上的寄生效应将日益成为制约电路性能的关键因素。并且随着芯片工作频率的不断提高，互连线寄生效应会造成芯片性能的下降，主要表现在噪声串扰、能量损失、信号延迟和波形扭曲等方面。对互连线寄生参数进行准确而快速的提取，直接关于芯片的设计质量。

在数字与模拟混合信号集成电路的设计中，差分信号以其较强的抗干扰能力得到了越来越广泛的应用，差分布线对被越来越多地应用在电路中关键信号的处理中。若差分对的走线不对称，有可能会产生比较严重的电磁干扰问题，以至于干扰到电路的正常工作。这要求在布线的过程中保持差分对走线的对称性，所以在布线阶段完成后，有必要对差分线网的布线结果进行检查。

在对以上两类问题的处理过程中，几何形体的处理与匹配是很重要的部分，直接关系到计算结果的准确性和效率。在有些情况下，对于一个大规模的版图，可能只需要对其中的少量线网进行计算，但一般的方法都是一次性处理全部版图数据，造成大量不必要的计算，导致时间上的浪费。

本文所提出的方法基于 “按需计算”的思想，通过处理目标导体及其周围相邻的少量导体就能得到正确结果，避免了对不必要布线形体的处理，对于“计算少量线网”的应用需求，在时间上大大缩短。本文的方法能够较好地处理“寄生电容参数提取”与“差分线网匹配检测”两类问题中所涉及到的布线形体处理，同时对于一次性计算版图全部线网的需求，也能够很好地适应。

关键词：电子设计自动化；寄生参数提取；差分对布线；布线形体处理

Abstract

With the advances in semiconductor process technology, the IC chip feature sizes decrease, the device delay is no longer occupies the main position, the interconnect line parasitics will increasingly become a key factor in restricting the performance of the circuit. With the continuous improvement of the chip operating frequency, interconnect parasitics will cause a decline in chip performance, mainly in the crosstalk noise, energy loss, signal delay and waveform distortions. Accurate and rapid extraction of interconnect parasitic parameters directly on the chip design quality.

In VLSI design, the differential signal its strong anti-jamming capability has been more widely used, the differential structure design has been increasingly applied in the signal processing circuit. If the differential pair alignment asymmetry may produce more serious EMI problems that interfere with normal circuit operation. This requires to maintain the symmetry of the differential pair traces in the wiring process after the completion of the wiring stage, it is necessary to check the differential line network cabling.

Geometry processing and matching is very important in the processing of the above two problems, in part, directly related to the accuracy and efficiency of the calculation results. In some cases, for a large-scale territory may only need a small amount of wire mesh which is calculated, but the general methods are one-time deal with all the layout data, resulting in a large number of unnecessary calculations, resulting in a waste of time.

In this paper, a kind of "on-demand computing" approach can only deal with the target conductor and a limited number of adjacent conductors to avoid processing unnecessary wiring body for the calculation of a small amount of wire mesh "application requirements, time significantly shortened. In this paper the method can handle "parasitic capacitance parameter extraction" and "differential line network to match the detected two problems related to wiring body to deal with the territory for a one-time calculation of net demand, and can well to adapt.

Keywords: EDA; parasitic parameter extraction; differential pair routing; geometry processing in layout routing

目 录

第 1 章 引言

1.1 集成电路技术的发展

集成电路(Integrated Circuit, IC)的出现是世界电子技术的一场革命，对人类社会的发展产生了深远的影响，对人类文明进步起到了不可估量的作用。自上世纪 60 年代被发明以来，集成电路技术一直遵循着摩尔定律高速而持续地进步，到现在早已经发展到超大规模和甚大规模的阶段。集成电路技术的进步为现代电子计算机技术发展的提供了坚实可靠的物质基础。

尽管已经逐步逼近特征尺寸的物理极限，但集成电路技术前进的脚步并没有停止。未来 10 年内，集成电路的特征尺寸仍将以每三年缩小 30%的速度继续发展，同时将伴随着晶体管密度以及时钟频率的快速进步。目前特征尺寸已经迈过了 45 纳米的门坎，向其物理极限继续前进，届时芯片集成度将达到10^8以至10^9的级别，在可以预见的未来，芯片复杂度仍将快速增加，对半导体制造工艺和芯片设计技术提出了越来越大的挑战。

电子设计自动化(Electronic Design Automation, EDA)技术的出现，正是为了应对芯片设计领域内的挑战，它能够有效地缩短芯片设计周期、降低设计成本并提高芯片的质量。EDA 技术广泛应用在机械、电子、化工、通信、航空、医学、军事等各个领域，可以说有集成电路设计制造的地方，都有 EDA 技术的身影。EDA 技术的发展对于集成电路技术进步有着极大的促进作用。

数字集成电路的设计流程包括前端设计和后端设计两部分。在前端设计中，通常使用 verilog/VHDL 等语言来进行行为级的描述，然后进行电路仿真和逻辑综合等。后端设计的任务就是按照前端的设计结果来生成真正的原理图和版图。EDA 软件的功能主要是进行按给定的工艺进行后端设计，包括布图规划、布局、布线及物理综合等几个重要步骤和阶段，以及版图寄生参数提取和静态时序分析等工具模块。通常将这一套工具整合在一起形成一个集成开发环境。目前世界领先的 EDA 工具开发企业有 Synopsys、Cadence、Mentor 等。

在接下来的两节，我们对集成电路中互连寄生参数提取与差分对布线分别进行简单介绍。它们是需要进行布线形体处理的两种应用情景，本文提出的算法也将针对它们进行应用。

1.2 互连寄生参数提取

随着半导体工艺技术的进步，集成电路芯片特征尺寸不断减小，器件延迟已经不再占有主要地位，而互连线延迟正在成为电路性能的重要参考标准。在目前的工艺下，器件延迟已经远小于互连线延迟，而且未来这种差距将不断拉大。并且随着芯片工作频率的不断提高，互连线寄生效应会造成芯片性能的下降，主要表现在噪声串扰、能量损失、信号延迟和波形扭曲等方面。当前工业界所采用的低介电常数介质工艺能够减小互连导体的电阻和电容，在一定程度上降低互连延迟的效应。但这并不能改变互连线延迟占主导地位的情况，只能在一定程度上来缓解。互连线上的寄生效应将日益成为制约电路性能的关键因素。所以对互连线寄生参数进行准确而快速的提取，是提高集成电路芯片设计质量的重要保证。

寄生参数的提取主要就是通过电磁场计算，来得到电阻、电容、电感等用来描述互连线电磁特性的电学参数[14]，它是进行后续静态时序分析的重要基础。随着电子工业的不断进步，寄生参数提取领域不断得到丰富，除了集成电路版图互连线，模拟/数字混合电路衬底耦合、器件封装中的管脚渗漏、集成微机电系统中的传感器、激励器分析[17]以及印制电路板(Printed Circuit Board, PCB)连线耦合等，都已成为寄生参数提取领域内的研究内容[37]。

由于本文的工作主要与寄生电容的提取有关，所以此处只介绍寄生电容的相关提取方法。

按照求解维度来划分，寄生电容提取技术可分为 1 维、2 维、2.5 维（或称准 3 维）以及 3 维几大类，目前广泛使用的主要是后三种技术。

2 维电容提取技术主要通过解析或数值方法，对 3 维结构的一个 2 维截面进行计算。虽然 2 维结构模型能够考虑到所有的情况，但它不能准确描述 3 维结构的互连线环境，应用范围不如 2.5 维（准 3 维）及 3 维模型广泛。

2.5 维提取技术[14]的主要思想是利用 2 维方法来解决 3 维的问题，这部分内容将在第 2 章介绍。

3 维提取技术被研究地最为广泛，它利用数值方法来求解静电场方程，以求解 3 维互连线结构的电容，这种方法的求解精度很高，但计算量非常大。3 维电容提取大体可分为两种[14]，即解析模型法和数值模拟法。解析模型法的优点是计算速度快，但精度较低，难以处理复杂的三维结构。数值模拟主要求解麦克斯韦方程组，化为微分方程组后有很多求解方法，有限差分法[28]，有限元方法[26]，边界元方法[16][38]。其中边界元方法以更少的变量和更高的准确度得到广泛应用[13][27][32][42]。

全芯片寄生电容的提取过程一般分为三个步骤：

第一步，工艺预处理，即对互连线结构进行建模。按照给定的工艺横截面的描述信息来建立数以万计的测试模型，再利用 2 维或 3 维场求解器对这些测试模型进行计算[13]，并将所得到的结果建立模式库或拟合出电容解析公式。

第二步是对互连线几何参数进行提取，这是预处理过程中不可或缺的一部分，同时它的计算量也是非常大的。若一个几何模式需要 10 个参数进行描述，且每一个参数有 6 个采样点，那么最后生成的模式库就存在 6^{10} 个结构。但如果用更少的参数来描述几何模式，那么最后的计算精度将难以保证。

第三步根据步骤二得到的几何参数以及步骤一得到的模式库来计算寄生电容。用提取到的几何参数去匹配模式库中的某个模式，但可能会有找不到匹配模式的情况。这时可以利用启发式方法，利用已有的模式及几何参数来近似得到电容值。

1.3 混合信号电路差分对布线

在模拟/数字混合信号电路的设计中，差分信号以其较强的抗干扰能力得到了越来越广泛的应用[1]，差分结构设计被越来越多地应用在电路中关键信号的处理中。通俗来讲，差分信号，就是用极性相反的两个信号来传输数据，依靠两信号的电位差来进行逻辑状态的判断。一般要求在布线的过程中应保持这一对传输信号的导线线宽和间距不变。以这种方式进行布线的一对线网走线，称为差分对走线。按差分对进行布线，可以使信号间的噪声干扰达到最低[2]。如果布线密度比较大，差分对之间会产生比较紧密的耦合，布线越靠近，差分信号之间的耦合也越强，相应地更不容易受到噪声干扰[3]。

若差分对的走线不对称，有可能会产生比较严重的电磁干扰(Electro-Magnetic Interference, EMI)问题[4]，以至于干扰到电路的正常工作。而且差分对线网中通孔的相对位置，也是影响信号稳定性的因素之一[5]。所以就要求在布线的过程中，应使差分对的走线尽可能的对称，并且通孔的位置也应该能够很好地保持走线的对称性。所以在布线阶段完成后，有必要对差分线网的布线结果进行检查，主要包括两个方面，一是计算差分对走线平行部分占总长度的比值，二是统计满足对称要求的通孔数量。

1.4 论文工作及章节安排

在互连寄生参数提取中，几何形体的处理与匹配是一个很重要的部分，关系到能否准确快速地计算版图互连线的寄生电容。在有些情况下，对于一个大规模

的版图，可能只需要提取其中某几个线网的寄生参数，所以直观上看只需要处理这几个线网的几何形体以及其邻近的一些互连线即可。文献[11]提出了一种针对布线导体的扫描线方法，它的特点是一次性处理版图上的全部导体。对于提取整个版图全部线网寄生参数的需求，这样做是可以的，但对于前面提到的“只提取个别线网”的应用情况，会造成大量不必要的计算，导致时间上的浪费。

对于差分对线网的布线匹配检测，也存在和寄生参数提取中形体处理相类似的问题。由于差分对线网都是成对出现，所以检测时一般只需要考虑这对线网所包含的布线形体。那么显然也需要在匹配检测过程中避免处理不必要的互连线。在此过程中所涉及到的形体处理与寄生参数提取存在较大的相似性，本文分析并提炼了这种相似性，将二者合并处理。

本文以“按需计算”的方式来处理“提取个别线网寄生参数”的情况，只处理与目标导体邻近的有限数量的导体，避免了对无关布线形体的处理，节省了时间。在此过程中，保存对相关导体的计算结果，不同线网间几乎没有需要重复处理的形体，所以本文的方法对“提取全部线网”的需求也能很好地适应。

余下章节内容安排如下：

第 2 章首先介绍了 OpenAccess(OA)数据平台的相关内容，然后介绍 2.5 维电容提取模型、寄生电容网络的建立和差分对线网的匹配规则，以及三者所涉及到的形体处理的需求。

第 3 章详细介绍本文所提出的“按需计算”方法，主要包括数据结构的设计和相关算法实现。

第 4 章以第 3 章为基础，详细介绍以“按需计算”的方法解决 2.5 维寄生参数提取以及差分对线网匹配检测的问题。

第 5 章对文章做出总结，并对今后的改进提出展望。

第 2 章 OA 数据环境与两种集成电路布线后处理

本章第 1 节对 OpenAccess（简称 OA）做出概述，并对其中涉及布线形体表示的内容进行了简单的介绍。接下来介绍两种集成电路布线后的形体处理，分别涉及版图寄生电容提取，和混合信号电路差分对布线检测。其中对版图寄生电容提取的介绍将分“2.5 维电容模型”和“电容寄生网络”两个部分，分别安排在第 2 和第 3 节。第 4 节将介绍混合信号电路差分对布线检测的相关内容。

2.1 OA数据环境简介

在 EDA 领域，信息的处理有着数据量大、数据类型繁多的特点，很多 EDA 开发厂商都有自己的数据平台。如 Synopsys 的 Milkyway、Mentor Graphics 的 Falcon 等等。不同的 EDA 软件所采用的数据格式有很大差别，很多数据格式之间互不兼容。不同工具之间的数据差异使得许多设计资源无法共享，难以利用不同的工具进行优势互补，给集成电路的设计、验证和制造带来了很大阻碍。

OpenAccess[9]的出现让业界看到了解决众多数据格式互不兼容问题的希望。OpenAccess 数据平台是一款专门面向 EDA 设计的开源软件类库，以 C++作为开发语言。OA 以统一的形式进行数据描述和存储，并且对于运行在不同操作系统平台上的 EDA 工具，OA 也以统一的数据格式保证彼此的兼容。

OA 不仅提供了统一的数据格式，它还有着设计良好的面向对象体系结构和简明清晰的 API 接口，为 EDA 软件的开发带来了极大的便利。相对于同类型的其它产品，OA 能够以更少的内存代价来提供更优越的性能。越来越多的 EDA 软件产品采用 OA 作为基础数据平台，OA 正逐渐成为 EDA 领域的开发标准。本文中所涉及到的集成电路布线形体就是基于 OA 所提供的相关数据结构来表示。下面分别介绍 OA 的层次式“域”的划分，以及本文后续内容涉及到的相关 C++类。

2.1.1 OA设计层次的三个域

随着特征尺寸的减小，同样面积大小的芯片的复杂度越来越高，对芯片设计提出了挑战。OA 类库通过合理的抽象，将一个十分复杂的体系结构以分层次、多级化的形式来表示，简化了问题的描述。它能够改变一个层次的接口定义，确不影响其它层次的引用，这对于大型项目的开发有着非常重要的意义。

一个多级电路在 OA 的表示中对应着三个域：Module、Block 和 Occurrence.

