庄文君　李玉兴　编著

集成电路布图设计自动化

上海交通大学出版社

集成电路布图设计自动化

庄文君　李玉兴　编著

上海交通大学出版社

内 容 简 介

本书是一本介绍大规模和超大规模集成电路(LSI/VLSI)布图设计自动化基本理论和方法的专著。

全书共分九章。前三章介绍了布图设计自动化的发展现状、主要的设计模式以及设计对象的描述方法。第四、五章讨论了各种自动布局设计方法。第六、七章介绍了面向线网及面向布线区域的各种布线设计方法。第八章讨论了数据库在布图设计中的应用。最后一章概述了布图设计自动化系统的构成及各子系统的作用。全书既介绍了历史上布图设计的主要理论和方法，又着重讨论了当前最新的一些研究成果，并附有详尽的参考文献目录。

本书可作为高等院校有关专业高年级学生或研究生的参考书、专业选修课或研究生课的教材，也可供从事计算机设计和研制、集成电路的设计和研制、软件算法的研究和应用的有关科技、工程技术人员参考。

集成电路布图设计自动化

上海交通大学出版社出版
(淮海中路1984弄19号)
新华书店上海发行所发行
常熟文化印刷厂排印

开本 850×1168 1/32 印张 13.625 字数 351000
1986年8月第1版 1986年10月第1次印刷
印数 1—3300册
统一书号：15324·194 科技书目：132-258

定 价：2.60元

前　言

在集成电路的发展进入超大规模时代的今天，计算机辅助设计和设计自动化已成为集成电路发展必不可少的支柱之一。布图设计自动化是其中重要的一个组成部分。近十年来，LSI/VLSI的布图设计自动化的研究和应用引起了人们极大的重视，与此同时，布图设计自动化的理论、方法与实际应用也得到了相当快的发展。为了适应我国 LSI/VLSI 布图设计自动化研究、应用以及教学的需要，我们结合国内研究工作的实践(包括作者在这方面的工作)，并综合了国内外在研究和应用中的主要理论和方法，编著了这本布图设计自动化的专著。希望在这方面对我国的工作和教学有所裨益。

本书内容共分九章，分述了布图设计自动化方面的主要模式和主要的设计理论与方法。本书一方面着眼于布图设计自动化技术的应用，对布图设计系统各部分的主要方法作了较为全面的介绍；另一方面为了更好地适应研究、教学的需要，对布图设计自动化中有代表性的各种算法，包括历史的发展过程以及国内外研究的一些最新成果也作了介绍。

本书初稿第一、二、三、四、五、六、七和九章及附录由庄文君同志撰写，第八章及第三章的一部分内容由李玉兴同志撰写。李玉兴同志并对前六章内容进行了文字修订和整理工作。全书最后的审定由庄文君、李玉兴共同完成。李玉兴同志为本书的出版作了大量的工作。

本书的主要内容，作者曾在清华大学、上海交通大学、安徽大学及电子工业部雷达局等单位，对研究生及高年级本科生作过讲授；对有关的科研、教学、工程技术人员也作过较系统的介绍。

本书由中国科学院半导体研究所王守觉研究员审阅，并得到上海交通大学林栋梁教授的关心和支持，在此谨表示衷心的感谢。由于限于作者的水平、书中错误和疏漏在所难免，作者恳切地希望使用本书的读者批评指正。

1984 年 12 月

目　录

第一章 引 论

1.1 布图设计自动化的发展现状及重要性

一、布图设计自动化的发展现状

近年来，集成电路技术取得了显著的进展。其主要标志是，单片集成度的迅速提高和单门(逻辑电路)平均价格的显著降低。从七十年代以来，集成度的提高大约是每两年翻一番。目前存贮器的集成度已是六十年代中期的一万倍。国外已研制成功256K RAM电路和32位单片微处理机。由于集成电路的发展已迈入了超大规模集成电路(VLSI)的时代，这势必促使以电子计算机为代表的电子设备迅速地向着超小型化、高集成度、高可靠性的方向发展。大规模集成电路(LSI)和超大规模集成电路已被广泛地应用于计算机、自动控制、通讯、雷达、测量、家用和医用电子设备等各个方面，它们已成为现代电子技术发展的一个极其重要的基础。

随着LSI/VLSI规模、品种、数量的不断发展，其设计自动化的问题也就愈来愈受到人们广泛的关注。人们已经从实践中得到了一个无用置疑的结论：LSI/VLSI的进一步发展(尤其是随机逻辑电路)离开设计自动化(DA)或计算机辅助设计(CAD)将寸步难行。近十几年来，许多国家的研究机构、高等院校以及有关的公司都不断投入巨额资金，开展对LSI/VLSI设计自动化、CAD的理论和方法、以及实用系统等进行研究和开发，并且已经在系统描述(system description)、系统分析及分划(system analysis and division)、逻辑模拟(logical simulation)、电路分析(circuit analysis)、工艺模拟(process simulation)、器件模拟(device simulation)、布图设计(layout)、自动测试(automatic test)以

及数据库（data base）等方面得到了相当大的发展。图1.1为LSI/VLSI计算机辅助设计系统的示意图。由于设计过程中课题的繁多和问题的复杂，许多方面目前离开真正的自动化设计还有不少距离。但即使如此，在这些方面也已广泛地利用计算机进行辅助设计，特别是在设计验证等方面，计算机已成为一个必不可少的工具。

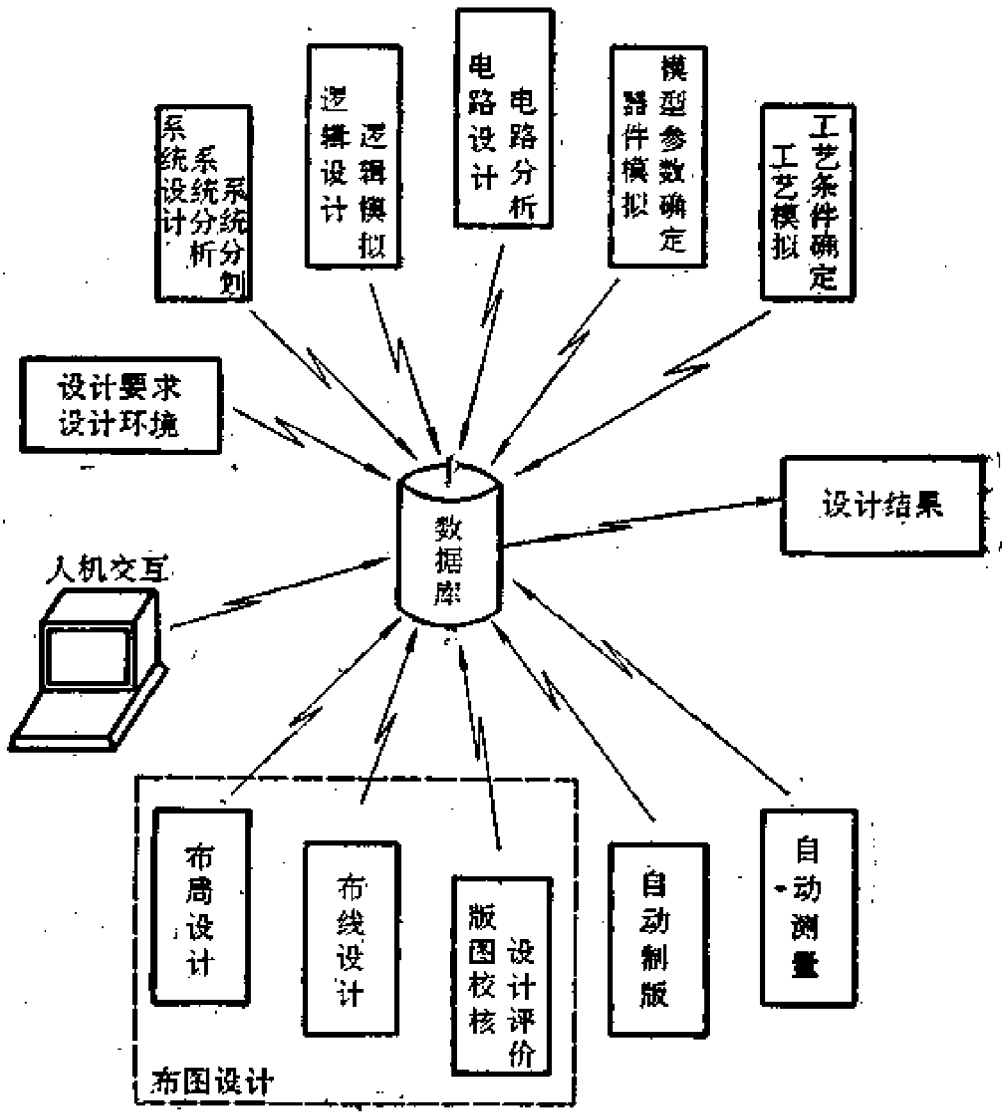


图1.1 LSI/VLSI设计系统示意图

二、布图设计自动化的重要性

布图设计是整个集成电路设计过程中与产品研制和生产直接相关的一个设计过程，它直接关系到LSI/VLSI的设计周期、成本、正确性和产品质量，而且也是人工设计中费时最长和差错率最高的设计步骤之一。正因为如此，它也就成了近年来在设计自动化方面发展最快、自动化程度最高的领域之一。

布图设计自动化在LSI/VLSI发展中的重要性，主要表现在下述两个方面：

1. 设计周期短、成本低、效率高

据报道，英特尔公司在设计INTEL 8080微处理机时，采用人工方式设计，总共化费40人年，设计费用为2千万美元。后来设计的二个微处理机也分别用去了10人年。HP公司设计一台32位微处理机(含45万个元件)，化费60人年。图1.2表明，在人工设计时，设计周期是与设计规模的2～3次方成正比的。因此，对于10万个门(gate)左右的随机逻辑来说(属于VLSI)，人工设计将几乎是不可能的。由此看来，必须采用设计自动化或计算机辅助设计，才能满足LSI/VLSI的研制及生产的需要。

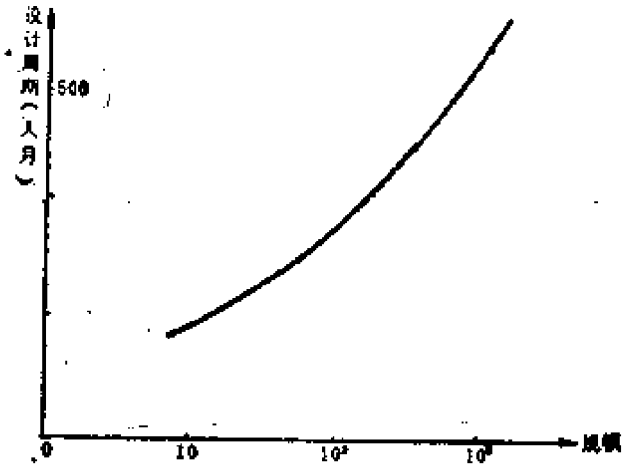


图1.2 人工设计周期与设计规模关系示意图

目前已开发的布图设计系统的设计效率，与人工设计相比可以提高一个数量级以上。这一点，可以从下面三个有代表性系统的设计能力上看出来：

(1) 美国 IBM 公司

使用 IBM 公司 EDS 系统，设计一台含 5000 门的微处理机，设计周期为 20 人月，设计过程在一台 IBM370/168 机上实现。其中布图设计共化费机时 376 分钟，如图 1.3 所示。如果设计 700 门左右的电路，设计周期为 7 人天。