Module 域主要对电路进行逻辑上的描述，以 Hierarchy 结构来表示电路设计。Module 域中的数据包括线网连通性信息以及扩展对象的连接信息，通常是硬件描述语言（Verilog 或 VHDL）综合的结果，它将整个芯片划分为若干个区块，不同的区块对应着不同层次的逻辑结构。

Block 域以 Hierarchy 结构表示芯片的物理实现。Block 域中的数据包括线网的连接以及各种物理形体的几何信息，如导线、通孔、线网终端以及寄生元素等等。Block 域中的层次划分比 Module 域要简单。如图 2-1 所示，线网 A 由 A1 和 A2 两个部分连接而成，在 Module 域中以一个逻辑线网的形式存在，而对应地在 Block 域中，线网 A 分为三个部分，分别是 A1，A2，和导线连接。

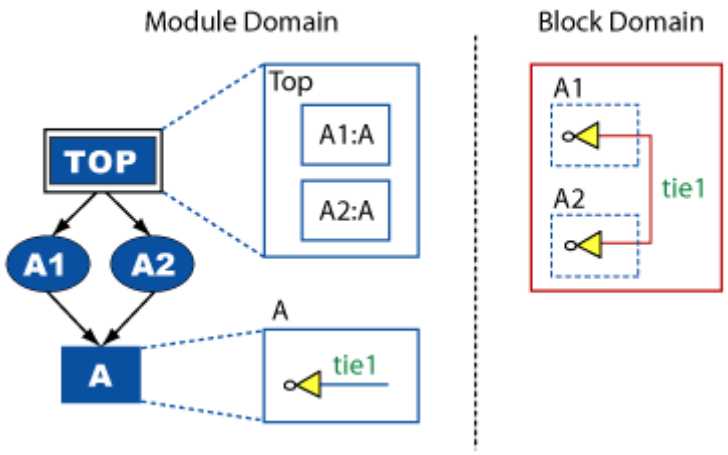

图 2-1　线网 A 分别在 Module 域和 Block 域中的表示

Occurrence 域可以看作是 Module 和 Block 两个域的联合体，它的结构形式是展开的，其中的多级边界保持了可连接性，以便支持对多级结构进行自底向上的遍历，并可以对每一个位于 Module 域和 Block 域中对应的对象进行访问。不同于前两个域，在 Occurrence 域中不能够改变器件连接关系，也就是说在这个域中不可以添加线网连接。

以上三个域，可以看作是同一个芯片按不同概念所进行的描述，并不是将一个芯片划分为三个部分，而是将这个芯片以三种不同的方式来解释，三个域对应的是同一个芯片。就如同一个人的三个不同身份。

2.1.2 本文所用相关C++类的介绍

OA 类库十分庞大，但是 API 文档就包含超过 600 多个类。本文的工作主要是对布线形体的处理，所以在这里只介绍与之有关的类及相关功能。

oaDesign，代表一个芯片设计，包含该芯片全部的设计数据。以下简称 design。

通常一个芯片设计数据对应一个 design 对象，并且一个 design 可包含对其它 design 的引用。

oaBlock，表示版图的整个布线区域（以下简称 block），是一个 design 中全部 Block 域对象的顶层结构。从 block 对象可获取版图区域外边框等几何信息。一个 design 内只含一个 block，并且在 Block 域中，对其它 design 的引用也是以 block 的形式出现，当前 block 与被引用的 block 坐标系是不一样的，可以通过一定的方法互相转化。调用 getNets()函数可获得包含版图全部线网的集合。

oaNet，表示版图中一个线网，其中包含导体、通孔以及线网终端。通过它可以获得线网的全部导体、通孔的引用信息，以及线网驱动端和接收端的使用情况。调用 getShapes()，可得到线网全部布线形体的集合。通常布线形体的类型为 oaShape，调用 oaShape 对象的 getType()函数可以判断该 shape 是否为一段导体。

oaPathSeg，表示一段导体，是 oaShape 的派生类，该段导体直观上可抽象为一条线段，也就是 oaPathSeg 所表示的这段导体是不改变走向的。从 oaPathSeg 对象可以获得导体中心线两端在版图中的坐标、导体的线宽等几何参数。图 2-2 是一段 oaPathSeg 所表示的导体，图中列出了导体中心线两端的坐标，以及导体的线宽。相应的，OA 还提供了 oaPath，以表示一段折线形式的导线。

oaVia，表示通孔，可以获得通孔所连接的上、下层号，通孔的外边框及其尺寸等几何信息，通过 getViaDef()成员函数，可以得到通孔的工艺信息，不同的通孔对应着不同的工艺信息，在一个芯片设计数据库中，同种通孔可能存在多个实体，但其对应的工艺信息只有一份。oaVia 的外边框通常是矩形。

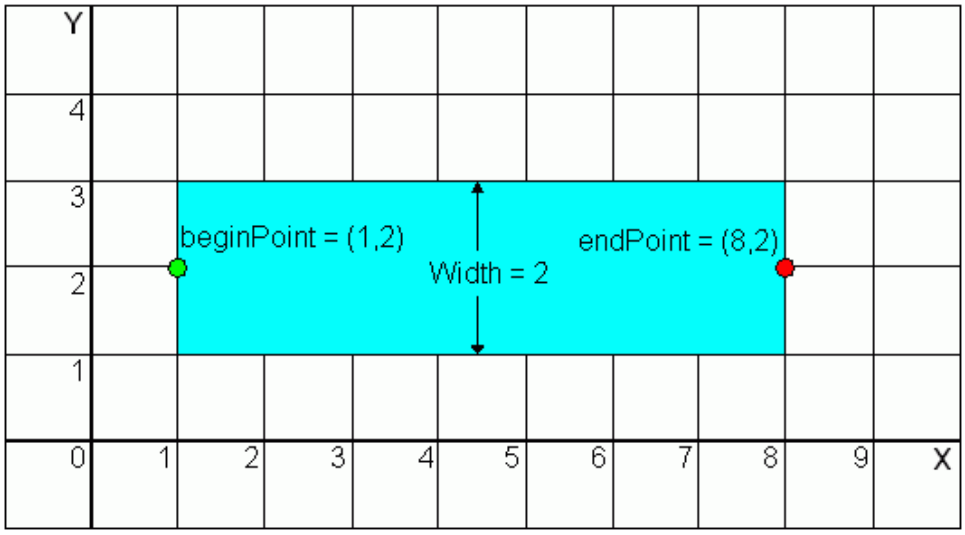

图 2-2　oaPathSeg 示意图

oaTerm 和 oaInstTerm，表示线网终端，可能是线网驱动端（driver），也可能

是接收端（receiver），前者所表示的终端属于当前 block 内线网的终端，后者表示当前 block 内所引用的其它 block 内含的终端，被引用的 block 内的终端在当前 block 中映射为 oaInstTerm 类型。从 oaTerm 不能够直接获取其边框信息，因为 oaTerm 内包含了若干称之为 fig 的组成部件，这些 fig 通常是矩形，一般是将所有 fig 的外矩形合并为一个能包含全部 fig 的大矩形，来代表 oaTerm 的外边框。可以用类似方法来获取 oaInstTerm 的外边框，但其中有一点不同。因为 oaInstTerm 所表示的终端实际上包含在其它 block 内，所以从 oaInstTerm 所读取的坐标信息是其所在 block 坐标系中的，那么就需要将其转换为当前 block 内的坐标，才能够得到正确的几何信息。

oaPoint，表示一个二维坐标系上的点，可获取其横、纵坐标。

oaBox，表示一个矩形，一个 oaBox 存储了矩形的几何信息，并提供一组对该矩形进行操作的函数，如交叉、合并、取中心点等等。oaPathSeg 以及 oaVia 都提供了直接获取表示其几何外边框 oaBox 对象的函数。

oaTech，包含版图的工艺参数，本文主要从中获取版图布线的层号信息。一种工艺只对应一个 oaTech 对象，可以通过调用 design 对象的 getTech()函数来得到。

2.2 二点五维寄生电容提取

2.2.1 二点五维电容提取模型

典型的版图电容提取流程以一定数量互异的简单导体环境作为基本模式，使用二维或三维场求解器计算出导体位于这些环境下的电容，并用其中的一种或几种组合来匹配实际版图中的导体环境，再用插值或拟合公式的方法来分段计算导体的单位长度电容值。

二点五维电容模型是一种导体环境模型，如图 2-3 所示。图中所示的三维区域是一个立方体环境，求解结构就是一个三维的模式。这个环境中所包含的参数有导体的长、宽、高，同

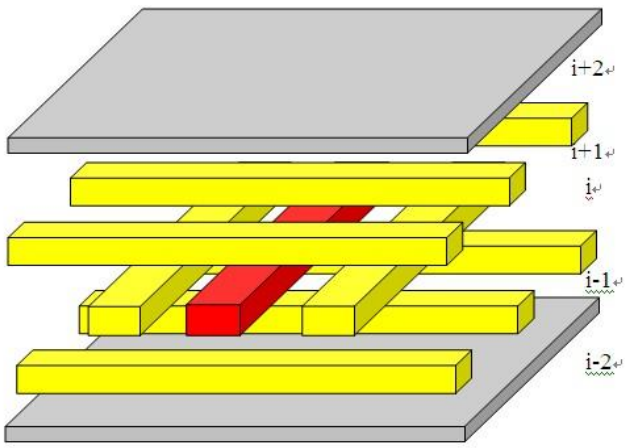

图 2-3 二点五维电容模型示意图

层平行导体的间距，邻层间的垂直距离，导体层的厚度，以及各层的布线面积比（布线面积比的概念将在后面介绍）。

图 2-4 给出了二点五维电容模型示意图，这是一个简化的二点五维电容模型。在这个模型中，可以把电容根据目标导线层和其他层之间的关系也分成三种电容，即同层之间的侧向耦合电容，邻层之间的交叉电容和隔层之间的对地板电容。其中邻层之间的交叉电容，为目标导线与邻层金属线的重叠电容，目标导线侧面和邻层金属线上底面的边缘电容以及邻层金属线侧面和目标导线下底面的边缘电容三部分之和。同层侧向耦合电容，是目标导线与同层邻线的耦合电容。隔层的对地板电容，是目标导线与隔层地平板之间的重叠电容和目标导线侧面与隔层地平板间的边缘电容之和。

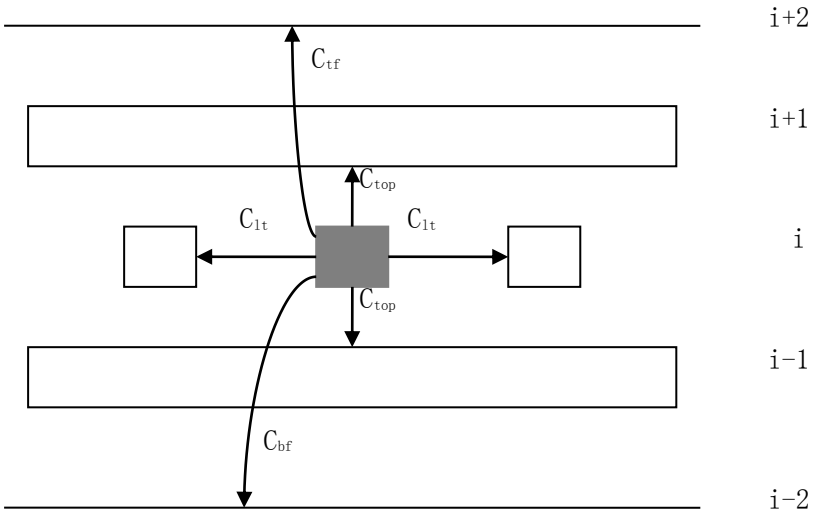

图 2-4 简化的 2.5 维电容模型截面图

2.5 维建模可以遵循五个准则。假设目标导体所在层号为 i，

1)地平板和目标导体的同层相邻导体，在建模中必须要考虑；

2)当第 i 层布线超过一定密度时，i+1 层与 i-1 层之间的耦合效应可以忽略；

3)i±2 层可以当成地平板，且 i±层以外的金属层可以不用考虑；

4)在同一层中，只需考虑距目标导体最近的相邻导体，且左右两侧的相邻导体之间是相互独立的，相邻导体的宽度所造成的影响可以忽略；

5)i±1 层金属线对第 i 层金属线影响的相互关系是可以被忽略的，也就是说第 i-1 层和 i+1 层对其的耦合作用是可以分别单独考虑的。

2.2.2 用于建立电容数据库的拟合公式

拟合公式有三个变量：主导体线宽、同层左右相邻导线间距的平均值、相邻层的布线面积比。假设当前求解导体位于 i 层，那么相邻层导体的布线面积占整个

版图区域面积的比值，就是相邻层的布线面积比。在实际计算中，往往选取一定大小且覆盖被求解导体的矩形窗口，分别投影到邻层，此时用包含在该投影区域内的导体布线面积占该矩形窗口面积的比值做为计算时所需要的布线面积比。

计算导体结点电容所采用的拟合公式如下：

$r=0$ 时

$$\begin{cases} C_{it}=a_0+a_1\dfrac{1}{s}+a_2\left(\dfrac{1}{s}\right)^2+\ldots+a_k\left(\dfrac{1}{s}\right)^k \\ C_{tf}=C_{bf}=a_0+bw+a_1s+a_2s^2+\ldots+a_ks^k \end{cases}$$

$r\neq 0$ 时

$$\begin{cases} C_{it}=a_0+a_1\dfrac{1}{s}+a_2\left(\dfrac{1}{s}\right)^2+\ldots+a_k\left(\dfrac{1}{s}\right)^k \\ C_{tp}=a+bw+cr+dwr \\ C_{tf}=C_{bf}=a_0+a_1r+a_2r^2+\ldots+a_kr^k+\left(b_0+b_1r+b_2r^2+\ldots+b_kr^k\right)w \end{cases}$$

其中 s 代表同层导体的平均线间距，w 代表导体线宽，r 是邻层布线面积比。此公式为两种情况，在 r 为 0 或非 0 的情况，对应着不同的计算公式。

如图 2-2 所示，位于 i 层中间的主导体共有 6 部分电容：在相同层与左右相邻导线间的两个耦合电容 Clt，与第 i+1 层和第 i-1 层导体之间两个耦合电容 Ctop，与第 i+2 层和第 i-2 层金属板之间两个耦合电容 Ctf 和 Cbf。由于求解区域具有对称性，所以只要拟合对称的两个耦合电容之中的一个就可以了。（实际处理的时候是左右的电容值相加除以 2，当做 Clt，上下电容值相加除以 2，当做 Ctop）。

因为拟合的时候是按照上下左右完全对称来处理的，所以 r 代表的是 i±1 的布线面积比，实际上，如果遇到当前层的上面一层不存在，而下面一层存在，或者当前层的下面一层不存在，而上面一层存在，此时，可以分别按照两种布线面积比计算电容值后取平均值。对左右线间距的处理类似。

2.2.3 二点五维电容提取中的形体处理

根据上一小节所介绍的拟合公式，在提取版图电容的过程中，需要获得导体的三个参数：线宽、左右间距、邻层布线面积比。线宽很容易获得，可直接从 oaPathSeg 对象中读取。但另外两个参数的获取并不容易，需要对版图导体的几何形体进行一定处理和计算才能得到。

首先要明确“相邻导体”的概念。如图 2-5 所示，三根垂直方向的平行导体 A、B、C，其中 A 被划分为 3 段，第一段与 B 的间距为 S1，第二段与 C 的间距为 S2，很明显，S1 不等于 S2。应该注意，在第 1 段所在区域中，只有 A 和 B 是相邻的，那么在这个区域里 A 和 B 是相邻导体，但 A 和 C 不是相邻导体；同样，在导体 A 第 2 段所在区域中，A 与 C 是相邻导体。在本文中，判断两导体是否相邻，要根据实际的导体环境分段来考虑，所以对相邻导体的查找也要分段来进行。