(2) 美国 BELL 实验室

BELL 实验室设计一台 32 位 CMOS 微处理机(含 10 万晶体管)，设计周期为 15 个人月，如图 1.4 所示。若设计规模为 500 单元(cell)的电路，其设计周期仅为 5～7 人天。

(3) 日本 NTT MASAHINO 电子通讯实验室

日本 NTT MASAHINO 电子通讯实验室设计 32 位 CMOS 微处理机（含 7.8 万个晶体管），其设计周期为 2 个人月，如图 1.5 所示。

同时应该指出，这些设计系统的设计质量已基本上可与人工设计相匹敌。

2．易进行设计正确性验证

由于人工设计或人机交互式设计方式，主要是依据设计者的经验和专业素养，而无一定的方法可言，再加上极其繁杂的数据处理工作(仅就 4K RAM 而言，全部版图数据可达 36 万)，这就难免出现这样那样的差错。按一般规律，如果 1000 个晶体管的布图设计第一次的成功率为 2^{-1}，那么 35 万个晶体管的第一次设计成功率则仅为 2^{-230}，其一次成功率几乎为 0，由此可见差错率是相当高的。同时此类错误也相当难检查，在有些情况下甚至是不可能的。采用设计自动化技术进行布图设计，必然遵循一定的方法，数据可由计算机统一管理。与此同时，可发展一系列相应的正确性验证的手段，从而就可基本上解决设计正确性的验证问题，

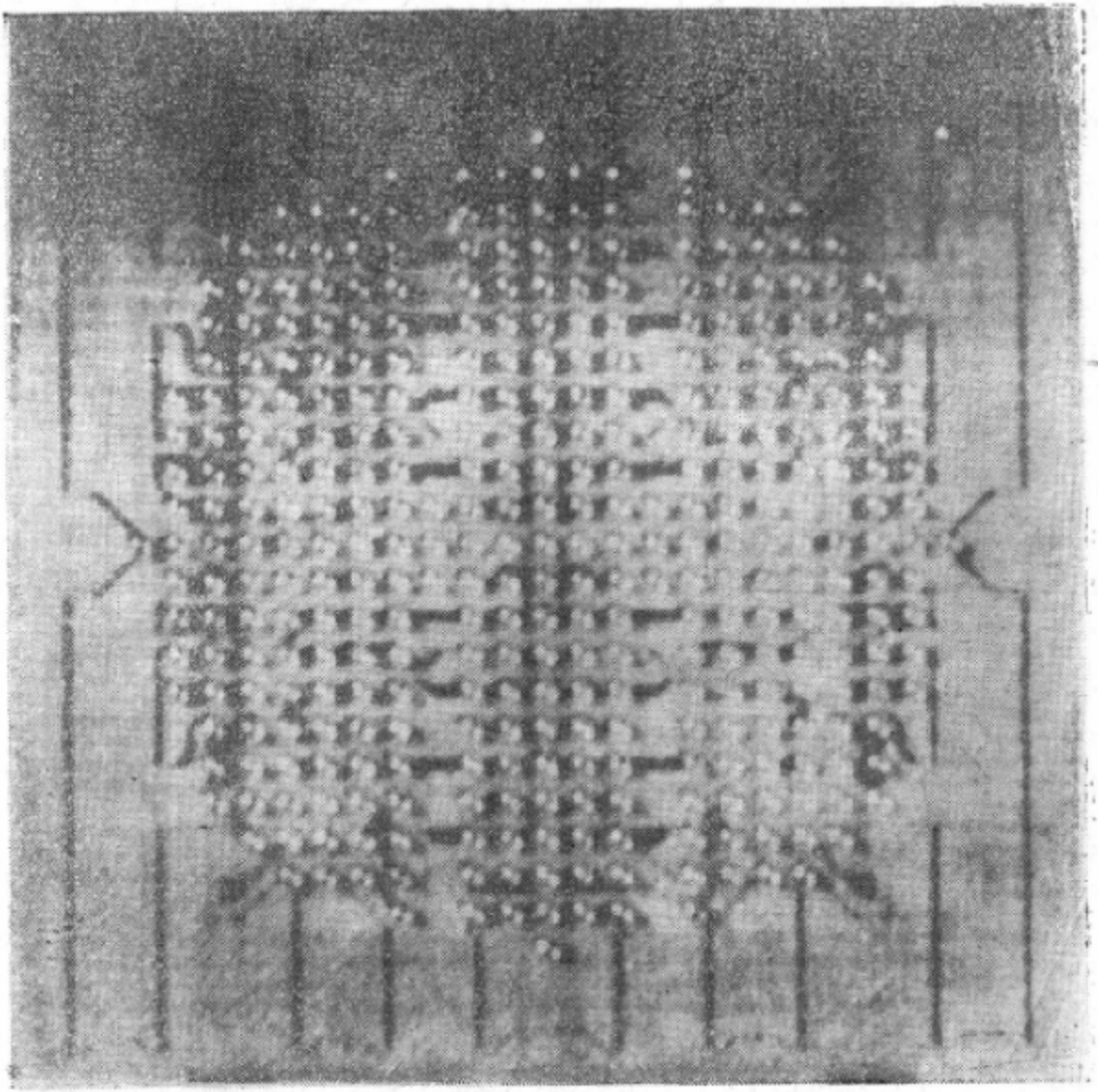


图 1.3　IBM 公司含 5000 门的微处理机设计

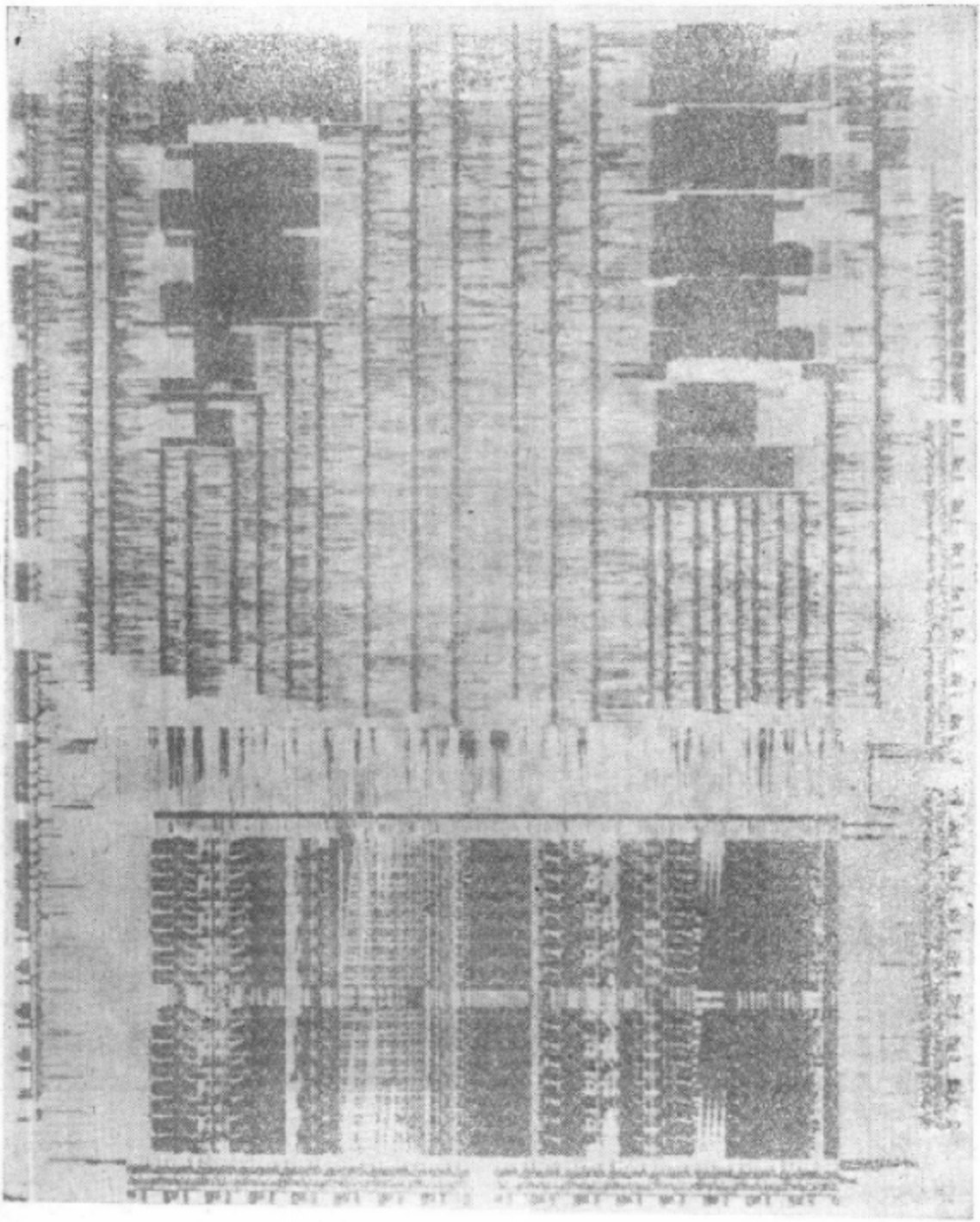


图1.4 BELL实验室32位CMOS微处理机设计

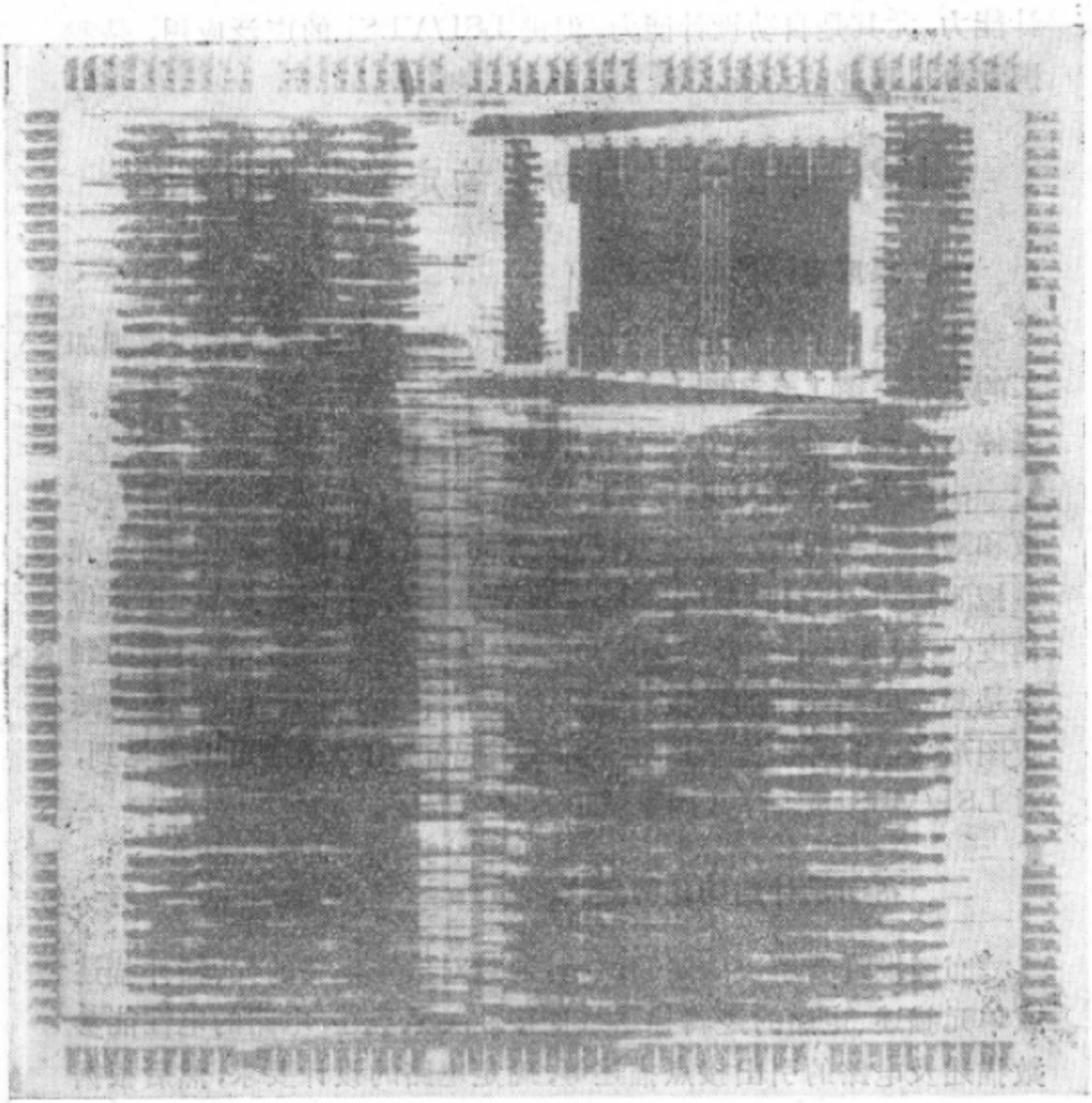


图 1.5　日本 NTT MASAHINO 电子通讯实验室 32 位 CMOS 微处理机设计

大大提高设计成功率。

由此可见，设计自动化（包括布图设计自动化）是发展 LSI/VLSI 必不可少的基础和手段。我国的 LSI/VLSI 研制和生产虽然起步较晚，但是完全有可能尽快赶上去。为此，发展独立的设计能力，尤其是自动设计能力，促进 LSI/VLSI 的广泛应用，是我国计算机事业的当务之急。

1.2 布图设计自动化的定义及其目标

一、集成电路的生产过程

集成电路的生产过程就是在芯片(chip)上进行一系列微细加工的过程。图 1.6 和图 1.7 分别显示了在芯片上制作一个双极型晶体管和一个金属氧化物半导体晶体管(MOS)的过程。

由图 1.6、1.7 可见，为了在芯片上精确确定的区域内进行扩散和最终布线的要求，必须进行多次的光刻，也即需要一整套的光刻掩膜版。这些掩膜版(mask)要求具有相当高的图形精度和定位精度(一般其精度量级为 10^{-6}m)。从某种意义上讲，掩膜版的制备是集成电路生产的关键。图 1.8 是一个 RS 触发器电路及其掩膜图形（已将多层掩膜图形套画在一起）。由此图形可以想见到，对 LSI/VLSI 来讲，其掩膜图形的复杂程度。

二、布图设计自动化的定义

布图设计要解决的问题就是：通过对给定电路的元、器件描述或单元描述、电路的逻辑描述或连接性关系描述，电路的电性能参数描述及电路的引出接点描述等，确定电路的设计要求，然后根据采用的集成电路工艺条件，将电路的描述自动地转变为集成电路制造所需要的掩膜图集。简言之，布图设计就是根据电路和工艺的要求自动完成芯片上元、器件或单元—功能块的安置，并实现它们之间所需要的互连。上面提到的生产工艺条件是指工艺类型、

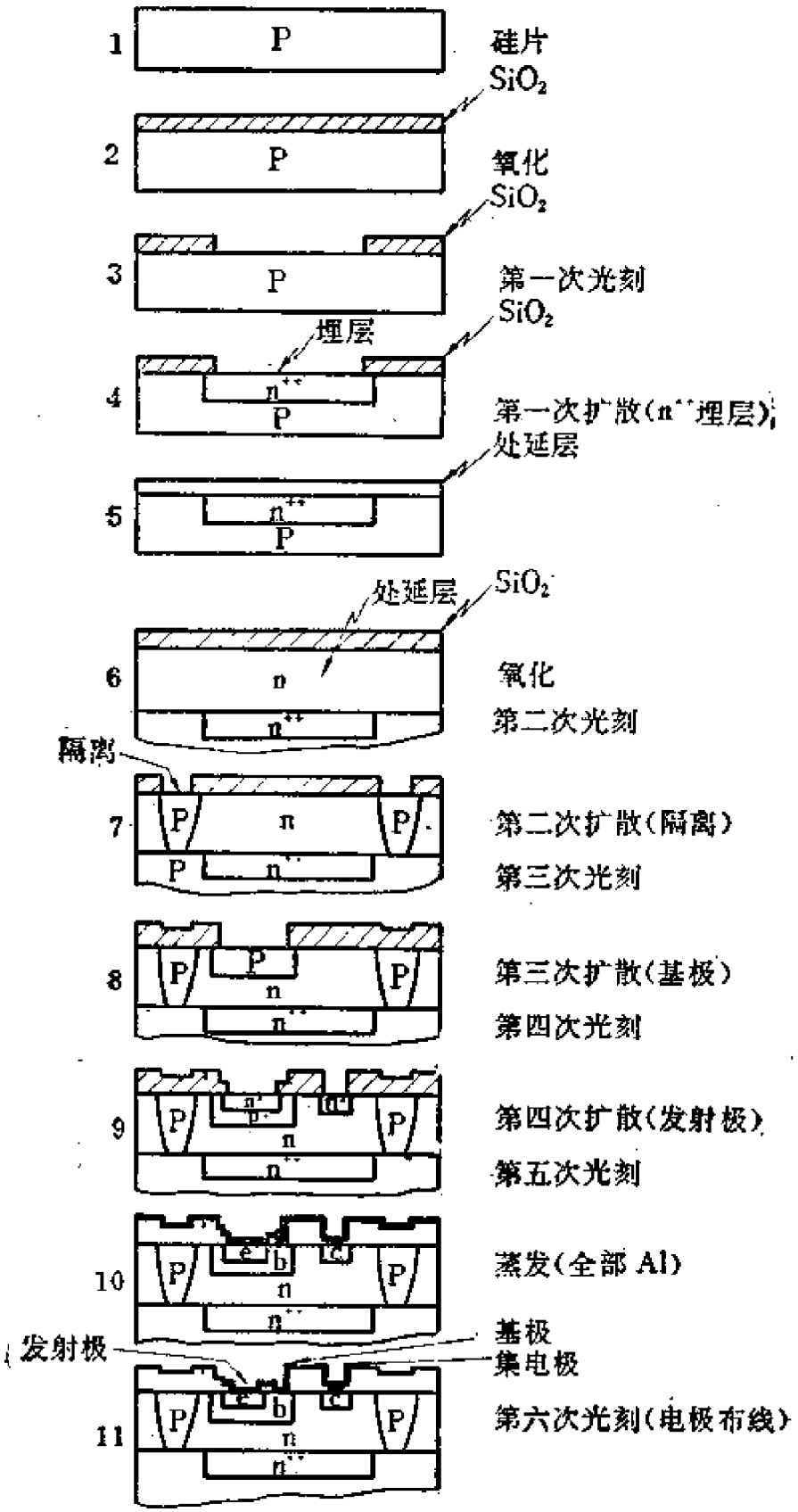


图 1.6　集成电路双极型晶体管制造工艺(PN 结隔离)