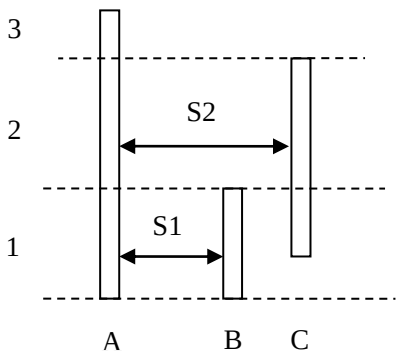

图 2-5 相邻导体示意图

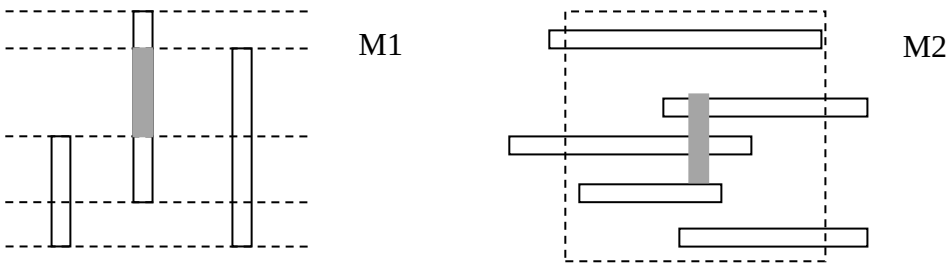

图 2-6　布线面积比示意图

图 2-6 是布线面积比的示意图。左侧为 M1 层布线区域，图中所要示意的就是中间导体灰色的一段的上层布线面积比。图 2-6 右侧是 M2 层布线区域，矩形虚线框是布线面积比的求解区域，图中标示了 M1 层呈灰色的那段导体的位置，可看到其位于矩形区域的中心。那么这里的布线面积比就是虚线围成的矩形区域内 M2 层导体所占面积，除以该矩形区域面积。

2. 3 寄生电容网络

2.3.1 OA的寄生效应模型

在 2.5 维电容提取模型中，线网被划分为若干个结点，每个结点单独计算出一个电容值。线网所有结点电容计算完成后，将这些结点按照它们在线网中的相对位置连接起来，便形成了一个寄生电容网络。

OA 中的寄生效应模型有两类，分别是简化模型和详细模型。本文的工作涉及

后者，所以在这里只对后者进行介绍。

OA 中的详细寄生效应模型主要由 oaParasiticNetwork 及其相关派生类来描述。对于给定的线网，存在一个 oaParasiticNetwork 类型的对象作为该线网的寄生效应网络，这个网络可以看作一个图，其中包含若干节点和设备，分别用以表示寄生效应中的各个元素。通过对线网中所包含的导体和通孔等元素的几何形体进行分析，判断出它们彼此的连接关系，就可以依照这个框架将线网的各个电容结点连接起来，形成一个寄生电容网络。

在 OA 数据库中，可以为一个线网创建相关联的寄生网络，即 oaParasiticNetwork 对象，这个线网位于 Block 域或 Occurrence 域。通过对版图中所有线网进行遍历，我们可以创建与这些线网相关联的寄生网络。线网连接关系的改变，如线网的创建、销毁以及线网终端位置的变化，将会引起相关联寄生网络的卸载。下面将介绍与寄生网络有关的类。

oaParasiticNetwork，它将寄生参数表示成网络上的节点与设备，所形成的寄生网络可分别与 Block 域和 Occurrence 域中的线网进行关联。对于 P/G 线网，由于其规模很大，相应的寄生网络会包含非常多的节点与设备，当规模超过一定限度时，这些节点与设备将不能全部放置于内存中。为此，OA 的处理方法是将一个大规模的寄生网络划分为多个小规模网络，每个小网络也是 oaParasiticNetwork 类型的对象，并有自己的名字和几何边框，以描述顶层寄生网络如何划分。每个小网络可继续进行划分，那么整个寄生网络就形成了一个分层次的树状结构。下层网络可访问上层网络中的节点和设备，而上层网络被卸载时，下层网络也将被卸载。

oaNode/oaGroundedNode，表示寄生网络中的节点，在布线拓扑结构中表示一个点的位置，在 oaParasiticNetwork 中用以对设备和终端进行连接。2.2 节中提到对导体进行划分，即将一个较长导体划分为若干较短导体，那么这里每一段较短导体就可以表示为一个节点，节点的位置对应于分段导体的位置。在实际建立寄生电容网络时，一般采用 oaNode 的派生类 oaGroundedNode，其存储的电容值就是该节点与隐含接地板间的电容。

oaDevice，是表示寄生网络设备的抽象基类，每一类设备对应 oaDevice 的一个派生类，以表示指定类型的寄生元素，如电阻器、电容器和电感器。寄生网络中每一个设备有两个端点（oaNode 类型），这两个端点可认为是对称的。在本文所应用的寄生网络中，连接两个节点的设备一般是 oaResistor 类型，表示一个电阻器。由于这里不需要考虑电感效应，为简化模型，在寄生网络中只放置了电阻器。

oaSubNetwork，用以表示寄生网络中全部设备和节点的一个子集，一般用来描述设备/节点和对应布线形体之间的关系，如一条导体连接到一个通孔，或几条

线连接至一个点，等等。

寄生参数相关数据量通常很大，为了优化内存占用量，OA 采取了分页策略，允许在某一时刻只将网络中一部分数据放于内存。当应用程序需要某一部分数据，而该数据不在内存中，OA 将自动从外存将这部分数据加载进来，并将暂时不需要的数据保存到外存中去，以减小内存占用，类似于操作系统的内存分页策略。若一个已经存在相关联的寄生网络，则可调用 oaParasiticNetwork::load()函数进行加载，同时 OA 针对该寄生网络维护一个计数器，用以记录程序中有多少个对该寄生网络的引用。调用 oaParasiticNetwork::unload()可以将寄生网络卸载，同时计数器减一，若减一后计数器并不为 0，则该寄生网络仍被放于内存中，直到计数器为 0，才将之从内存中清除。当寄生网络被卸载，其中的设备与节点将不再可用，若此时应用程序引用这些节点和设备，将导致不可预测的后果。

2.3.2 与寄生网络有关的形体处理

版图中的线网按照物理结构大至可分为三个组成部分：导线、通孔和终端（包括 driver 和 receiver）。这三个部分都具有一定的几何形体，存在着一定的重叠连接，以构成一个连通的线网。结点电容寄生网络就构建在这个基础之上。如果把线网比作树木，那么寄生网络就是附着在树木上的藤条，若要正确建立寄生网络，必须先确定线网各部分的几何连接信息。

如图 2-7 所示的线网连接，导体与通孔存在一定的几何重叠。简单的说，与寄生网络有关的形体处理就是判断若干几何图形的连接关系。

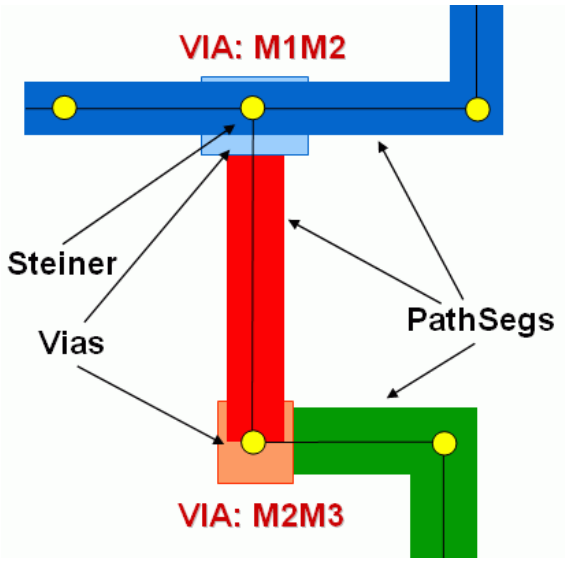

图 2-7　线网连接示意图

OA 中的 oaRoute 类可以表示一组布线元素的连接关系，可以将线网分成若干没有分支的部分，那么每个部分就可以用 oaRoute 来表示。但分支之间的连接关系仍然不能确定，无法将各分支组成一个完整的连接线网。而且在有些数据库中，并没有建立 oaRoute，这时连线网分支内部的连接关系都判断不了。所以有必要采取一种统一的方式来判断线网各布线形体之间的连接关系。

2.4 差分网线的匹配规则

导线的“匹配长度”定义如图 2-8 所示，其中线网 A 和线网 B 构成一个差分对。在情况(a)中，有两条位于 M1 层的平行导线，分别属于线网 A 和线网 B，那么在这一区域，线网 A 相对于线网 B 的匹配长度为 S，也就是两导线在此区域内重叠

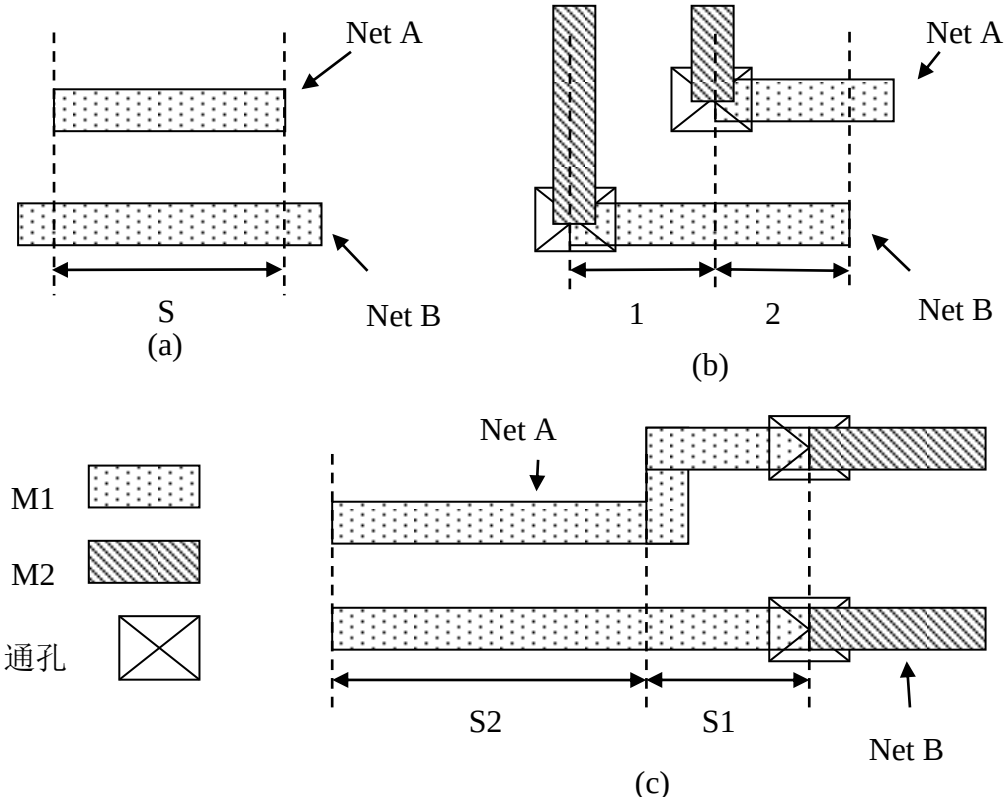

图 2-8　导线匹配长度示意图

部分的长度；若无特殊情况，线网 B 在这一区域相对于线网 A 的匹配长度也为 S.

然而并不是所有的匹配长度都是两平行导线间的重叠长度，在(b)情况中，可以看到两条 M1 层的水平导线，分别经过通孔与 M2 层导线相连。这里我们考虑 M1 层的情况，此区域中线网 A 相对于线网 B 的匹配长度，即为两导线间的重叠长度 S2. 而线网 B 相对于线网 A 的匹配长度，则没有这么简单。可以看到两层导线的走向在通孔处发生了改变，而为了让 M2 层的导线之间也保持一定的间距，M1 层位于下方的水平导线必须要比上方导线多出一段长度，也就是图中长度为 S1 的那部分。按照一般的布线工艺，相邻层导线走向是垂直正交的，所以(b)所示的情况是很常见的。那么 S1 所表示的部分，可以视为对导线走向发生改变所做出的一种适应性调整，所以也被计入匹配长度的计算，也就是说线网 B 相对于线网 A 的匹配长度为 S1+S2. 但此处有一个前提，M2 层导线间距必须满足匹配要求，

否则 S1 所示的适应性调整将被视为无效，不计入匹配长度中。

(c)所示情况比较少见，可以看到，M2 层导线的走向与 M1 相同，且间距大于 M1 层。M1 层两段水平方向的导线经过通孔连接到 M2 层。注意在 S1 区域，M1 层导线间距发生改变，这一段可以看作一种适应性的调整，以正确连接到 M2 层导线，那么它可以记入到匹配长度中。也就是说，对于(c)所示情况，在 M1 层中，线网 A 和 B 相对于彼此的匹配长度都是 S1+S2.

综合以上三种情况，每段导线的匹配长度分为两个部分，可以用公式(2-1)来表示：

$$L = l_{overlap} + l_{offset} \tag{2-1}$$

其中$l_{overlap}$为两段平行导线重叠部分的长度，l_{offset}表示导线为了改变走向或改变间距所作出的适应性调整。显然，对于中(a)的情况，$l_{overlap} = S$，$l_{offset} = 0$，而对于(b)和(c)，$l_{overlap} = S2$ ，$l_{offset} = S1$.

不是所有情况的l_{offset}都可以计入匹配长度中。考虑图 2-8 的情况(b)，若 M2 层的两条导线经通孔出发去往相反的两个方向，则 S1 所示区域不再被看作适应性调整，那么在 M1 层，线网 A 和 B 相对于彼此的匹配长度都为 S2.