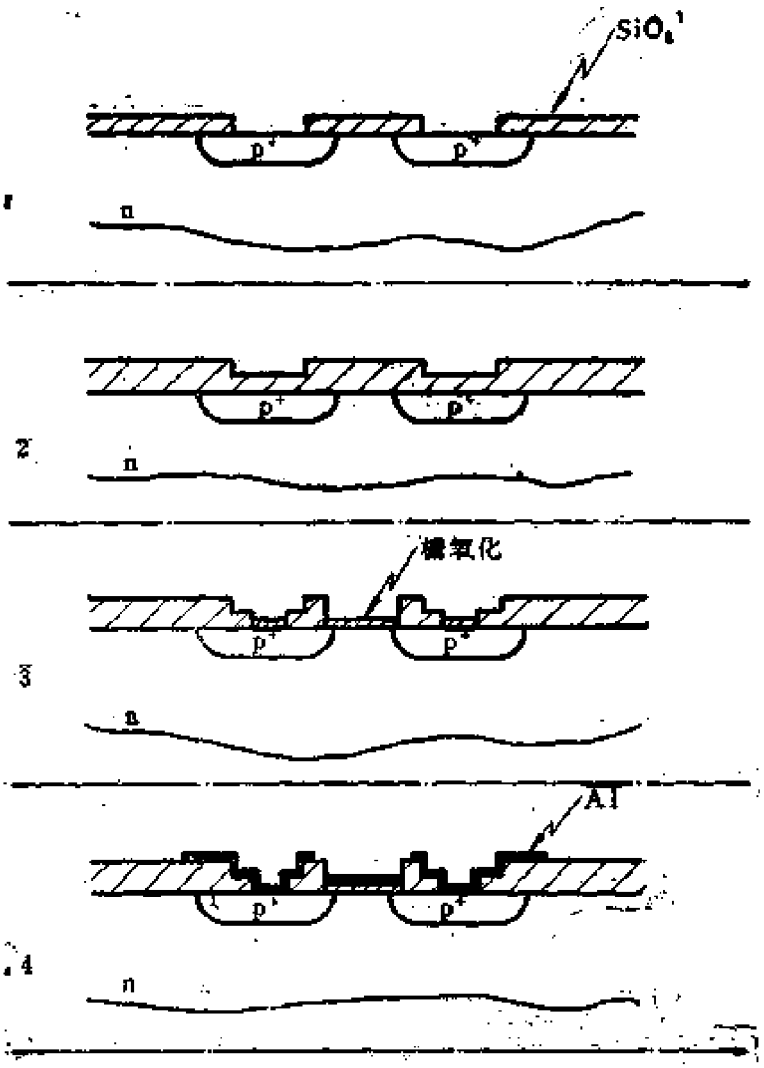


图 1.7　集成电路金属氧化物半导体晶体管制造工艺

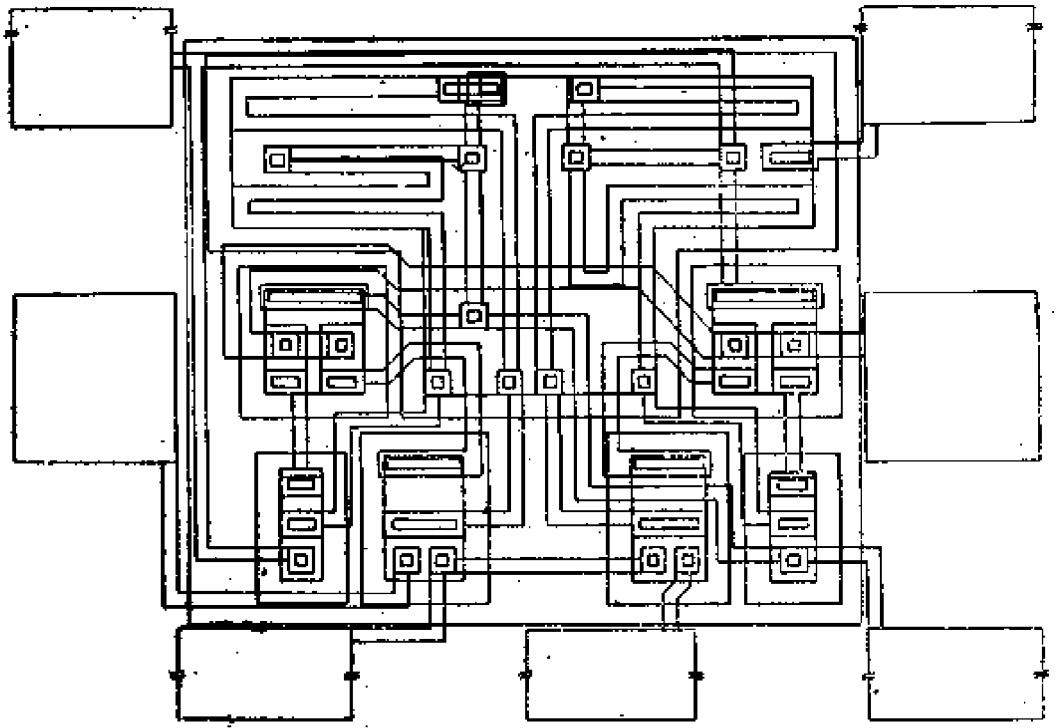


图 1.8　RS 触发器电路的掩膜图形

布线层数及层特性、连线宽度及间隔、通孔大小、接点尺寸、套刻精度等。

三、布图设计自动化的目标

在 LSI/VLSI 布图设计中，一般并不直接以晶体管等电路元件为基础来进行设计，而是以单元电路(逻辑门、寄存器等)为基础。而单元电路的布图设计，则一般由人工完成或由标准单元库提供。

1．布图设计由人工完成或标准单元库提供的原因

① 人工在小规模电路的设计中，尤其是在以晶体管等电路元件为基础的布图设计中，远比计算机灵活、合理，而且设计质量高，并且设计速度并不太慢。然而，高质量的单元电路布图设计却非常有利于提高整个 LSI/VLSI 布图设计的质量。

②单元电路往往具有较好的通用性，可作为标准单元来使用，并将有利于降低设计成本。

③ 将有利于缩短 LSI/VLSI 的布图设计周期。

④ 将有利于设计正确性验证及提高设计成功率。

如果我们把单元电路的布图设计结果看作是带连线接点的模块，那么 LSI/VLSI 的布图设计目标就可以描述为这样一个问题：对给定的大规模或超大规模集成电路，在满足各项电路及工艺要求的条件下，如何在尽可能小的区域内（或在给定区域内），互不重迭地安置电路所需要的全部模块，并且完成它们所有必须的互连。

2. 布图综合考虑的设计目标

在布图设计中，除了完成从电路图（或逻辑图）到掩膜图集的转换外，还将综合地考虑下述设计目标：

① 满足电路的各项电性能参数的要求及工艺要求。

② 使设计结果具有足够高的集成度。集成度指标通常以芯片上单位面积内所含晶体管或等价逻辑门的数量来表示。

③ 使设计过程具有足够高的设计成功率，其中主要的是指布线成功率。

④ 使设计结果具有较高的设计质量，如尽可能减少通孔数量、缩短连线长度、使寄生参数的影响限制在电路所允许的范围之内等。

此外，还必须考虑设计周期，设计成本以及结果验证等要求。

在具体电路的设计过程中，根据特定的工艺条件，往往还会对布图设计提出一些专门的要求。如某些线网的树型要求、长度限制等等。这些要求，在设计过程中也必须加以考虑。

1.3 布图设计自动化的复杂性及研究方法

一、布图设计自动化的研究方法

对于一个复杂的大规模或超大规模集成电路来说，往往很难设想其最佳的布图设计结果是什么样的。换句话讲，设计质量的

精确评价往往是很难做到的。而如何去获得一个满足全部设计目标的令人满意的布图结果，则更是一个相当困难的问题。因此，在布图设计中，一般是把整个设计过程处理成若干相对独立的子过程的序列，使得整个设计要求将分步地得到满足，而各子过程面临的问题将得到相对的简化。图 1.9 是布图设计过程的示意图。需要着重指出的是，布图问题就其实质来讲是一个统一的问题。子过程相互之间实际上是互相影响、互相制约的。因此在对某一个

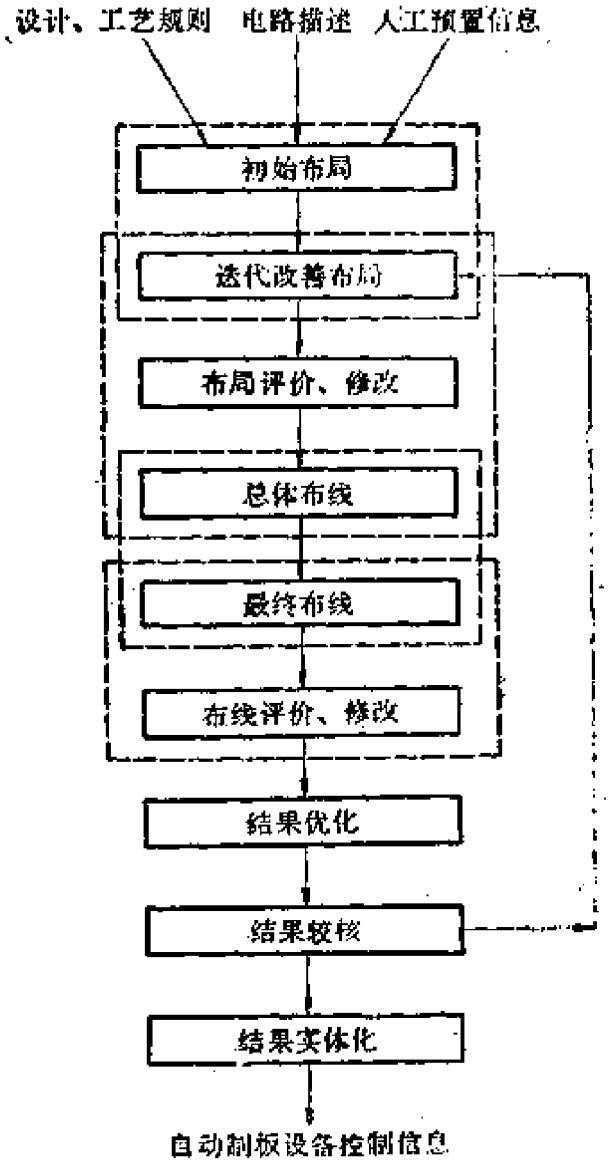


图 1.9　布图设计过程示意图

过程的处理中，注意它与其它子过程的关系以及对全过程的影响将是十分重要的。实际上，在不同的布图系统中，子过程的划分及综合也是各不相同的。与此同时，在布图设计中还往往采用分级设计 (hierachical layout) 的方法[26]*。图 1.10 给出了分级设计示意图。这样一来，就把一个复杂的 LSI/VLSI 布图设计问题，按一定的设计规则(连接关系、逻辑类型、设计方式类型等)分解成若干个子问题。如果子问题仍然十分复杂，那么还可以继续分级，使每次设计所考虑的对象都比较易于处理，并且将有利于设计质量的提高。