通孔位置的匹配是在导线匹配的基础之上进行判断的。一个通孔要连接上下两层导线，如果有这样一对通孔，它们分属于一个差分对中两个线网，如果这对通孔所连接的两对导线是分别匹配的，并且每段导线与通孔连接的末端可全部记入匹配长度中，那么这对通孔的位置就是匹配的。

假设图 2-9 中线网 A 和线网 B 构成差分对，且 M1 和 M2 层布线间距满足匹

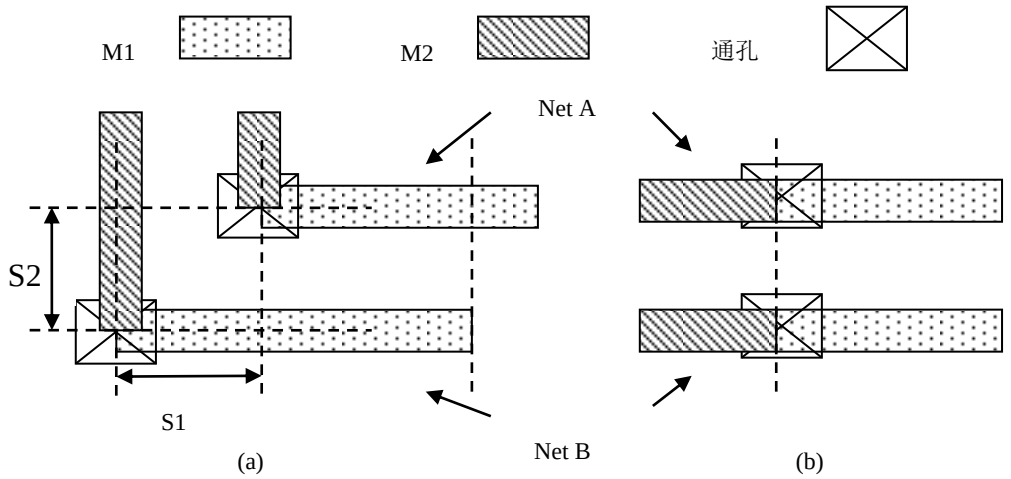

图 2-9　通孔位置匹配示意图

配的要求。在大多数情况下，相邻两层布线是垂直正交的，在情况(a)中，S1 和 S2 所示区域分别是 M1 层和 M2 层对导线走向改变所做出的适应性调整，四段导线与

通孔连接的末端可全部记入匹配长度中，根据前面的规则，可以认为这一对通孔的位置是匹配的。对于(b)的情况，导线经通孔换层后，走向没有发生改变，但可以发现无论 M1 还是 M2 层的成对导线，它们与通孔连接的末端连接与其走向垂直，也就是末端可全部记入匹配长度中，所以这一对通孔的位置是匹配的。若(b)中 M1 层这对导线与通孔连接的末端连线不与其走向垂直，那么这个末端部分将不能记入匹配长度中，因为它不在两导线的重叠区域，所以这时两个通孔的位置就不是匹配的。

2.5 本章小节

本章首先对 OpenAccess 进行了简要介绍，并列举了本文涉及的一组类 C++类。接下来介绍了“布线后寄生电容提取”和“差分线网布匹配检测”两种典型集成电路布线形体处理的应用。本文余下的工作将围绕这两种应用中所涉及到的布线形体处理来展开。

第 3 章 基于 OA 数据环境的改进形体处理算法

第 2 章介绍了“布线后寄生电容提取”和“差分线网匹配检测”两种应用中所涉及到的布线形体处理。本章接下来将分析这两种布线形体处理之间的共同之处，并基于这些共同之处设计一种数据结构来处理这两类问题，同时给出该数据结构的建立算法及相关操作。

3.1 对形体处理的分析

第 2 章讲述了集成电路布线形体处理的几种应用，这几种应用之间存在一些共同点，下面将具体分析并提炼这些共同之处，皆以垂直走向的导体为例。

A. 根据第 2 章的分析得知，导体的左右间距是分段描述的。要获得这些间距信息，需要导体两侧一定距离内的若干导体参加。这实际上相当于在若干平行导体中查找位于给定水平区间内的导体。

B. 计算邻层布线面积比，需要得到给定大小的矩形窗口中导体的总面积。这可以分两步来进行，首先，确定哪些导体位于该矩形窗口所划定的水平范围内，这相当于 A 中所归纳的问题；然后，再在这些导体中，确定哪些导体位于该矩形窗口所划定的垂直范围内。经这两步所选出的导体就是邻层中位于该矩形区域内的导体。

C. 在创建寄生网络的过程中，需要判断线网几何形体的连接关系。其中很大一部分工作，是判断通孔与导体的连接关系，也就是判断哪些导体与通孔的几何边框有重叠。而通孔的几何外形成矩形，所以此处的问题与 B 所归纳的类似，只是这里的“矩形区域”要比 B 中小的多。

D. 检测差分线网的匹配，很大一部分工作就是检查两个差分对线网之间布线的平行长度，那么这样就与 A 的问题相类似了。

综上所述，这 4 个问题可归结出一个共同点，即“查找给定一维区间内存在的平行导体”，并且这个一维区间在平面上的方向，可能与导体平行，也可能垂直。

以上 4 个问题可归结为两种操作，一个是在若干平行导体中查找位于给定的与导体走向垂直的区间内的导体；另一个是在若干平行导体中查找位于给定的与导体走向平行的区间内的导体。

第 1 章提到过，本文的工作主要针对快速处理大规模版图中的少部分线网的

需要，同时又能适应处理全部线网的需要。本章所介绍的数据结构，就是针对这两种需要而设计的。

3.2 数据结构的设计

首先，各平行导体应该是有序排列的。以垂直走向的导体为例，排序的依据是导体两端点的横坐标，也就是将若干垂直走向导体按照它们在水平方向上的相对位置进行排序。这样，导体间的左右相邻关系就确定了，可以很方便地获知相邻导体的数据。

直观的方法，是将每一层导线按照相对位置进行排序，然后用二分法进行查找操作。如果需要一次性提取版图上全部线网数据，那么这种方法实现起来是简便易行的。但很多时候，我们的目的是只提取版图中某几个线网的参数，那么对于大规模的版图来说，把每一层导线全部进行排序，会涉及到很多不需要考虑的导体，这就有些浪费计算时间了。

显然，对于一根待处理的导体，我们需要获得的只是它左右相邻的有限的几根导体，并不需要知道版图上其它导体的情况。当然，如果需要一次性处理版图上的全部线网，那么将每一层上所有平行导体进行一次性排序是合理的。但有时只需要提取个别线网的数据，那么对于大规模的版图而言，提取线网数据前要处理全部导体会浪费很多时间。

经过分析可以发现，在处理单个线网的导体时，只需要知道该导体邻近的几个平行导体的情况就可以了，也就是说只需要对该导体及其邻近几个导体进行排序即可。根据 OA 数据库的特点，导体之间是没有次序的，也就是说在编制程序时，通过调用 OA 提供的 API 所得到的导体是无序的，无法直接获得导体的相邻情况。

本文的解决办法是，将每一层的版图区域进行划分。假设当前层布线为垂直走向，那么可将该层划分为若干垂直区域，并使这些区域的水平宽度相等。该层所有导体分布在这些区域中，若布线密度比较均匀，那么各个区域内所含导体数量是比较接近的。区域宽度的选取要适中，若过小，则划分区域的数量会很大，相应的存储空间也会增大；若过大，则区域内包含的导体数量会很多，那么操作中涉及到的无用导体数量会相应增多。一般使每个区域内包含 10 根导体比较合适。如果布线密度不均匀，则会出现有的区域包含导体过少甚至没有的情况，将造成空间的浪费。

如图 3-1 所示，矩形代表版图区域，四条虚线将之划分为 5 个等宽区域。对于水平方向布线的层区域，划分方法类似。如果版图布线密度比较均匀，那么每个区域内所包含的导体数量也会是相近的。

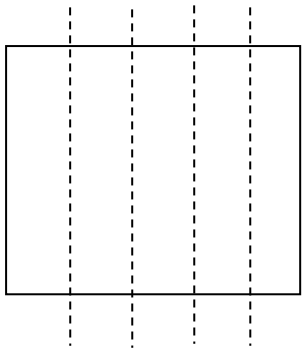

图 3-1 区域划分示意图

从 OA 数据库中可以获得版图的全部线网，但这些线网没有固定的顺序，并且从线网中获取的各个导体之间也没有固定顺序。我们需要知道每个区域包含了哪些导体，这就需要将所有导体遍历一次，根据导体的位置信息将之插入到对应区域的集合中去。

插入操作完成后，每个区域所对应集合内将包含若干导体，但集合内各导体之间是没有顺序的。每当需要查询某导体及其相邻导体的信息，先判断该导体位于哪一组内，若该组内导体没有排序，则先对其进行排序，再查询相邻导体信息；若已排序，则可直接获得。

这样，只对线网各导体以及它们的邻近导体进行操作即可，而不用对所有的平行导体进行处理。当版图所含线网数量很多，而只需要提取某几个线网数据时，这种分组划分的方法能够减少很多不必要的计算。

相关数据结构设计如下(C++代码描述)：

```
class Point {
    int p;
    oaPathSeg *parent;
};
```

类 Point 表示导体的端点。其中 p 表示端点在导体走向上的坐标，对于水平导体来说，p 是端点的 x 坐标，对于垂直导体则是端点的 y 坐标。parent 表示该端点所属的导体。

```
class Section {
    typedef    std::vector<oaPathSeg*>    seg_list_t;
    typedef    std::vector<Point>    point_list_t;
bool sorted;
    seg_list_t    seg_list;
    point_list_t    point_list;
public:
```

```
    void insert(const oaPathSeg *ps);
    void sort();
    std::vector<oaPathSeg*>    get_pathseg(int a, int b);
};
```

类 Section 表示将版图区域划分所得的“组”，其中 sorted 是该组已排序与否的标志，组内的导体存储在 seg_list 中，用标准库类型 std::vector 来实现。point_list 是组内各导体端点按坐标大小排序后的序列。成员函数 get_pathseg 接收两个参数，[a,b]作为一个区间，此函数用来实现查找位于该区间内的所有导体。查找算法请见表 3-3.

```
class Layer {
    typedef std::vector<Section> section_list_t;
int seg_count;
    int seg_width;
    section_list_t    section_list;
public:
    Section& get_section(int i);
    std::vector<oaPathSeg*>    get_pathseg(int a, int b);
};
```

类 Layer 表示一个布线层的组织结构，将该层所有划分后的“组”结合在一起。其中 seg_count 表示该层所包含“组”的个数，seg_width 表示组宽度，所有的“组”按照相对位置排列排列成一个数组，因为元素个数不确定，所以用 std::vector 来表示该数组。成员函数 get_pathseg 接收两个参数，[a,b]作为一个区间，此函数用来实现查找位于该区间内的所有导体。查找算法请见表 3-2.

```
class Layout {
    typedef    std::map<int,Layer>    layer_map_t;
    layer_map_t    layer_map;
public:
    void add_layer(int i);
    void remove_layer(int i);
    Layer& get_layer(int i);
};
```

类 Layout 表示版图，主要作用是将代表每一层的 Layout 对象组织在一起。由于版图各层序号可能不连续，所以这里将层号和 Layer 对象建立映射关系，用标准

库中 std::map 类型来实现。

3.3 数据结构的建立算法

版图每层被划分为若干等宽区域，将这些区域按顺序从 0 开始编号。以垂直方向区域为例，显然这些区域排列在水平方向上。将导体（显然也是垂直走向）的横坐标除以区域宽度，所得结果取整，便得到了该导体所属的区域。这里应该注意，版图的横坐标范围应该从 0 开始，否则要减去一定的偏移量。假设起始坐标为 a，那么在计算导体所在区域时，只要将导体坐标减去 a 后再进行计算就可以了。

表 3-1 描述了将版图所有导体放置到所属区域内的算法。

表 3-1 导体分组插入算法

输入：版图导体集合 输出：每一层导体分组结果

1. For each oaPathSeg in the Layout
2. L = oaPathSeg.getLayerNum(); //得到层号 L
3. layer = Layout.get_layer(L); //得到第 L 层
4. if oaPathSeg is horizontal
5. p = oaPathSeg.getY();
6. else if oaPathSeg is vertical
7. p = oaPathSeg.getX();
8. endif
9. section = layer.get_section(p/w); //w 为 section 宽度
10. section.insert(oaPathSeg);
11. EndFor

3.4 导体的查找操作

对导体的查找操作包括两个方面：一，查找主导体两侧的邻近导体；二，在与导体走向平行的区间内查找所包含的导体。

表 3-2 所示算法描述了获取主导体两侧若干邻近导体的过程。

表 3-2 查找邻近导体算法

输入：布线层 layer，垂直方向导体 p，最大间距 s
输出：p 两侧距离 s 范围内的全部导体的集合 p_set

```
1. k = p.getX();
2. a = (k-s)/w;                          //w 为划分区域宽度
3. b = (k+s)/w;
4. For each I in [a,b]
5.       section = layer.get_section(I);
6.       if (section is not sorted)
7.            section.sort();
8.       endif
9.       For each seg in section.seg_list
10.           if (seg in [k-s, k+s])
11.                p_set.insert(seg);
12.           endif
13.      EndFor
14.EndFor
```

前面提到过，在计算导体上下层布线面积时，需要在一个与导体走向平行的区间内查找所包含的导体。表 3-3 所示算法描述了这一过程。

表 3-3 平行区间内导体查找算法

输入：垂直方向区域 section，垂直方向区间[a,b]
输出：包含在区间[a,b]内的导体集合 p_set

```
1.  if (section is not sorted)
2.       section.sort();
3.  endif
4.  p1 = section.point_list 中位于区间[a,b]中坐标最小的点
5.  p2 = section.point_list 中位于区间[a,b]中坐标最大的点
6.  For each p in [p1,p2]
7.       p_set.insert(p.parent);   //p.parent 表示端点 p 所在的导体
8.  EndFor
```

3.5 本章小节

本章分析了“布线后寄生电容提取”和“差分线网匹配检测”两种应用所涉及的布线形体处理之间的共同之处，并基于此设计了一种数据结构来处理这两类问题，同时给出了该数据结构的建立及查找算法。第 4 章内容将基于本章所介绍的数据结构。

第 4 章 形体处理在电容提取与差分线网匹配检测上的应用

第 3 章介绍了一种数据结构，用来提供一组基础操作，来处理电容提取与差分线网匹配检测两种应用中所涉及到的布线形体处理。本章将基于这些基础操作，实现对寄生电容的提取和差分线网匹配的检测。

4.1 导体节点电容的计算

4.1.1 导体左右间距的获取

对导体进行分段处理的目的，就是要获得主导体与左右相邻导体的间距。基本思想是，从导体端点出发，画与导体走向垂直的延长线，当对所有导体的两端垂直延长线画完后，每段导体将被这些延长线划分为若干部分，可以认为导体的每个部分之间的左右间距是不同的。如图 4-1 所示，ABCD 四根导体被分段处理，注意 B 的下数第 2、3 两段的左右间距分别相同，类似的，C 的下数 2、3 两段左右间距也分别相同。对于这样属于同一导体、左右间距分别相同且相邻的两段，要合并成一段处理。图 4-1(b)中，BC 的灰色区域表示了合并后结果。

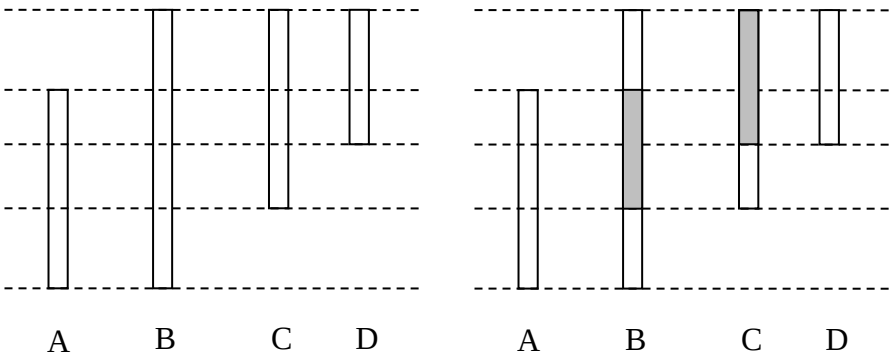

图 4-1　导体分段示意图

下面介绍具体的分段方法。首先要明确一点，这里对导体左右间距的范围是有一定要求的，一般是 1-10 倍线宽，超过 10 倍线宽则认为是无穷远。所以只要需要选取主导体左右各 10 倍线宽范围内的若干导体进行分段处理即可。

画延长线的方式，相当于从与导体走向垂直的方向进行扫描，所以本文后面

将把“延长线”称之为“扫描线”。

先从各导体端点出发，画与导体走向垂直的扫描线，同时要记录下扫描线与导体的关系。由于各导体长度不等，将导体逐个画出的扫描线的相对位置并不是第一时间确定的，所以要将这些扫描线进行排序。因为可能有两段导体的某对端点在同一直线上，所以与同一条扫描线可能连接多段导体，那么也要将这些导体按照相对位置进行排序。扫描线排序完成后，导体位于两相邻扫描线间的部分，就是分割出来的一段，段长就是这两条扫描线间的距离。

根据前面的介绍，导体两端各关联一条扫描线，这时要对导体的两个端点进行区分。将在导体走向上坐标较小的一端称为“始端”，坐标较大的称为“尾端”。每根导体被依次分割的各部分可组成一个序列，每分割出一个部分，就将该部分追加到该导体所对应的序列中。对于同一根导体，如果当前分割部分的左右间距与上一次分割出来的相同，那么就把当前分割部分的长度累加到上一次分割出来的部分，而并不追加新的部分。

在对导体进行分割的过程中，最重要的就是记录每一段的左右间距。两条扫描线之间的区域，可称之为一个“切片”，那么导体位于当前切片中的部分的左右间距，就是与同时位于该切片中的左右相邻导体之间的距离。

现在从第一个切片（位于第 1、2 条扫描线间的区域）开始，将该区域内所含导体加入到一个有序列表中。再将这些导体所分出的段追加到所对应的分段序列中。若有导体的尾端位于该切片中，则将这样的导体从有序列表中移除。接下来处理第二个切片，将始端位于第二个切片内的导体插入至有序列表的正确位置中，再对该有序列表中的导体进行分段操作，然后将尾端位第二个切片中的导体从列表中移除。这样一直进行到最后一个切片。

在对导体进行分段的过程中，不可避免地要涉及到不属于当前线网的导体。对于这样的导体，可以分两种情况来处理。如果这些导体属于后面待提取参数的线网，则对它进行正常的分段操作，并记录间距信息；若这些导体所属的线网不在准备处理之列，也就是说不想提取这些线网的信息，则不对导体进行分段处理，以避免不必要的计算。而对于后面待处理的线网，当需要导体分段信息时，对于已经完成分段的导体，则不需要再次进行分段操作。

图 4-2 所示过程是对图 4-1 中四段导体进行分段的示意。图中由上至下四个区域分别对应图 4-1 中切片自底向上进行滑动中导体的分割情况。图 4-2 左列是四个切片所念导体的变化示意，右列对应着每段导体被分割出来的序列。

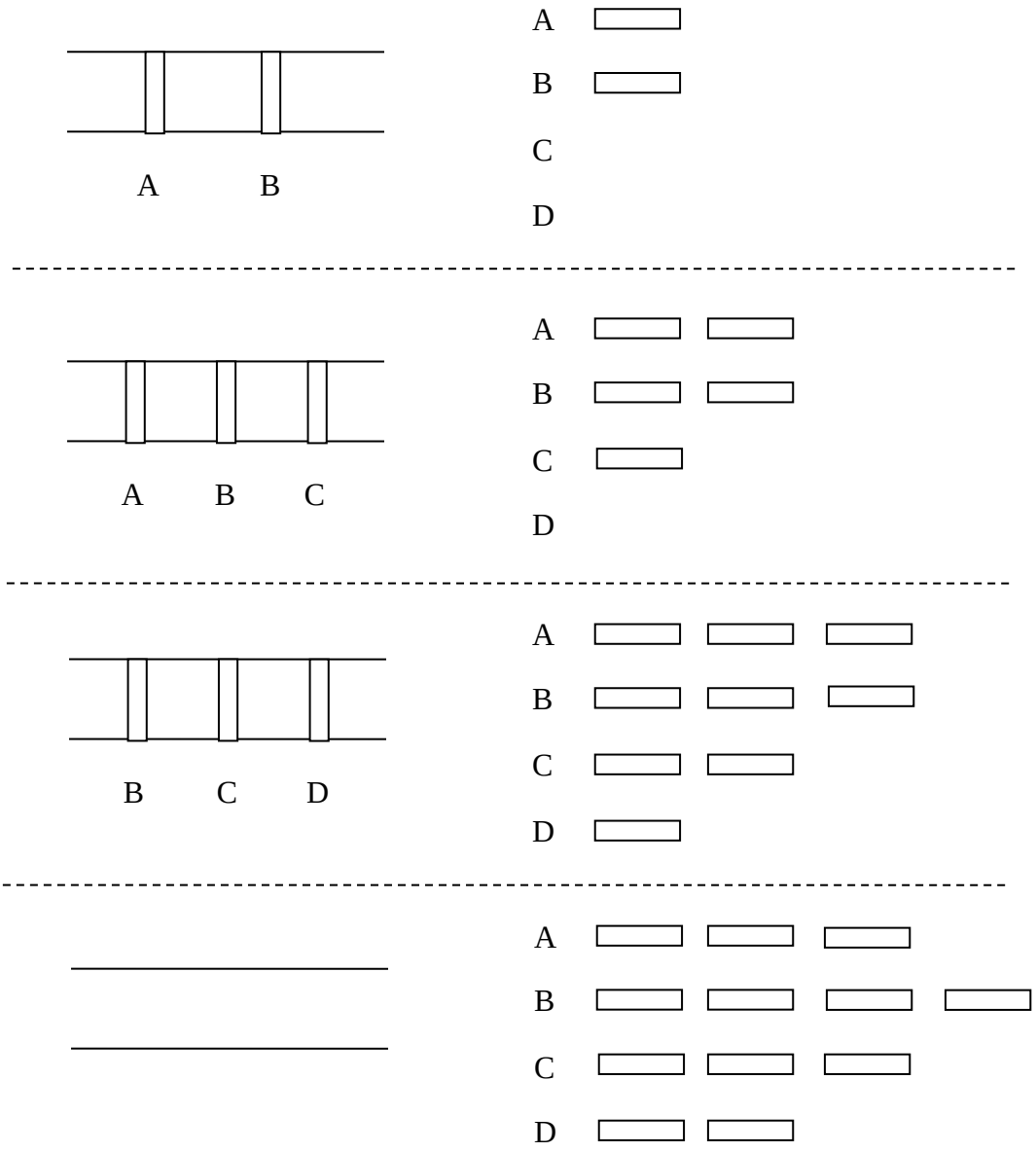

图 4-2 导体分段过程

这里有一个问题应该注意。如图 4-3 所示情况，ABC 是三段分属三个线网的导体，AC 互相不在彼此的最大间距范围中，但 B 分别位于 A 和 C 的最大间距范围内。也就是对 A 进行分段处理时，C 没有包含在内，但包含了 B；而对 C 进行分段时，B 包含在内，而没有包含 A。

假设先对 A 进行分段，显然 B 会被分成两段，一段位于 1 所示区域内，而另一段就是 B 余下的部分。那么再对 C 进行分段时，参加分段操作的 B 应该就是余下的这一段，而不是整段 B 导体。也就是说对导体分段时，应该注意该导体之前已经分段的结果。

以下是相关数据结构的设计：

```
class SubSeg{
    PathSegPeer *parent;
    int p;
    int left;
    int right;
    int length;
};
```

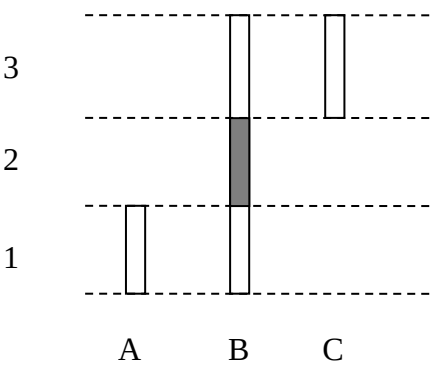

图 4-3 未完全分段的导体

类 SubSeg 表示导体被分成的“段”，其中的 parent 表示该段所属的导体，p 表示它的起始点位置，left 和 right 分别表示该段的左右间距，length 表示该段的长度。

```
class PathSeg Peer{
    oaPathSeg *peer;
    std::vector<SubSeg*> subs;
public:
    bool append(SubSeg *s);
    void end();
};
```

类 PathSegPeer 代表导体，主要用于存储分段后的 SubSeg 序列。在程序中，每一个 PathSegPeer 存在一个与之对应的唯一的 oaPathSeg 对象。其中的 peer 字段表示其所对应的 oaPathSeg 对象，其分段后的序列存储在 subs 中。

```
class SweepLine {
    int p;
    std::vector<PathSegPeer*>   begins;
    std::vector<PathSegPeer*>   ends;
public:
    void insert_begin(PathSegPeer *p);
    void insert_end(PathSegPeer *p);
};
```

类 SweepLine 表示扫描线。字段 p 是该扫描线的坐标，对于水平扫描线，该值是其纵坐标；对于垂直扫描线，该值是其横坐标。begins 用于存储“始端”位于

该扫描线上的导体，ends 用于存储“尾端”位于该扫描线上的导体。

```
class SweepLineList {
    int layer;
    std::vector<SweepLine*> lines;
public:
    void insert_peer(PathSegPeer *p);
    void sort();
};
```

类 SweepLineList 用于将处理当前层的扫描线组织起来，并按照相对位置进行排序。其中 layer 是当前层号，该层所有的扫描线存于 lines 中。

```
class Slab {
    int top;
    int bottom;
    std::vector<PathSegPeer*>    peer_list;
public:
    void add(PathSegPeer *p);
    void remove(PathSegPeer *p);
};
```

类 Slab 表示切片，其中 top 和 bottom 分别表示切片顶部和底部坐标，peer_list 是位于切片中的导体所对应的 PathSegPeer 对象。

表 4-1 列出了对导体进行分段的算法。

表 4-1 导体分段算法

输入：相邻平行导体序列 p_list 输出：序列中导体的分段信息
1. For each p in p_list 2. b = p.begin(); e = p.end(); 3. if (no sweepline at b) 4. create sweepline at b then insert into sweep_list; 5. if (no sweepline at e) 6. create sweepline at e then insert into sweep_list; 7. s1 = sweepline at b; 8. s1.insert_begin(p);

```
9.      s2 = sweepline at e;
10.     s2.insert_end(p);
11.   EndFor
12.   sweep_list.sort();
13.   For each slab in all Slabs
14.     For each peer in sweepline.begin_list
15.       slab.add(peer);
16.     EndFor
17.     For each peer in slab.peer_list
18.       sub = new SubSeg(slab.top – slab.bottom);
19.       peer.append(sub);
20.     EndFor
21.     For each peer in sweepline.end_list
22.       peer.end();
23.       slab.remove(peer);
24.     EndFor
25.   EndFor
```

在提取导体电容的过程中，对于当前的 oaPathSeg，查找到与之对应的 PathSegPeer 对象，即可获得该导体的分段信息。

4.1.2 邻层布线面积比的计算

本文第 2 章介绍过布线面积比的概念。第 3 章提到过相关的计算过程，这里将分两步来详细介绍在计算“布线面积比”的过程，以下将以垂直走向导体为例。

第一步，根据窗口所划定的矩形区域，确定其水平方向的范围内存在哪些导体。第 3 章介绍过，版图已经按与导体相同的走向划分为若干等宽区域。那么根据水平范围的区间坐标，可以确定哪些区域落在该水平范围中。对于完全落在该水平范围中的区域，可以将区域中所含导体全部进行下一步考查；对于部分落在该水平范围中的区域，则将该区域中导体排序后，选出落在水平范围中的导体，以进行下一步考查。

第二步，要确定矩形区域的垂直方向范围内存在哪些导体。在每个等宽区域中，导体的端点是按序排列的，对于现在以之为例的垂直导体，那么这些端点按照垂直方向坐标的大小来排列。这样可以根据给定矩形区域的垂直方向坐标范围

来确定哪些导体的端点包含在该垂直范围内。这里用来检测的区域就是第一步中矩形窗口水平范围所包含的那些区域。

以上两步的目的就是选出落在给定矩形区域内的导体。之后就可以计算每个导体的几何图形与给定矩形区域的重叠面积，把所选全部导体与矩形区域的重叠面积加起来，再除以该矩形区域的面积，就得到了布线面积比。

表 4-2 所示算法将以上两步骤结合，来计算下层布面线面积比，对于上层布线面积比，计算过程是类似的，不再赘述。

表 4-2 计算布线面积比

输入：导体分段结点 sub，版图区域划分数据 layout
矩形窗口 K，其中 K 与 sub 走向垂直的范围为[a,b]，
K 与 sub 走向平行的范围为[c,d]
输出：sub 下层布线面积比 r1

```
1.  layer = sub.get_layer();
2.  if (layer == 1)
3.      r1 = 0;
4.  else
5.      L1 = layout.get_layer(layer-1);
6.      sections = L1.get_sections(a,b);
7.      For each section in sections
8.          paths = section.get_pathseg(c,d);
9.          For each path in paths
10.             p_set.insert(path);
11.         EndFor
12.     EndFor
13. endif
14. For each path in p_set
15.     s += path's area that overlapped with K
16. EndFor
17. r1 = s/((b-a)*(d-c))
```

4.2 电容结点寄生网络的建立

4.2.1 布线形体的连接

对于布线形体的连接，主要的工作就是判断导体与通孔、导体与终端的连接情况。通孔和终端的几何外形都是矩形，而且面积很小，可归结为判断一个小矩形与导体几何外形重叠的问题。

本文的思想是，将位置邻近的导体与通孔（或终端）放在一起进行判断，以避免对远距离无关部分的检测，以节省计算时间。这就涉及到判断一个通孔与哪些导体邻近的问题。

由于通孔的尺寸通常较小，比导体线宽大不了多少，所以可以根据通孔的几何位置来判断它位于哪一个导体分组区域中，该区域内的导体，就是与通孔相邻近的，进一步还可以根据导体的排序情况来确定与该通孔更为靠近的导体，具体方法与计算布线面积比时查找导体的过程类似，不再赘述。

判断导体与通孔（或终端）的重叠，实际就是判断两个矩形重叠的问题。下面介绍本文的判断方法。

将线网中的导体、通孔和终端所对应的矩形接层号分组，分别添加至不同的检测区域，目的是检测每一层上各对象间的连接情况。其中，导体所对应的矩形被添加到导体层号所对应的区域；对于通孔，则分别向它的上下两个层号所对应的两个区域内添加两个矩形；终端的处理与导体相同。

接下来就是对每一层的检测。这个过程主要是判断平面上两个矩形是否相交，再根据一定条件来判断这两个矩形所代表的物体是否连通。如图 4-4 所示，四个矩形 A、B、C、D，A 与 B 显然相交，C 与 D 相邻，但它们有一条边是重合的，所以也判为相交。用一条垂直方向上的直线，从左至右开始依次扫描每一个矩形。每当遇到一个矩形的左边，就将该条边存至一个集合；遇到矩形的右边，则将其所对应的左边从集合中删除。这样，当集合中有两条边按照其垂直方向上的位置，存在重叠时，则可以断定这两条边所属的矩形存在重叠。对于 C 和 D 的情况，可以先添加 D 的左边，再处理 C 的右边，这样即可判断出 C 与 D 相交，而不会将这种情况漏掉。

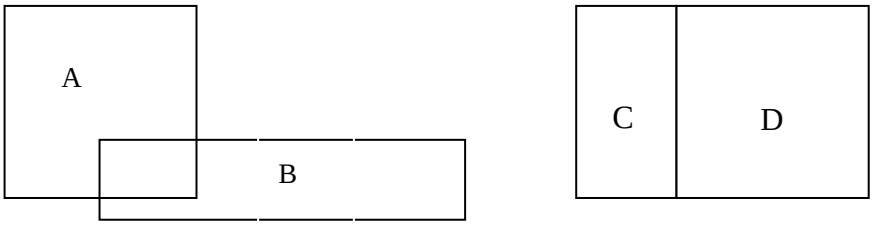

图 4-4 矩形重叠示意图

这样，判断两矩形相交的问题，转化为判断一条直线上的两条线段是否存在重叠的问题。两两检测是最简便的办法，但效率较低。这里采用划分区间的办法。首先将包括全部线段坐标区域的直线，以 1 为单位划分成一个个单位区间。再把每条线段按照它们的坐标范围，分别存至所包含的单位区间内。如图 4-5 所示，假设三条线段的坐标都是整数，其中 A 落在[3,6]区间，则将 A 分别存至[3,4][4,5][5,6]三个单位区间。而对于 B，在存入[5,6]区间的过程中，发现存在线段 A，则认为 A 与 B 存在重叠。类似的过程，可知 C 与 A 和 B 不存在重叠。