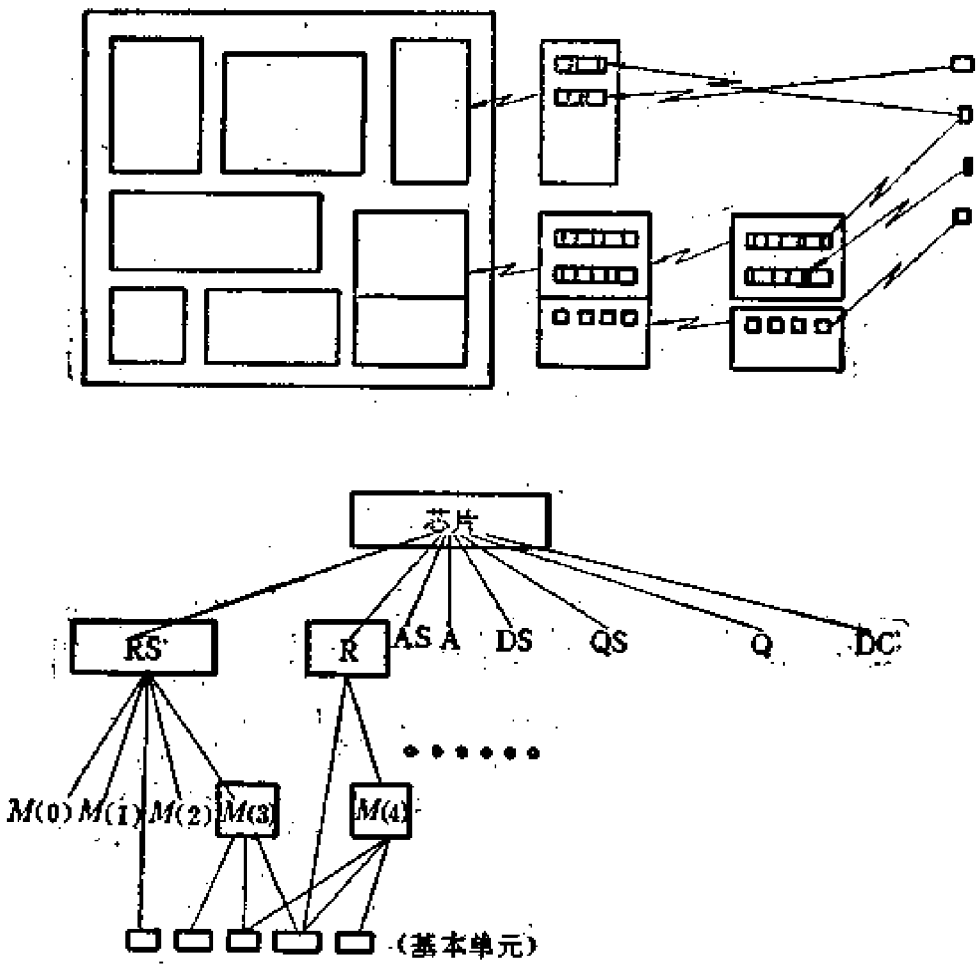


图 1.10　分级设计示意图

* [　]中的数表示参考文献的编号，以便读者在书末查阅。

二、布图设计自动化的复杂性

布图设计自动化面临的问题是十分复杂的，即使采用分级设计的办法，布图问题所分解成的子问题或子过程还是很复杂的。这从以下两个方面可以看出：

1．布局设计的复杂性

对于布局(placement)设计子过程，可从二个方面描述其复杂性：

(1) 模块安置面积最小化的要求

如果仅考虑模块(block)的形状、大小，而不考虑模块间的互连关系，并且设计目标也仅考虑安置面积的最小化，则布局问题就可以蜕变为一个类似"下料"问题的难题。然而，实际的布局问题则必须考虑模块之间的连接关系、相应的连线区面积以及布线成功率等问题。这样一来，其困难程度将远远超过"下料"问题。

(2) 模块间连线总长度最短化的要求

如果仅仅考虑模块间的连接性关系，而不考虑模块的形状和大小，此时就可以将模块拓扑为一个点。若设每条连线仅连接二个模块，并且再以"安置后使连线总长度最短化"为设计目标(这目标与可布性有一定对应关系)，则布局问题即蜕变成一个典型的二次分配问题，其解的可能方式为 $n!$ 种。若要求其最优解，则将是一个相当复杂的问题。何况在实际布局问题中，模块的形状、大小以及多接点连线的问题是必须考虑的，而且设计目标事实上要比上述目标复杂得多。由此可见，布局问题的复杂程度是相当高的。

2．布线设计的复杂性

在布线(routing)设计子过程中，我们只要举一些局部问题来分析，就足以反映出整个问题的复杂性了。

(1) 一条线网的布线问题

假定我们仅考虑其中的一条线网(net)的布线问题，并以其

连线长度最短为设计目标。当其所连接点的位置已被确定，并且采用最小链接树方式实现互连时(见图 1.11)，那么这个问题就是人们所熟知的“巡回售货员”问题，即图论中“求给定图中权数最小的哈密顿圈”的 NP——完备问题。若采用 steiner 树的方式实现互连时(见图 1.12)，则这个问题也是一个典型的 NP——完备问题，即图论中“给定一个连通的加权有向图 $G_d(V, E, F)$和 V 的一个子集 V′，在 G 中求一个连通 V′ 中全部顶点的最小树”的问题。

图 1.11 采用最小链接树方式实现互连

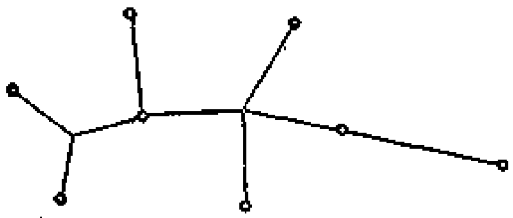


图 1.12 采用 steiner 树方式实现互连

在实际问题中，所要考虑的是成千上万条连线的问题。既要考虑这些连线的相互影响，又要考虑实际布线空间的限制等因素。其复杂性是可以想见的。

(2) 通道区布线问题

当然，我们还可以从另一个角度来说明布线问题的复杂性。如果我们仅考虑一个布线子域的布线问题——通道区布线(channel routing)，并且以完成布线所需要的布线区面积最小化为设计目标，则已经证明，一维的(双边)通道区布线是一个 NP——完备问题。而对于二维的通道区布线问题(switchingbox)，则至今还没有一个令人满意的解决方法。对于一个实际的布线问题，往往存在着几十个甚至上百个布线通道区。另外，还必须考虑如何进行线网的合理分配以及各布线通道区之间的影响等问题。

综上所述，可见布图设计自动化所面临的将是一系列极其困难而又引人入胜的问题，正因为如此就决定了它的研究和开发工作的一些特点。

3．布图设计自动化的特点

(1) 建立标准化的布图设计模型

在布图设计自动化中，一般采用标准化、规范化的布图设计模型，以建立一个有明确定义，便于研究和发展的设计方法。

(2) 采用分级设计方法

在布图设计自动化中，一般采取分过程的设计方法和分级设计方法，以降低各设计过程和各级设计中的问题复杂性。

(3) 使用启发性算法

由于几乎在每个设计过程中都面临着计算复杂性是指数复杂性的问题，因此在布图设计自动化中广泛采用模拟人的设计经验和思维方式的所谓启发式(heuristic method)算法，并且特别注意算法有效性的评价。

(4) 布图设计是一个迭代过程

在布图设计自动化中，整体的设计目标决定着各设计过程的设计要求；而整体设计的结果又非常强地依赖着各设计过程的设计结果。在分级设计中，上一级的设计和下一级的设计之间也存在着类似的关系。由于存在着这一特点，那么不难预见，LSI/VLSI布图设计的过程将实质上是一个迭代过程。这也就是说，在整个布图设计过程中将要求进行频繁的调整和优化。显然，这种设计过程对数据的结构和管理要求比较高，最好面向数据库。

(5) 采用积累式的方式建立布图设计系统

由于问题的复杂性和实际工程应用的需要，布图设计自动化系统的研制、开发，一般采取渐进的、积累式的发展方式。

三、两类求解算法

在布图设计自动化中，对计算复杂性为指数复杂性的问题，以及对部分计算复杂性为多项式($O(n^K)$) (K 足够大)的问题。实际应用的求解算法大致可分为两大类——二端算法模式和三端算法模式(见图 1.13, 1.14)。

图 1.13　二端算法模式

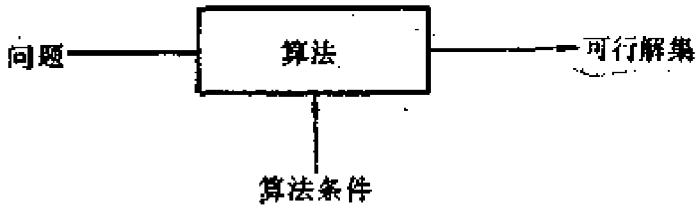


图 1.14　三端算法模式

1．二端算法模式

二端算法模式一般是一些“最佳化”算法。算法的目标是：对尽可能多的情况求得最佳解或准最佳解。满足上述目标的二端模式算法，一般有较好的求解精度稳定性，但求解效率稳定性较差，甚至用同一算法处理规模相当的不同的问题，有时其处理效率可相差几个数量级。

2．三端算法模式

三端算法模式一般是一些“启发式”算法。算法的目标是：以相当高的效率求得问题的近似解，并通过控制不同的算法条件（如各种初始条件的选择，算法参量的变化等），产生一个可行解的集合。然后从中选取最佳的一个解作为算法的结果解。一般来说，三端模式算法的每次求解都具有相当好的求解效率稳定性，而且效率相当高。但是有限次的算法条件选择，并不能保证可行解的集合覆盖实际需要的令人满意的结果解。

显然，发展一个对尽可能多的问题，既具有相当好的求解精度稳定性，又具有较高的求解效率及稳定性的二端模式算法；或发展一个对尽可能多的问题，经过较少次算法条件选择，即可覆盖令人满意的结果解的三端模式算法，就是我们在布图设计自动化中，解决指数复杂性和当 K 足够大时的多项式复杂性问题的研究方向。

第二章　布图设计的主要模式及其特点

2.1　传统设计方式

一、传统设计方式及其特点

传统的人工设计方式或人机交互式设计方式，主要根据设计者的经验和设计素养，从逻辑图或电路图的描述开始，考虑具体的电路要求和工艺要求，进行电路的元、器件安置和互连设计；然后将拓扑设计的结果精确地以1:500或1:1000的比例绘制成分层的掩膜版图；再进行繁复的数字化工作之后，由自动制版设备制成满足工艺要求的整套掩膜版（见图2.1）。整个设计过程是一个大量反复调整的过程。对于LSI/VLSI，尤其是随机逻辑电路，需要处理的数据约有几十万到几百万个。仅其数字化的工作量就是极其可观的。

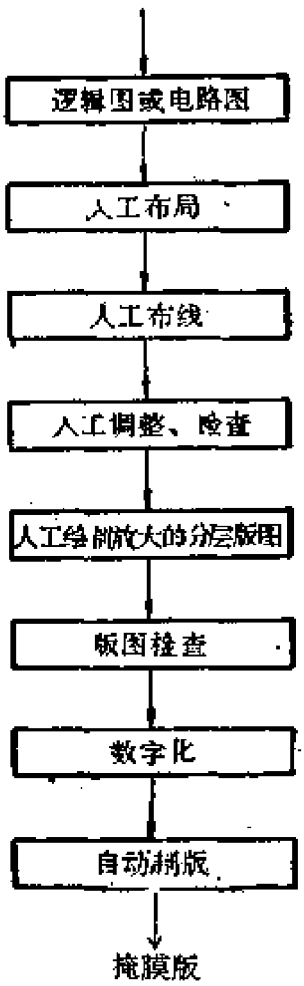


图2.1　传统设计模式

二、传统设计方式的优缺点

传统的设计方式具有周期长、差错率高，而且设计正确性验证困难等缺点。因此使用这种设计方式，只有样品生产出来之后，才能检测布图设计的正确性。另外这种设计方式的调试费用是随着电路的规

模和复杂性成指数关系上升的。不过，由于主要设计过程是人工进行的，因此，这种设计方式解决问题的灵活性比较大。如果设计成功，芯片的布图密度可以相当高。这种设计模式适用于单元电路、中、小规模大批量生产的电路，规则电路（如 RAM，ROM 等）以及特殊要求的高质量电路的布图设计等。