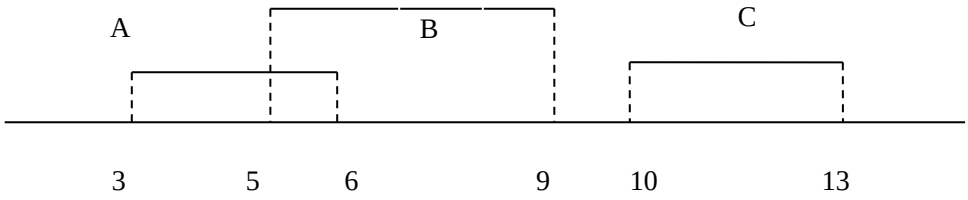

图 4-5 线段重叠示意图

实际处理中，存在一个问题，即线段的坐标取值。目前所处理的坐标，从 0 到 100 万不等，如果划分出 100 万个区间，显然会对空间造成较大浪费。这里可以采取离散化的办法，单个线网的区域跨度不是很大，所以可以将每个坐标值映射至一个较小的整数，再按上面的方法进行处理。

具体的作法是，先统计所有矩形的左边两端在垂直方向上的坐标值，再从小到大排序，然后将第一个值映射为“1”，第二个映射为“2”，以此类推，这样并不会影响线段间重叠的判断。

两个矩形相交，并不代表这两个矩形所表示的物体相连。如图 4-6 所示，三段导体，A、B、C，显然它们两两相交，但按照实际的连线规则，两段导体的

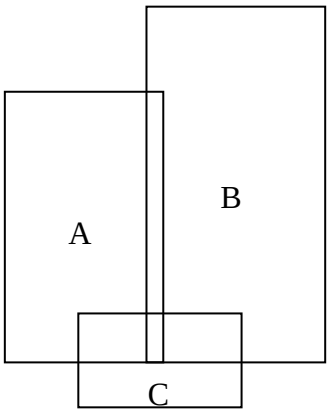

图 4-6

中线必须首尾相连，所以这里应该是 A 与 C 相连，C 与 B 相连，注意 A 与 B 的中线不相连，所以 A 与 B 所代表的导体也不是相连的。

比较困难的是导体与终端的连接判断。如图 4-7 所示，点填充的矩形代表终端，中间打“X”的代表通孔，斜线填充的代表导体。

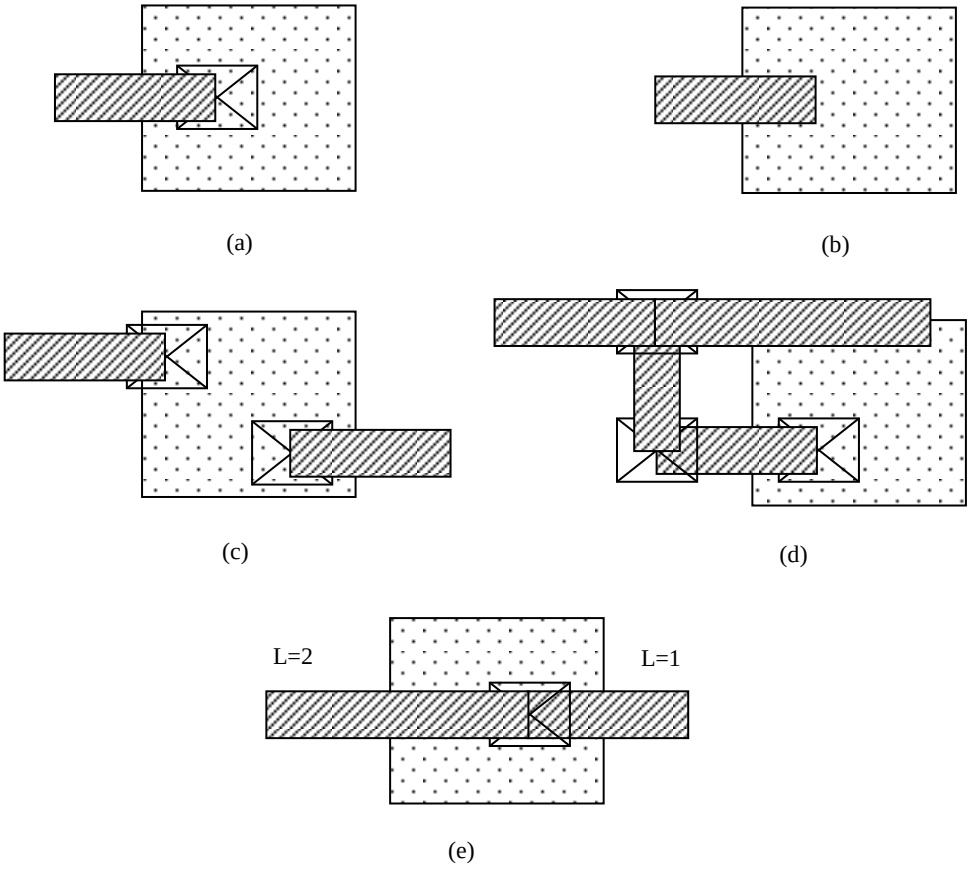

图 4-7 导体、通孔和终端连接情况示意图

(a)的情况最典型，一段导体通过一个通孔连至终端。(b)与(a)不同的是，这段导体没有经过通孔，而是直接连在了终端上，而且这段导体的层号不一定是 1，也可能是 2 或 3，或其它。在(c)中，一个终端上连接了两个通孔，代表两个通路。

而对于(d)，假设信号从左上方的导体的左侧传进来，那么它将分成两个支路，一支走向这个终端，另一支沿右上方的导体向右前进。这里虽然右上方的导体与终端区域在垂直方向上存在重叠，但实际它们是不相连的。

(e)中，左边的导体层号为 2，右边的导体层号为 1，中间的通孔连通 1 和 2 层，若假设信号从左侧进来，则在通孔处分支，一支走向该终端，而另一支沿右边的导体向右前进。

综合以上情况，单纯地根据图形是否相交来判断终端是否与某些通路相连，

是不准确的。经过分析可以发现，尽管同时与一个终端存在相交的图形会有很多，但它们都分别属于某一个“连通分量”。

这里解释一下“连通分量”，类似于图论中的“连通分量”，此处的“连通分量”是指，不通过终端发生连接的所有图形。也就是说，对于一下正确连接的线网来讲，参加检测的所有四类图形应该是连通的，所有图形能够形成一个“连通图”。而这里将这个“连通图”从终端处切割成彼此不连通的分量，即形成了多个“连通分量”。如图 4-7 中的(c)，这个终端即可将整个线网切成两个“连通分量”。

那么实际判断时，将与终端的图形相交的所有图形按所在的“连通分量”分组，每一组取层号最低的对象与这个终端相连，并取消终端与其他对象的连接关系。当所有的连接关系判断完成后，一个连通图就建立好了。

如图 4-8 所示，(a)、(b)、(c)、(d)和(e)分别是图 4-7 五个图所对应的连通分量示意，图 4-7 中每个图形的连接关系，就是图 4-8 中的连线。其中 P 代表导体，V 代表通孔，T 代表终端。可以看到，(a)(b)(d)(e)四图，分别只包含一个连通分量，而(c)图中，终端将之分为两个连通分量。

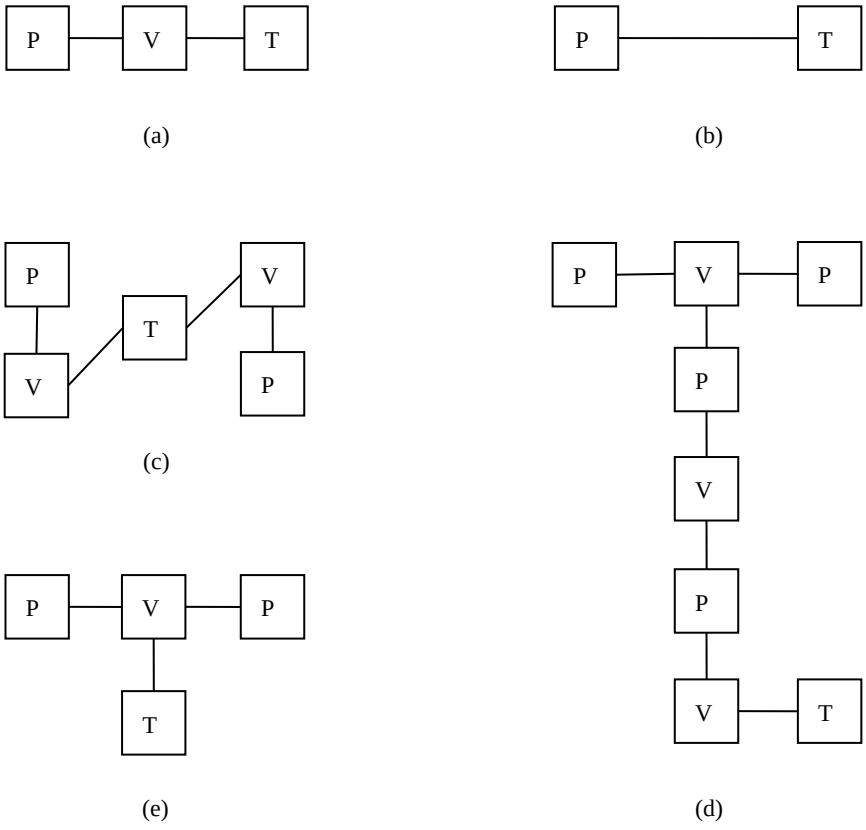

图 4-8　连通分量示意图

4.2.2 节点电容寄生网络的建立

在前面的过程中，代表线网中各对象的图形彼此之间已建立了连接关系，将每个图形看作一个结点，那么整个结构就是一个连通图。问题是从哪里开始遍历，也就是说信号方向如何确定？

目前所处理的都是单 driver 线网，也就是一个线网只有一个信号的出发点，那么找到 driver 所对应的结点，也就找到了遍历的起始点。信号的前进有两个特点，一，信号是单向的，也就是说对于同一条通路，信号只存在一个走向；二，信号会走向尽可能多的通路，也就是说如果从当前结点出，存在多于一个的通路，那么信号会流向这所有的通路。基于以上两个特点，可以从 driver 对应的结点出发，对这个连通图进行广度优先遍历，它很好地模拟了信号的流动过程。

在遍历过程中，依次创建 oaNode 结点和电阻器件，并计算每段导体的电容。用遍历的走向来模拟信号的走向，遍历结束后，整个线网的寄生网络就建立起来了。

由于实际信号的走向是固定的，于是可以将每个结点看作一个通路，并增加两个域，fromNode 和 toNode，分别代表信号进入和离开的 oaNode 结点。如图 4-9 所示，箭头方向为信号走向，信号从 fromNode 进入，从 toNode 离开。

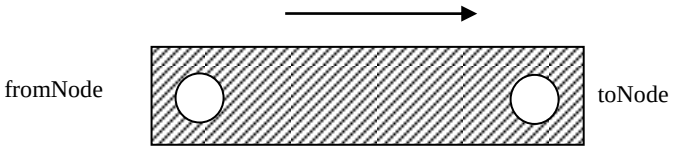

图 4-9 信号走向示意图

i). 遍历过程中各类结点的处理

首先初始化 Driver 结点。如图 4-10 所示，建立一个 oaNode 并赋给 driver 结点的 fromNode 域，并将此 oaNode 关联至这个作为 driver 的终端。将 toNode 置为空。图中打斜线的圆形表示关联至终端的 oaNode。

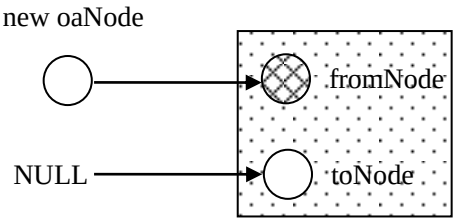

图 4-10 driver 结点初始化

接下来是各类连接情况的判断。

从终端至通孔。创建两个 oaNode，位置为该通孔矩形边框的中点，分别赋给通孔结点的 fromNode 和 toNode。在 terminal 的 fromNode 和通孔的 fromNode 之间创建一个电阻值为 0 的电阻器件，之所以这样处理，是因为终端的 fromNode 的位置，可能与通孔的 fromNode 位置不同，所以要以一个 0 电阻相连。在通孔的

fromNode 和 toNode 之间创建电阻器件 oaResistor，电阻值为这个通孔本身的电阻，可以从.itf 文件中查到相应数值来计算。如图 4-11 所示，存在三种连接情况，虽然通孔的中点所处位置不同，但可以作为同一种情况处理。右下方的斜线矩形是通孔的侧视图，信号走向为从上到下，或从下到上，这里假设为从上到下。

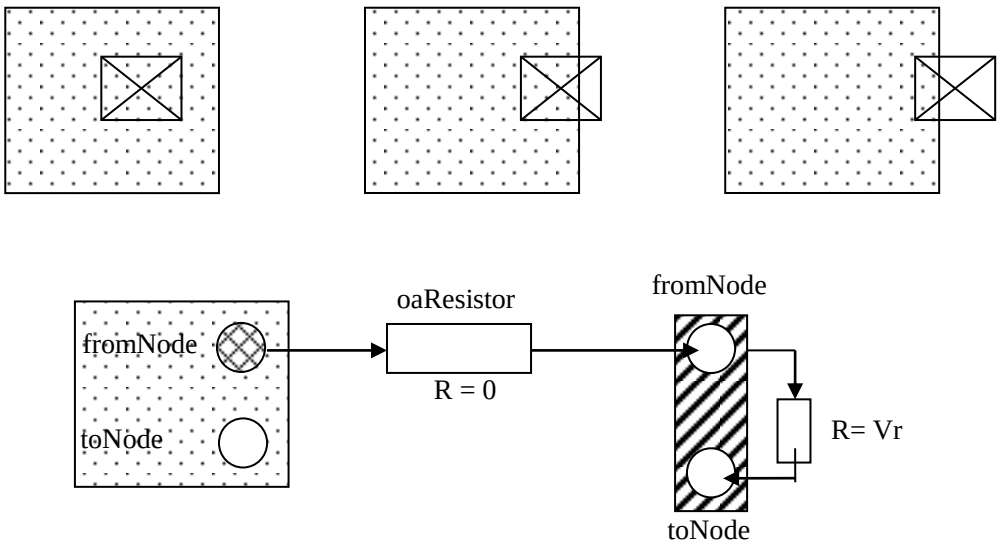

图 4-11 通孔与终端连接示意图

从终端至导体。如图 4-12 所示，在导体的首尾两端各创建一个 oaNode，分别作为 fromNode 和 toNode。在终端的 fromNode 和导体的 fromNode 之间创建一个