2.2 多元胞设计模式

一、多元胞设计模式及其特点

多元胞(polycell) 设计模式是以预先设计好的功能单元，也称标准单元(standard cell)为基础的。这些单元在电学上可以是不同类型的各种门电路，也可以是复杂的触发器、全加器等功能电路。但是，不管是那种类型的电路，要求这些单元的版图具有同样的高度，而宽度可以不同；除电源、地线接点之外，其它连接点要排

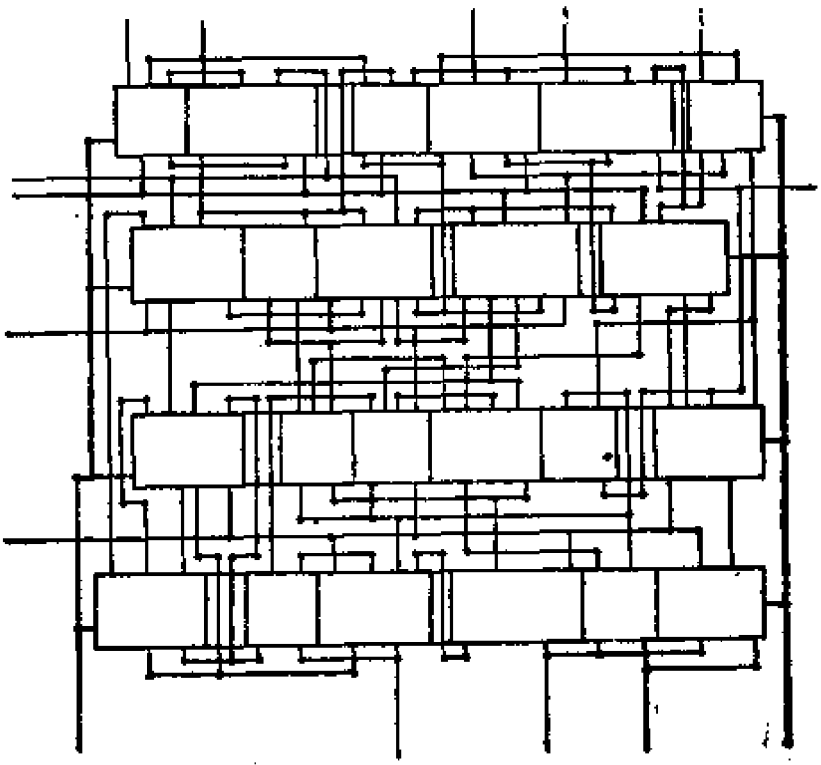


图 2.2 多元胞模式

列在单元的一边或相对的二边上。它们的电学和逻辑特性以及几何尺寸、接点位置等信息应该存放在一个标准单元库中，以便于检索和引用。设计时，根据电路的互连要求以及布图面积最小化的设计目标，将单元成行地排列，以此完成单元的布局(location)设计。电路的互连线(除单元内部的连线外)设计在单元行之间的水平通道区和单元行二端的垂直通道区之中。水平通道区的高度和垂直通道区的宽度可以依据布线的要求进行方便的调整。现有的好的布线算法可以保证100%地完成全部布线要求(见图2.2)。

二、多元胞设计模式的优缺点

这种设计模式是目前仅有的真正自动化布图的设计模式。它的设计方法相对地讲比较成熟，目前已有不少多元胞设计系统投入实际应用。如日立的LILAC系统[4]，贝尔实验室的LTX系统[5]，SIEMENS—AVESTA系统[6]，三菱电气公司的MILD系统[8]等。这种设计方式具有设计周期短、成本低、成功率高以及系统可发展性好等优点。如AVESTA系统，设计112个单元的电路布图，所需要的CPU时间仅为400秒(SIEMENS 4004/151计算机)；LTX系统，设计500单元规模的LSI芯片，其设计周期为一星期左右。

另外，设计实践还证明，采用分级设计的多元胞模式可成功地进行VLSI的布图设计。据1982年IEEE固态电路年会报道，日本NTT MUSAHINO电子通讯实验室，利用分级设计的多元胞模式成功地设计了一台32位CMOS微处理器[28]，其中含39种单元，单元总数为6222个，含78K晶体管(17K门电路和2304 bit RAM)，17125条连线和200个外接点。布图后芯片面积为$12\times12mm^2$，布图密度为542晶体管/mm^2，总的设计周期为2个人月(见图1.5)。

这种设计模式的问题是，当单元行的长度增加时，芯片的布图密度将随之下降。此外，由于单元标准化的要求，将使得某些子电

路的设计以及某些单元的外形调整感到困难。

三、多元胞模式的类型

我们称接点由一边引出(除电源、地线接点外)的单元为单边单元，接点由对边分别引出的单元为双边单元。单边单元一般是采取"背靠背"的形式，由二排单元构成单元行的排列方式（见图2.3)。

在实际应用的系统中，多元胞模式存在着多种类型，如图2.4所示。这些类型的主要差别在于：

① 单元的接点位置和外形要求的不同。

② 单元在芯片上排列方式的不同。

图2.4(a)为单边单元的多元胞模式，图2.4(b)为单边单元"行列式"结构的多元胞模式，图2.4(c)和图2.4(d)分别为双边单元相应的二种多元胞模式类型。一般地讲，双边单元类型的多元胞模式，将使得单元内部的布图更趋于合理，并且使整个芯片的布图设计具有更多一些灵活性。图2.4(e)为含宏单元（MACRO CELL)的多元胞模式。在这种类型中，标准单元的高度可在一定范围内变动。

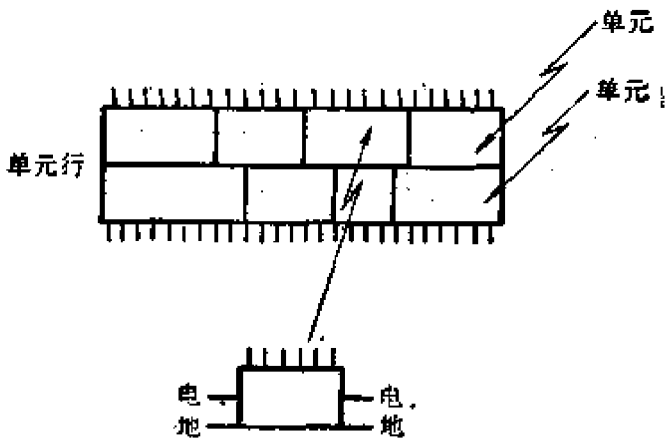


图2.3 单边单元及其在芯片上的排列方式

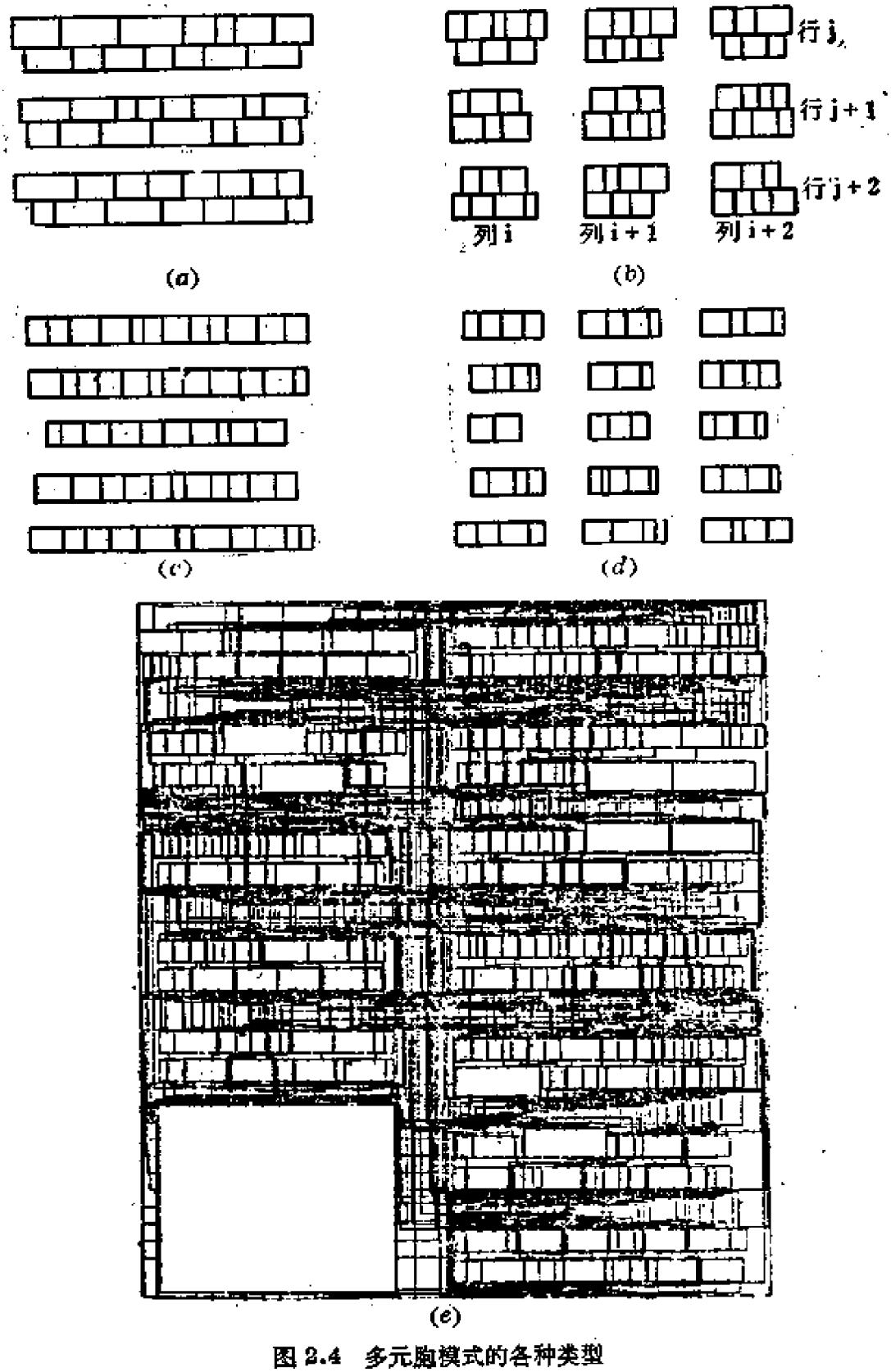


图 2.4　多元胞模式的各种类型

2.3 门阵列模式

一、门阵列设计模式及其特点

门阵列(gate array，master slice)设计模式利用预先制造好的所谓"母片"来进行布图设计。母片上通常以一定的间距成行成列地排列着大小和形状相同的基本单元电路(一般为门电路)。由于母片完全是规范化的，因此在针对具体电路的布图设计前，母片上各单元电路所含的电路元件(晶体管、电阻等)可事先制备好。在布图设计时，根据具体电路的要求，首先进行单元的分配，使得电路上的每一个单元都与母片上某一单元建立起严格的对应关系(通常母片上的单元数将大于电路上实际需要的单元数，因此母片上有些单元将是"冗余"的)，然后将单元间的布线区域划分为通道区，并以适当的原则将互连线分配到各个通道区。最后应用通道区布线算法实现单元的互连(见图 2.5)。