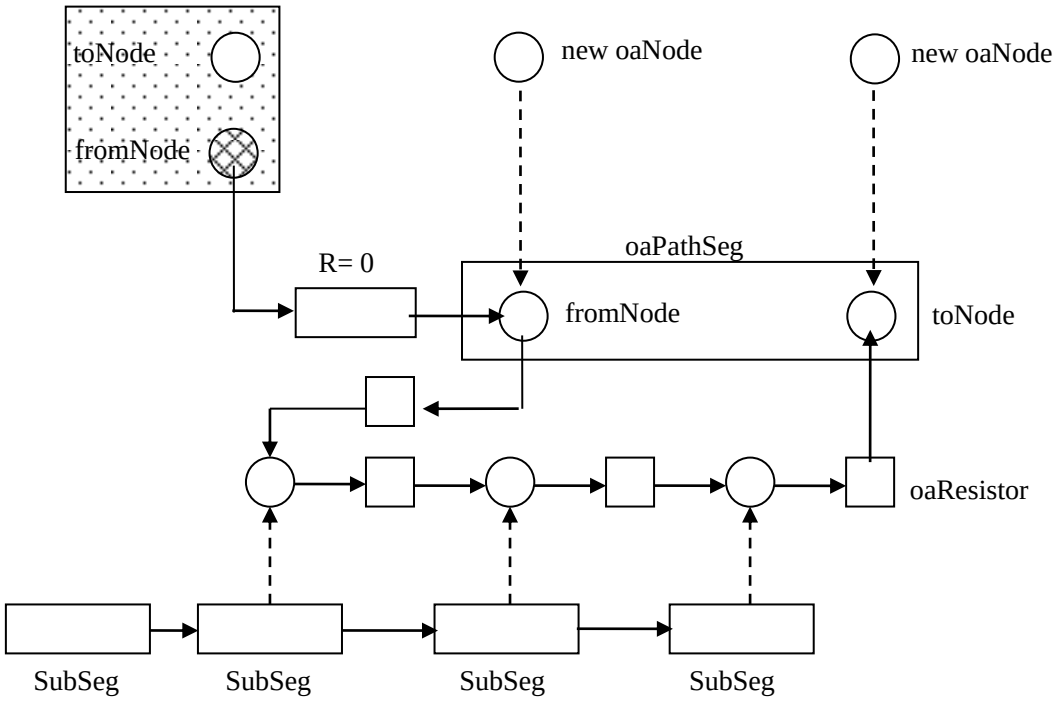

图 4-12 导体分段结点连接

电阻值为 0 的电阻器件，原因与终端和通孔相连情况类似，即导体的 fromNode 位置可能不同与终端的 fromNode 位置。

执行扫描线算法后，每段导体被切割为若干段 SubSeg，并组成一个链表。找到当前导体所对应的 SubSeg 链表，在每段 SubSeg 的起始位置创建一个 oaNode，并按照遍历过程中所确定的信号走向，在每两个相邻的 oaNode 间建立电阻器件，电阻值可以根据 SubSeg 的长度及层号信息，以及相应的从.itf 文件中获取的方块电阻值来计算。注意第一段 SubSeg 对应的 oaNode 应该是导体的 fromNode，所以不用新建相应的 oaNode。而每段 SubSeg 的电容值则根据相应的物理信息，按照相应的公式来计算，在此不赘述。最后将这串 oaNode 的首尾分别与当前导体的 fromNode 和 toNode 以电阻器件相连，电阻值按前面所述方法计算。

通孔和导体类型的结点，两两相连，共 4 种组合，都是将下一个结点的 fromNode 设置为前一个结点的 toNode，再根据信号走向确定下一个结点的 toNode，并在 fromNode 和 toNode 之前建立电阻器件，再计算相应电阻和电容值，过程与前面两种情况类似。

从导体至终端。如图 4-13 所示，在(a)中，信号由导体左侧进来，到终端终止。

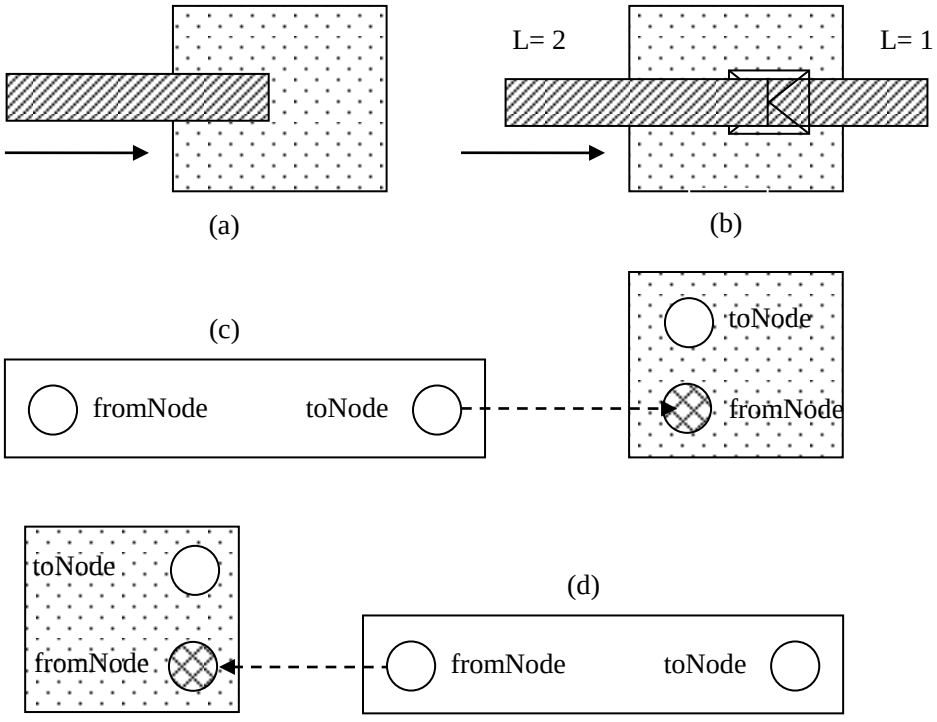

图 4-13 导体与终端连接

在这里，应该将导体的 toNode 作为终端的 fromNode。而对于(b)，信号从 2 层的导体进来，经 1、2 层间的通孔来到 1 层，一部分至终端，另一部分沿 1 层的导体

向右前进。按照之前关于终端连接情况的判断，信号应该是从通孔至导体，再到终端。所以这里应该将右段导体的 fromNode 作为终端的 fromNode。于是要做一下判断。

首先判断导体的 toNode 的位置是否在与终端相连的区域内，若在，则将终端的 fromNode 设置为导体的 toNode，如(c)所示。否则设置为导体的 fromNode，如(d)所示。因为若一个导体与一个 terminal 相连，其 fromNode 和 toNode 必有一个在终端的连接区域内。

对从通孔至终端的处理过程和导体至 terminal 一样，不再赘述。

ii) 对结点连接处理的总结

前面讲过，将每个结点看作一个通路，每个通路存在一个信号的流入点 fromNode，和一个信号的流出点 toNode。依照遍历的顺序，将下一个结点的 fromNode 设置为上一个结点的 toNode，也就是将它们两个设置为同一个 oaNode，那么结点之间就连接起来了。

如图 4-14 所示，对 terminal 类型结点的处理大致分三种情况，一，终端作为 driver；二，终端作为 receiver，并且没有信号流出的端口，也就是说信号到这里就终止了；三，终端作为 receiver，但存在一个信号流出的端口。

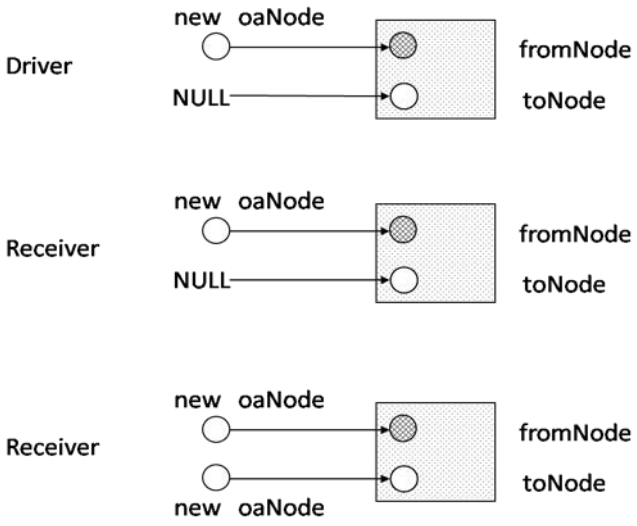

图 4-14 终端结点的处理

对于前两种情况，统一将 fromNode 赋一个 oaNode，而将 toNode 留空，设置为 NULL。第三种情况，fromNode 和 toNode 个赋予一个 oaNode，在两个 oaNode 之间创建一个电阻值为 0 的 oaResistor 器件。

对于导体和通孔类型结点的处理，上一小节已经讲过，不再赘述。

4.3 差分线网匹配检测的实现

4.3.1 差分线网匹配检测算法

现有某差分对包含线网 A 和 B，计算 A 相对 B 的匹配长度，可归结为这样的过程：对于线网 A 中每一段给定的导线 L，在 B 中查找与之相匹配的导线，并按第 2 章介绍的规则来计算导线 L 相对于线网 B 的匹配长度，最后把线网 A 所有导线的匹配长度加起来，即为 A 相对于 B 的匹配长度。

为了便于在某线网中查找指定的导线，需要将该线网的导线位置信息按一定方式组织起来。本文的作法是，将导线抽象为线段，并将同一层的平行线段按位置排序（对于水平导线，可以按其在垂直方向上的坐标由小到大进行排序，对垂直导线则按其在水平方向的坐标来排序），组成一个序列，每一个序列对应一个层号，最后将线网中若干个这样的序列放入一个名为“查找库”的集合，每一个线网对应一个“查找库”。“查找库”的具体组织结构将在后面介绍。记录导线位置信息的同时，还应记录该导线末端所连接通孔的信息。这样方便于在计算导线匹配长度的同时，判断通孔的匹配情况。

表 4-3 算法描述了计算线网 A 相对于 B 的匹配长度以及通孔匹配数量的过程。

表 4-3 匹配长度统计算法

输入：一对差分线网 A 和 B，第 i 层的间距为 Hi
输出：线网 A 相对于 B 的匹配长度 Lt，通孔匹配数量 Vt
1. 建立线网 B 的查找库
2. Lt = 0，Vc = 0
3. 将 A 中所有通孔标记为“未检测”状态
4. For 线网 A 的每段导线 La（层号为 i）
5. if (B 中第 i 层存在与 La 平行且间距为 Hi 的导线 Lb)
6. 计算 $l_{overlap}$ 和 l_{offset}
7. Lt = Lt + $l_{overlap}$ + l_{offset}
8. For La 所连接的每个通孔 v
9. if (v 的状态为“未检测”)
10. if (Lb 上存在与 v 匹配的通孔)
11. Vc = Vc + 1
12. endif
13. 将 v 标记为“已检测”

```
14.                endif
15.             EndFor
16.        endif
17. EndFor
```

计算线网 B 相对于 A 匹配长度的过程类似，只是其中不用再判断通孔的匹配情况，因为在计算 A 相对于 B 的匹配长度过程中，已经得出了两线网间匹配通孔的对数。

4.3.2 查找库的组织与实现

查找库的作用是将所有导线按位置信息组织起来，用以实现在指定区域内查找导线。以水平方向的导线为例，将所有水平导线所在的直线，按照各自的纵坐标由小到大排序。对于在同一直线上的不同导线，将导线各自的端点按照在直线上的相对位置排列，也就是按照端点的横坐标由小到大进行排列。这样，对于任一条水平导线而言，可利用二分查找快速获知在给定间距的两侧是否存在包含导线的直线。找到这样的直线后，再在原导线投影到此直线上的区间中，查找存在哪些导线。

考虑到可能有两条或多条导线的中心线位于同一条直线上，那么这些由导线所抽象成的线段可以如图 4 的方式来表示。图中两条水平线段位于同一条直线上，记它们端点的横坐标从小到大分别为 x1、x2、x3、x4，纵坐标为 y，对于每条线段，将两个端点中横坐标较小的点的类型记为 begin，另一个点的类型记为 end。

如图 4-15 中所示，两条线段的四个端点信息构成一个向量，向量中每个元素为一个二元组(x,type)，其中 x 是该端点的横坐标，type 为该点的类型(begin 或 end)。由于所有端点的纵坐标相同，那么一条直线只需要保存自己任一点的纵坐标即可。并且将这条直线上的所有端点按照横坐标进行升序排序，以方便于后面的查找操作。那么排序后，对于同一条线段上的两个端点，begin 类型的端点一定排在 end 类型之前。也就是说，对于任一个端点，根据它的类型信息，可以确定它和前、后哪个端点构成一条线段。

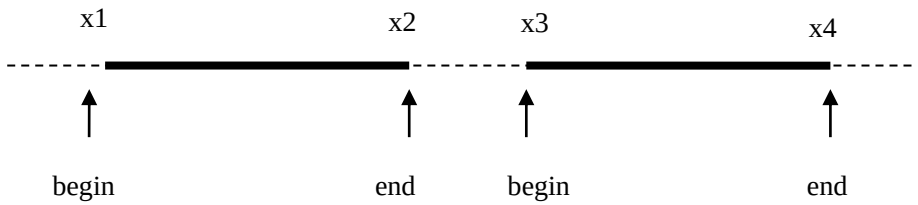

图 4-15 线段的表示方式

由于同一个差分对的两条线网不会有在同一位置重合的导体，那么可以将两条线网的导体所对应的线段存储在一个查找库中。

下面介绍具体的查找操作。要找到一根导线的“匹配导线”，首先查找给定间距的位置上，是否存在平行的导线。例如一条给定的水平方向线段，纵坐标为 y，对于给定的间距 gap，分别在 y-gap 和 y+gap 两个位置查找是否有对应的直线存在。因为所有导体所在的直线已经按照纵坐标的大小排序，所以可直接进行二分查找。

找到这样的直线后，在直线上做原线段在 y 方向上的投影。如图 4-16 所示，上方的线段在下方直线上的投影区域为[a,b]，其中包含了两条线段。

查找[a,b]内所包含的线段，可以按照以下方法进行。已知线段的端点按照横坐标排序。那么首先在这些端点的横坐标序列中，对 a 进行二分查找。那么当序列中包含 a 时，即可确定 a 在序列中的位置；若序列中不包含 a，那么可以确定 a 位于序列中哪两个元素之间。按同样的方式确定 b 在序列中的位置。这样也就可以确定哪些端点包含在区间[a, b]中。对于任一个包含在区间[a, b]中的端点 p，它所在的线段，就被认为包含在这个区间中。

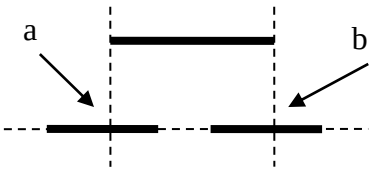

图 4-16 线段在直线上的投影

4.4 实验结果

这里采用本节介绍的方法对一些电路设计例子进行了测试，这些电路都采用了 180 纳米工艺。程序基于 OA 所提供的 C++类库开发，运行操作系统为 CentOS release 4.4，服务器 CPU 为 8*Intel® Xeon™，主频 3.00GHz.

表 4-4 为寄生电容提取的测试结果，用于比较的数据来自 Synopsys 公司开发的 Astro.

表 4-4 与 Astro 的运行结果对比

测试例子	线网数量	平均电容(pf)	平均相对误差(%)	cpu 时间
SBOX	170	0.045pf	1.04	0.68s
MMC	10600	0.025pf	4.26	3.26s
CFHC	12480	0.020pf	4.41	5.20s
ORCA	23655	0.025pf	16.5	11.4s

对于差分线网匹配检测的算法，本文采用的实验例子是 Synopsys 公司 AMSG 小组用于测试的一套混合信号电路。其中采用相邻层垂直正交的布线方式，包含 20 对差分线网。DP-REPORT 是依据本文方法所开发的用于评价差分线网布线质量的软件，它作为 NET-REPORT 的一个子程序来运行。NET-REPORT 是 Synopsys 公司 Galaxy Custom Designer 的一个程序模块，主要用于统计布线后的线网信息，其中包括线网总长、各层线长、线网的 Manhattan 长度、通孔使用情况等等。如果 NET-REPORT 发现当前处理的线网是差分线网，则会在完成前面所提到的基本任务后，调用 DP-REPORT 对该差分线网进行检测，并报告其布线质量。DP-REPORT 会报出两组数据，一组是差分线网的匹配长度，另一组是通孔的匹配个数。

表 4-5 列举了 6 对差分线网匹配情况的结果。其中同一差分对内的两条线网名字的开头大写字母相同。DP-REPORT 所计算得到的匹配长度，全部小于或等于线网总长，且通孔的匹配数量也小于或等于通孔总数。进一步通过 Galaxy Custom Designer 的图形界面工具观察了一部分线网的实际布线情况，并结合第 2 节所介绍的规则来判断每一层的匹配长度以及匹配的通孔数量，发现 DP-REPORT 的计算结果与事实相符。

针对含 4000 多条线网的最大测试电路，我们在表 4-6 中列出了 DP-REPORT 和 NET-REPORT 两程序的运行时间，其中 NET-REPORT 一列给出的是其统计所有差分线网所用的时间（不是统计全部线网所用时间）。从实验数据看出，DP-REPORT 占 NET-REPORT 的运行时间不到 5%。

表 4-5 六对差分线网布线的检测结果

线网	总长（um）	通孔数量	匹配长度（um）	通孔匹配数量
A1	413.970	3	406.075	3
A2	421.125	3	406.075	3
B1	554.035	4	541.6	4

B2	557.450	4	544	4
C1	468.770	7	452.735	6
C2	463.785	6	445.535	6
D1	551.665	3	543.795	3
D2	552.150	3	543.795	3
E1	144.100	2	141.16	2
E2	145.205	3	141.16	2
F1	333.785	4	324.675	3
F2	341.165	4	324.675	3

表 4-6 DP-REPORT 与 NET-REPORT 运行时间对比

NET-REPORT (s)	DP-REPORT (s)	DP-REPORT 时间所占百分比
15.46	0.74	4.79%

4.5 本章小节

本章在对第 3 章所介绍数据结构的运用基础上，实现了对寄生电容的提取和差分线网匹配的检测。在程序的实现上，第 3 章内容被封装为一个模块，供本章算法调用。这样易于实现代码分离，降低模块间的耦合，提高程序的可维护性。在最后的实验中，用本文方法计算出的电容结果与 Astro 较为接近，时间效率也比较好；通过统计 DP-REPORT 占主程序运行时间百分比可以看出，本文方法对差分线网匹配的检测效率也比较高。

第 5 章 总结与展望

5.1 工作总结

本文所提出的方法主要是对集成电路版图的布线形体处理，主要面向 2.5 维寄生参数提取和混合信号电路中差分对的布线质量检测。经过对以往“一次性处理全部版图形体”方法的分析，针对其在对一部分线网进行计算所占时间过长的不足，本文提出一种基于“按需计算”思想的改进算法，通过延迟对部分布线形体的处理，避免了在对一部分线网处理的过程中无关形体的相关处理，节约了处理时间。同时，又能够适应对全部线网的处理需求。

本文方法的主要思想是，将版图各层平行导体进行分组，若布线密度均匀，则每层所含导体数量会比较接近。当需要处理某根导体的数据时，再将该组导体进行排序，以得到相邻导体的信息。也就是说并不预先处理所有导体，而是等到真正需要进行计算时，再行处理，如此可避免对大部分无关导体的处理，节省计算时间。

接下来本文以该方法为基础，详细介绍了对寄生电容的提取和对差分线网布线检测的实现方法。最后经实验证实，该方法在保证一定计算准确度的同时，时间上的效率也比较高。

5.2 工作展望

在对分组导体的查找中，本文的方法是先将各导体端点进行排序，再以区间定位的方式进行查找。进一步的想法是，对于导体数量比较多的分组，可以用线段树的形式将各平行导体组织起来，以提高查找效率。

此外，若布线密度不均匀，导体分组后，可能会存在组间导体数量差别较大的情况，会造成对过多无关导体的处理。所以今后可以在将导体进行分组中，使各组导体数量尽可能地平均。

参考文献

[1] Johnson H. W. “High-Speed Signal Propagation: Advanced Black Magic[M]”. New Jersey: Pretice Hall PTR, 2003.

[2] 蔡国发，章杰，林培杰，等. 差分对信号完整性分析. 电子测量技术. 2012(1): 38-41.

[3] 唐世悦，高新成. 有限地平面上差分对共模噪声的分析. 电讯技术. 2008(9): 33-37.

[4] 王超，宋克柱. 印刷电路板差分对布线位置不对称引起的共模辐射仿真研究. 中国科学技术大学学报. 2007(11): 1393-1398.

[5] Matteo Cocchini, Jun Fan, Bruce Archambeault, James L. Knighten, Xin Chang, James L.Drewniak, Yaojiang Zhang and Samuel Connor. “Noise Coupling Between Power/Ground Nets Due To Differential Vias Transitions in a Multilayer PCB”. IEEE International Symposium on Electromagnetic Compatibility, August 2008, pp. 1-6.

[6] Umakanta Choudhury, A. Sangiovanni-Vincentelli. “Constraint Generation for Routing Analog Circuits”, in Proc. Design Automation Conference (DAC), 1990, pp. 561-566.

[7] Jia-Wei Fang, Kuan-Hsien Ho, and Yao-Wen Chang. “Routing for Chip-Package-Board Co-Design Considering Differential Pairs”, IEEE/ACM International Conference on ICCAD, 2008, pp. 512-517.

[8] Umakanta Choudhury and Alberto Sangiovanni-Vincentelli. “Automatic Generation of Parasitic Constraints for Performance-Constrained Physical Design of Analog Circuits”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1993, pp. 208-244.

[9] OpenAccess API Tutorial Eighth Edition (OA 2.2 DM4), Published by Silicon Integration Initiative, Inc, 2009.

[10] 公维冰, “数字集成电路物理设计阶段寄生提取与时延计算”, 兰州大学研究生学位论文, 2010.

[11] S. Napper, Technical white paper on RC extraction, EPIC Design Technol., 1995.

[12] T. Mitsuhashi and K. Yoshida, A resistance calculation algorithm and its application to circuits extraction, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Vol. 16, No. 3, pp.337-345, 1987.

[13] K. Nabors and J. White, Multipole-accelerated 3-D capacitance extraction algorithms for structures with conformal dielectrics, in Proc. 29th ACM/IEEE Design Automation Conf., 1992, pp.710-715.

[14] 喻文健, 王泽毅, 三维 VLSI 互连寄生电容提取的研究进展, 计算机辅助设计与图形学学报, 第 15 卷, 第 1 期, pp.21-28, 2003 年.

[15] 喻文健, 王泽毅, 王玉刚等, 一种可适应复杂互连电容结构的边界元形体处理方法, 半导体学报, 第 25 卷, 第 2 期, pp.214-220, 2004 年.

[16] N. K. Verghese and D.Allstot, SUBTRACT: program for the efficient evaluation of substrate parasitic in integrated circuits, in 1995 IEEE/ACM Int. Conf. Computer-Aided Design, 1995, pp.194-198.

[17] M. Bachtold, Efficient 3-D Computation of Electrostatic Fields and Forces in Microsystems, Ph.D. Thesis, Phys. Electron. Lab., ETH Zurich, Zurich, Switzerland, 1997.

[18] J.P.Hwang, REX-A VLSI parasitic extraction tool for electro migration and signal analysis, ACM/IEEE Design Automation Conference, pp. 717-722, 1991.

[19] X. Aragones, J. L. Gonzalez, and A. Rubio, Analysis and Solutions for Switching Noise Coupling in Mixed Signal ICs. Norwell, MA: Kluwer, 1999.

[20] Wenjian Yu,Zeyi Wang and Xianlong Hong, Preconditioned multi-zone boundary element analysis for fast 3D electric simulation, Engineering Analysis with Boundary Elements, 28(9), pp. 1035-1044, 2004.

[21] 田洪宇, 余志平, 田立林, 张文俊, 混合信号电路的衬底电阻网络模型, 微电子学, 第 33 卷, 第 01 期, pp.1-4, 2003 年 2 月.

[22] 王习仁, 喻文健, 王泽毅, 三维互连电阻解析与边界元耦合提取方法, 清华大学学报, 第 44 卷, 第 9 期, pp.1277-1281, 2004 年.

[23] R. Singh, A review of substrate coupling issues and modeling strategies, in Proc. IEEE Custom Integrated Circuits Conf., 1999, pp. 491–499.

[24] D. Su, M. J. Loinaz, S. Masui, and B. A.Wooley, Experimental results and modeling techniques for substrate noise and mixed-signal integrated circuits, IEEE J. Solid-State Circuits, vol. 28, pp. 420–430, Apr. 1993.

[25] M. Glegharbour and J. M. Drake, Calculation of multi terminal resistance in integrated circuits, IEEE Trans. Circuits Syst., Vol.33, No.4, pp. 462-465, 1986.

[26] G. Costache, Finite element method applied to skin effect problems in strip transmission lines, IEEE Trans. Microwave Theory Tech., vol. MTT-35, pp. 1009–1013, Nov. 1987.

[27] A. E. Ruehli and P. A. Brennan, Efficient capacitance calculations for three dimensional multi conductor systems, IEEE Trans. Microwave Theory Tech., vol. MTT-21, Feb. 1973.

[28] E. Dengi and R. Rohrer, Hierarchical 2-D field solution for capacitance extraction for VLSI interconnect modeling, in Proc. 34th ACM/IEEE DAC, Anaheim, CA, June 1997, pp. 127–132.

[29] F. J. Clement, E. Zysman, M. Kayal, and M. Declercq, LAYIN: Towarda global solution for parasitic coupling modeling and visualization, in Proc. IEEE 1994 Custom Integrated Circuits Conf., 1994, pp. 537–540.

[30] W. Sun, W. Hong, and W. W. Dai, Resistance extraction using super-convergence accelerated boundary element method, Asia-Pacific Microware Conference, Vol.3, pp. 1061-1064, 1997.

[31] A. J. Kemp and J.A. Pretoriu, The generation of a mesh for resistance calculation in inter-grated circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,Vol.7, No. 10, pp. 1029-1037, 1988.

[32] 郑蓝舟，喻文健，尹航，王泽毅，芯片级三维寄生电容的并行提取算法，计算机辅助设计与图形学学报，第 20 卷，第 11 期，pp. 1396-1402, 2008 年。

[33] Z. Wang and Q. Wu, A two-dimensional resistance simulator using the boundary element method, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Vol.11, No.4, pp. 497-504, 1992.

[34] U. Choudhury and A. Sangiovanni-Vicentelli, Automatic generation of analytical models for interconnect capacitances, IEEE Trans. Computer-Aided Design, vol. 14, no. 4, pp. 470–480, Apr. 1995.

[35] Synopsys, Raphael Reference Manual, Version 2003. 09, Mountain View, California, September 2003.

[36] W. Hong, W. K. Sun, Z. H. Zhu, H. Ji, B. Song, and W. Dai, A novel dimension reduction technique for the capacitanceextractionof3-D VLSI interconnects, IEEE Trans. Microwave Theory Tech., vol.46, no.8, pp. 1037–1044, 1996.

[37] KAO William H., LO Chi-Yuan SR, BASEL Mark, SINGH Raminderpal, Parasitic extraction: Current state of the art and future trends, in Proc. of IEEE Institute of Electrical and Electronics Engineers, Vol.89, No.5, pp. 729-739, 2001.

[38] N. Verghese, D. Allstot, and M. Wolfe, Verification techniques for substrate coupling and their application to mixed-signal IC design, IEEE J. Solid-State Circuits, vol. 31, pp. 354–365, Mar. 1996.

[39] N. Verghese, T. Schmerbeck, and D. Allstot, Simulation Techniques and Solutions for Mixed-Signal Coupling an Integrated Circuits, 2nd ed. Norwell, MA: Kluwer, 1995.

[40] L. Ladage andR. Leupers, Resistance extraction using a routing algorithm, ACM/IEEE Design Automation Conference, pp. 38-42, 1993.

[41] Wenjian Yu, Zeyi Wang and Jiangchun Gu, Fast capacitance extraction of actual3-D VLSI interconnects using quasi-multiple medium accelerated BEM, IEEE Trans. Microwave Theory Tech., 51(1), pp. 109-120, 2003.

[42] W. Yu and Z. Wang, Fast quasi-multiple medium method for 3-DBEM calculation of parasitic capacitance, Computers and Mathematics with Applications, Vol. 45, No. 12, pp. 1883-1894, 2003.

[43] Xiren Wang, Deyan Liu, Wenjian Yu and Zeyi Wang, Improved boundary element method for fast 3-D interconnect resistance extraction, IEICE Trans. on Electronics, Vol.E88-C No.2 pp. 232-240, Feb. 2005.

致　谢

衷心感谢喻文健老师，他广博的知识和严谨的治学态度深深影响着我。我所做课题的每一个环节都离不开他的悉心指导与点拨。在我思想上出现偏差时，喻老师及时、耐心地开导我，使我回到了正常轨道上。正是在他的帮助下，我得以克服困难，完成课题的研究。

感谢许静宇博士，不仅为我提供了难得的实习机会，让我在真正的企业环境中接受锻炼，而且在项目开发过程中对我耐心地指导，帮助我解决一个又一个的难题。还要感谢孙海洋学长，他不仅保持着哈工大人踏实严谨的作风，而且有着深厚的技术功底，指点我解决了很多关键性的技术问题。

感谢实验室的同学们，他们在学习和生活上给予了我很大的帮助，他们勤奋好学的精神也一直激励着我。

最后，感谢我的父母和朋友一直以来给我的支持与鼓励。

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：

个人简历、在学期间发表的学术论文与研究成果

个人简历

1985 年 7 月 24 日出生于黑龙江省五常市。

2004 年 8 月考入哈尔滨工业大学软件学院，2008 年 7 月本科毕业并获得学士学位。

2009 年 9 月考入清华大学计算机科学与技术系攻读计算机技术专业硕士学位至今。

发表的学术论文

[1] 汤启明，喻文健，许静宇，孙海洋. "基于库查找的差分对线网匹配检测算法". 计算机辅助设计与图形学学术会议, 2012 年(已录用)

[2] Weibing Gong, Wenjian Yu, Yongqiang Lu, Qiming Tang, Qiang Zhou, Yici Cai, "A parasitic extraction method of VLSI interconnects for pre-route timing analysis," in Proc. IEEE ICCCAS, Chengdu, China, July 2010, pp. 871-875.

参加的科研项目

[1] 国家科技重大专项“十一五”课题: 先进 EDA 工具平台开发（清华大学部分） (2008-2010).