二、门阵列设计模式的优缺点

这种设计模式自动化程度较高，设计周期短，设计成本也低。尤其是由于采用了预制的母片，这就可能在用户提交了电路要求之后，能相当快地得到所需要的 LSI/VLSI 产品。这种设计模式适合大批量的工业化生产，因此它很受人们重视。

从方法学的观点看，由于这种设计模式采用了最标准化和规范化的布图模型，因而对设计方法（算法）的研究和发展相当有利。目前已开发了不少实际的工程系统，如IBM 公司的 EDS 系统[13]，三菱电气公司的 MARS—M_3 布图系统[10]等。MARS—M_3 系统设计一个含 683 个门、716 条连线的电路，在 MELCOM—COSMO 700 机上的设计机时为 97.2 分(见图 2.6)。IBM 公司设计 700 门左右的电路，设计周期为 7 人天(一人可同时设计二个项目)，其中 50% 的电路可以全部地自动完成设计，平均每个电路仅有

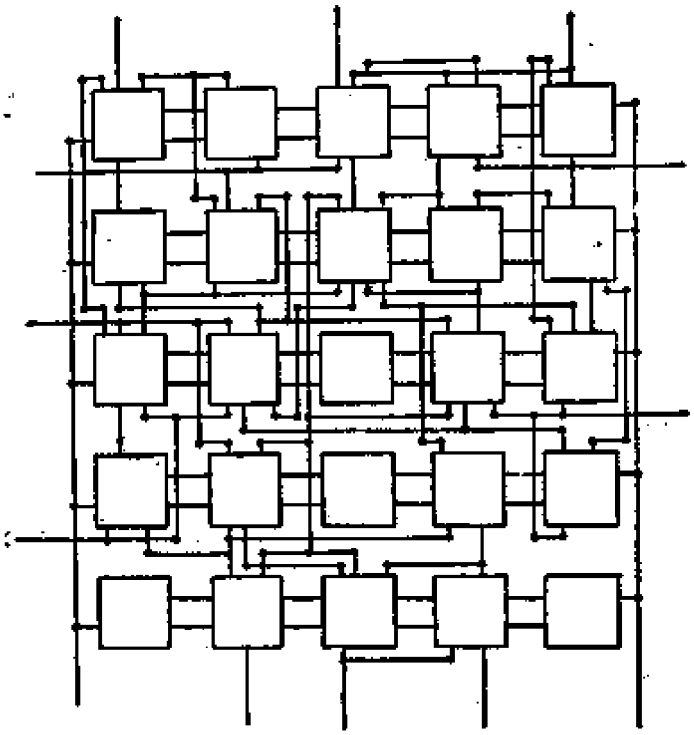


图 2.5　门阵列布图方式

1.5～1.6 个问题(剩线等)要人工去解决。IBM 公司并用 EDS 系统设计成功一个含 4923 门、10605 条连线的微处理机，母片上实际单元数为 7640 个，门利用率为 65%，芯片面积为 $7\times7mm^2$，剩线为 68 条，布通率达 99.4%，布图设计周期为二个月，设计所费机时为 376 分(IBM 370/168 机)(见图 1.3)。

这种设计模式的问题是：

1．布图密度低

有限品种的母片与不同电路的要求之间有矛盾。母片是统一生产的，其品种(不同规模和密度)有限。若用它去满足规模和复杂性各不相同的电路要求，由于需用母片上单一的基本单元(如四输入端与非门)去适应电路中各种单元的需要，则势必造成芯片面积利用率下降。一般来讲，其布图密度也较低。

2．不能保证自动地完成全部布线

由于这种设计模式对所需的全部互连不能保证自动地完成，

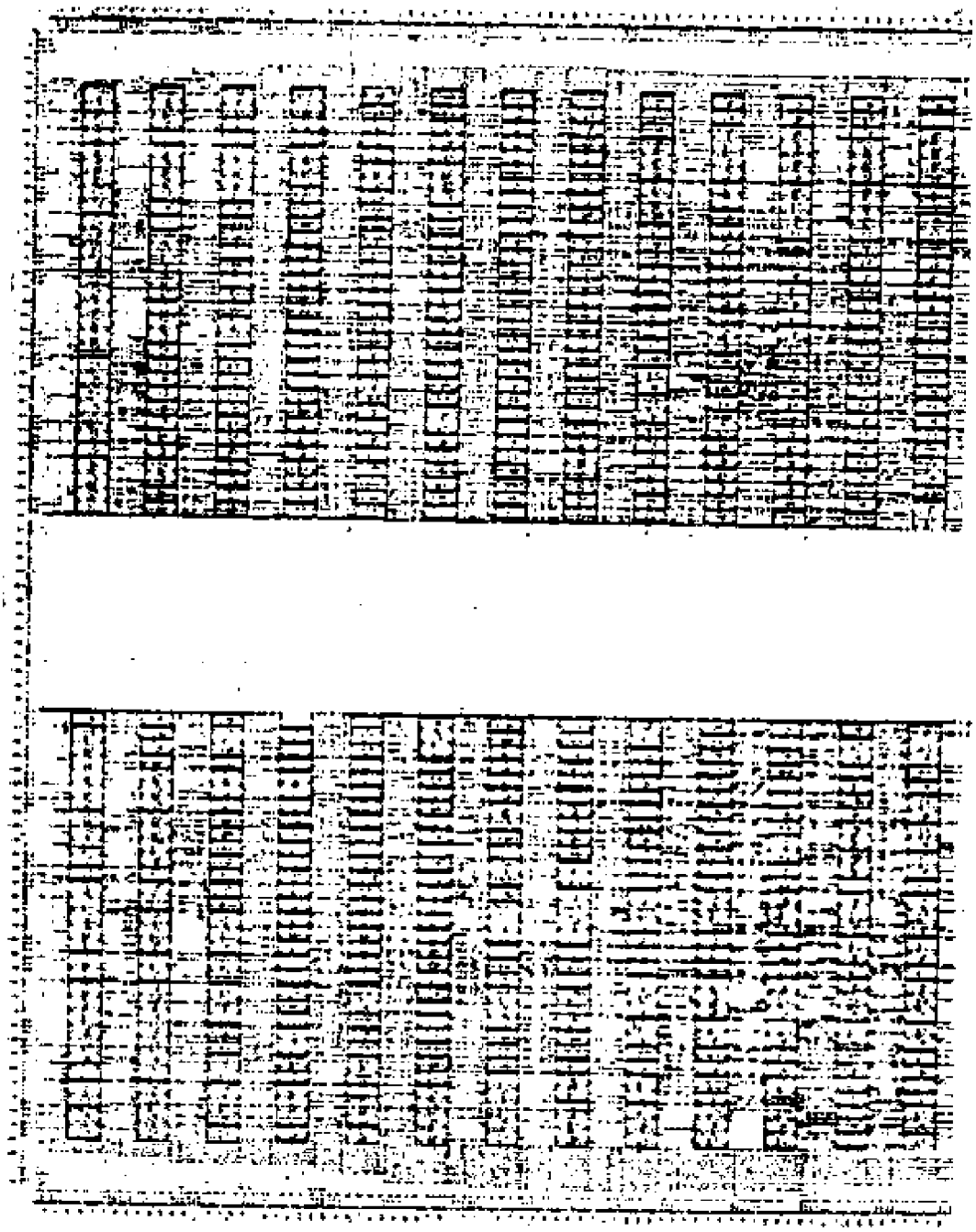


图 2.6　一个电路布图实例

因此往往需要人工进行剩线处理。处理剩线问题是很麻烦的，即使借助于较好的人机交互系统相辅助，处理起来也是相当困难的。比如说，为要把剩线布上去，可能要拆除某些已布好的线，然后把这两部分线重新布上去。拆去哪些线，又如何进行重布，有关这样的问题，都是至今难以解决的。人工处理剩线的速度是很慢的，一般来讲平均为2条/天。

三、门阵列模式的各种类型

在实际应用的系统中，门阵列模式有一些不同的类型：

1. 一般的门阵列

这种类型的设计目标是：最大程度地利用所有的门电路，并尽可能100%地完成布线。该类型适用于多品种的，并且总量相当大的工业化LSI的生产情况。

2. 砖墙式母片方式(brickwall masterslices)

这种类型是根据布线的需要，在门阵列中原单元位置上可以不放置门电路，而是利用该空间调整布线通道区。这种方式可使设计有一定的伸缩性，即多化费一点时间，可望获得较高的布图密度。一般来说，这种方式的布图密度将高于一般门阵列方式。

3. 带插入模块的门阵列方式

带插入模块的门阵列方式(masterslice with embedded macro)可在母片的门阵列中插入少数宏单元。这种含宏单元的母片主要是为某种具体的应用而专门设计的。它具有一般门阵列的优点，而且又能可望获得更满意的布图密度。

四、母片上基本单元的形式

为了提高芯片的利用率和更好地满足布线的需要，母片上的基本单元可以采用不同的形式。如三菱电气MARS—M3系统就采用了一种"对称"的双边单元，单元二边相对的接点都是电学上的等价接点（如图2.7）。布线时，每个单元不用的输入端可用作

"过渡通道"(feed through track)，冗余的单元则可用作"过渡通道单元"(feed through cell)。这种基本单元形式将使布线具有更大的灵活性和合理性。

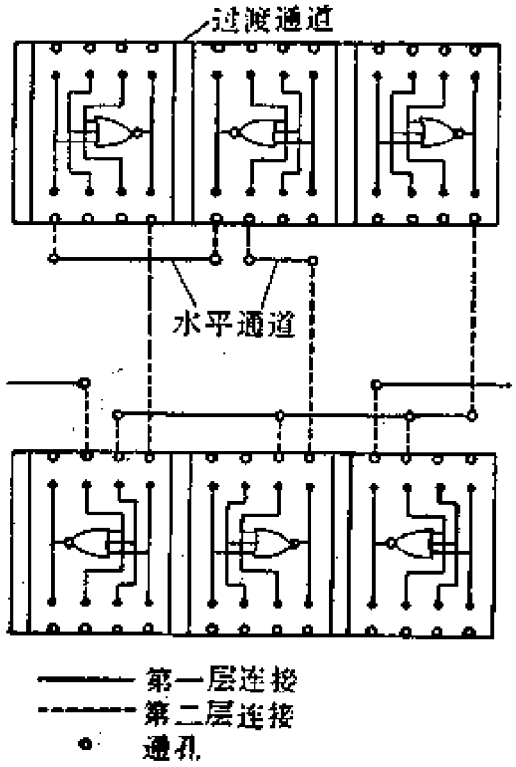


图 2.7　一种对称式的双边单元

通常采用的单一门单元的母片具有下述缺点：当门单元设计成较多的输入端时（如四输入端NOR），则母片上将有相当多的输入晶体管是无用的；而若是设计成具有较少输入端的门单元时(如二输入端 NOR)，则构成一个逻辑系统必然需要更多的门单元。根据这一情况，MITSUBISHI 电子公司采用了一种"对门"形式(paired—gate)的母片基本单元(如图 2.8)[14]，可使浪费

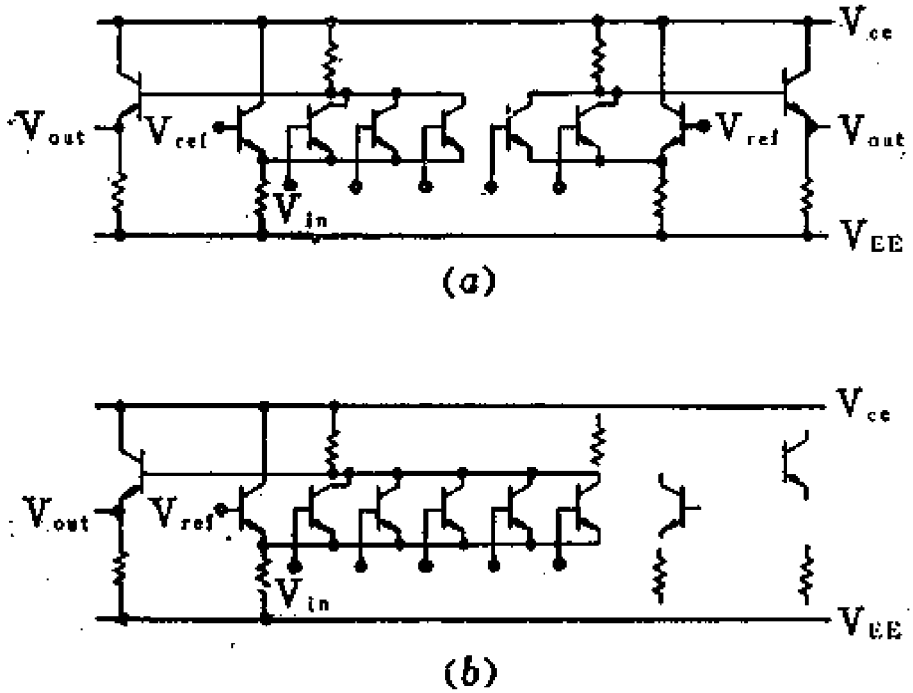


图 2.8　由两个内部门组成的"对门"形式

的输入晶体管数和所需要的门电路的数量都得到减少。

2.4 任意元模式

一、任意元设计模式及其特点

任意元（custom）设计模式也是利用预先设计好的功能单元的版图（各种门单元，寄存器，RAM，ROM 等）来进行电路的布图设计。这些单元版图的形状和大小是任意的。如果限定单元的形状为矩形，则可以称之谓“准任意元”(semicustom)模式[16]。任意元模式的单元引出接点可以排列在单元的四边上。这种设计模式是目前最接近于人工设计的布图模式。在布图设计时，根据电路的互连要求以及单元的外形参数进行单元的安置。此时，单元的形状和大小必将对单元的安置起着相当大的影响。当单元被安置之后，就把单元之间的布线区域划分为通道区。通过合理的相互连线的分配并且进行通道区布线，以实现电路的互连（如图2.9)。

二、任意元设计模式的优缺点

这种模式的单元设计具有很大的灵活性，可以是单元库中的基本单元，也可以是用其它设计方式得到的宏单元(多元胞方法以及后文将介绍的 PLA，gate matrix 等)，当然还可以是人工设计或符号法设计得到的功能电路单元。这种模式可望获得相当高的布图密度。随着 VLSI 的发展和分级设计的需要，这种设计模式正在引起人们愈来愈大的兴趣。

正因为该设计模式的单元设计所具有的灵活性，这必然给布图设计的算法带来一定的复杂性，使得迭代、调整和优化都比较困难。因此，一般情况下都需要利用人机交互式方式进行调整，尤其是在单元的安置过程中就更是如此。这种模式的布图设计方法尚处于不断的研究和开发中。

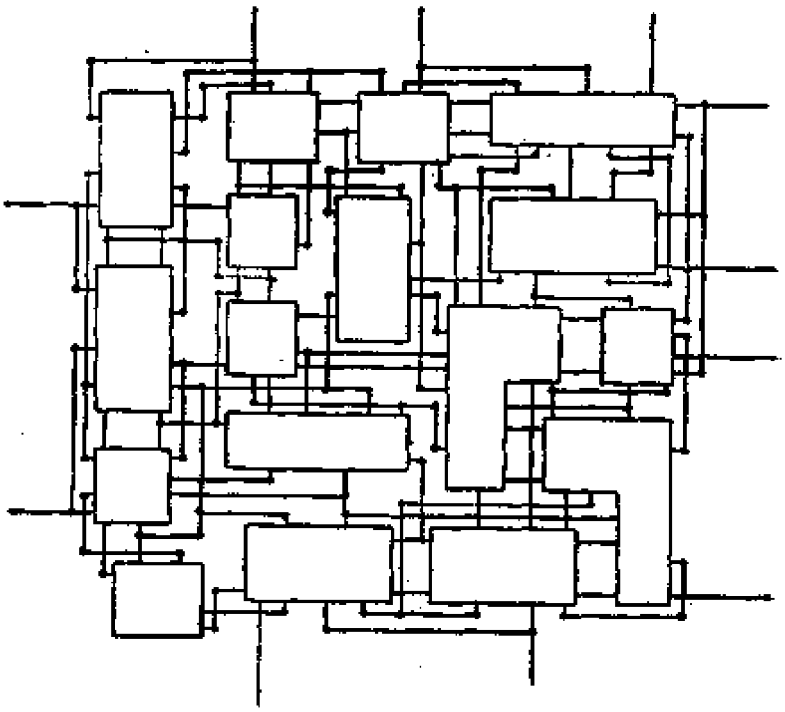


图 2.9　任意元模式的布图

2.5　可编程序逻辑阵列模式

一、可编程序逻辑阵列设计模式及其特点

可编程序逻辑阵列（PLA）模式通常面向随机的组合逻辑电路。设计时首先通过逻辑综合和转换，将电路等价地转换成一个规则的阵列式电路，使其输出成为输入信息的“与—或”函数。这样一来，电路就可用一个“与矩阵”和一个“或矩阵”来表示。从这种思想出发，基本的 PLA 将具有极其规则的布图结构，即可由二个分别称为“与阵列”和“或阵列”的类似只读存贮器（ROM）的阵列来组成它。“与阵列”的垂直线对应着电路的输入信号线，其水平线为“与阵列”的输出信号线。通过水平线与垂直线交叉点上器件的适当选择与安置，使得水平线成为输入信号的乘积项输出线。然后乘积项信号线作“或阵列”的水平输入，并用类似的方式形成“或阵列”的垂直输出线。阵列中器件的选择安置非常类似于

ROM 的编程(见图 2.10)。

二、可编程序逻辑阵列设计技术的优缺点

个阵列中的器件密度。另外还可以通过适当的归并，压缩冗余的布图空间，进行所谓的“折迭”以提高布图的密度。图 2.11 是图 2.10 经过“折迭”优化后的结果[19]、[20]、[21]。

2.6 其它布图设计方式

一、符号法设计方式

1. 设计方式特点

符号法设计方式(symbolic layout)[22]、[23]，是一种用计算机辅助设计的布图方式。从传统的人工布图设计过程中不难发现，即使对于一个极其复杂的电路，就其基本的元件、器件、连线、接触孔、接点等布图元素而言，其布图设计也会存在着不少共同性或实现规范化的可能性。符号法布图设计方式就是利用一个布图符号集，对布图设计中可规范化的布图元素进行描述，以降低传统人工设计中的复杂性和重复劳动，用以提高设计效率和缩短设计周期。同时减少出错的可能性并使设计正确性验证较易于进行。

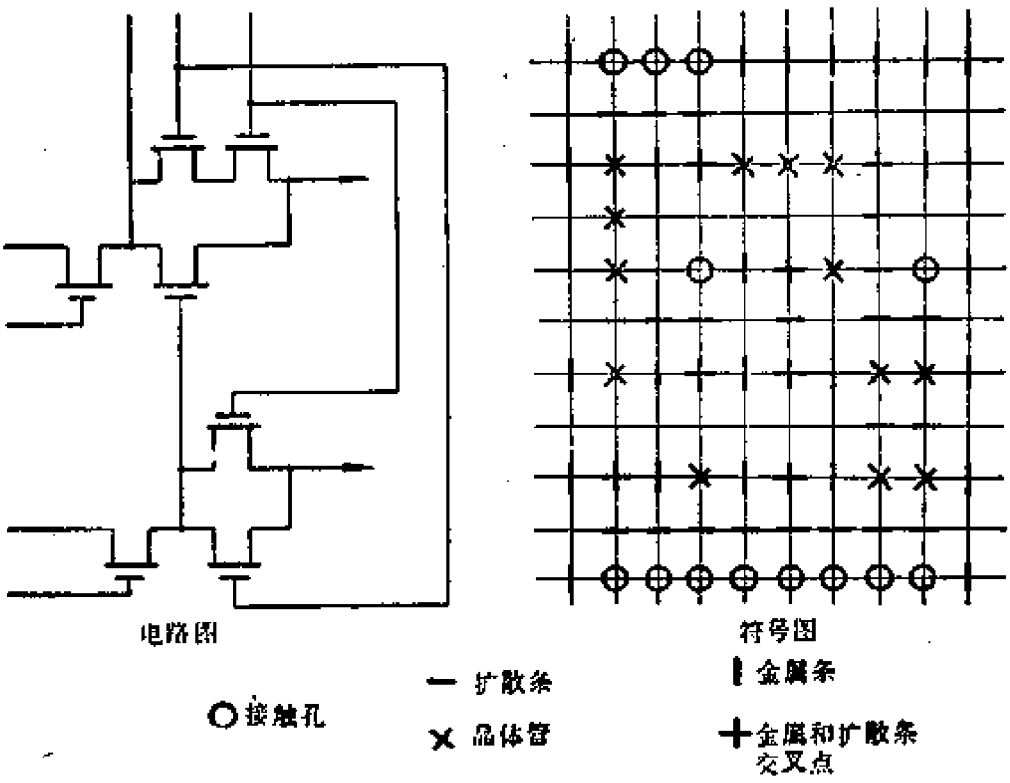
2. 设计过程

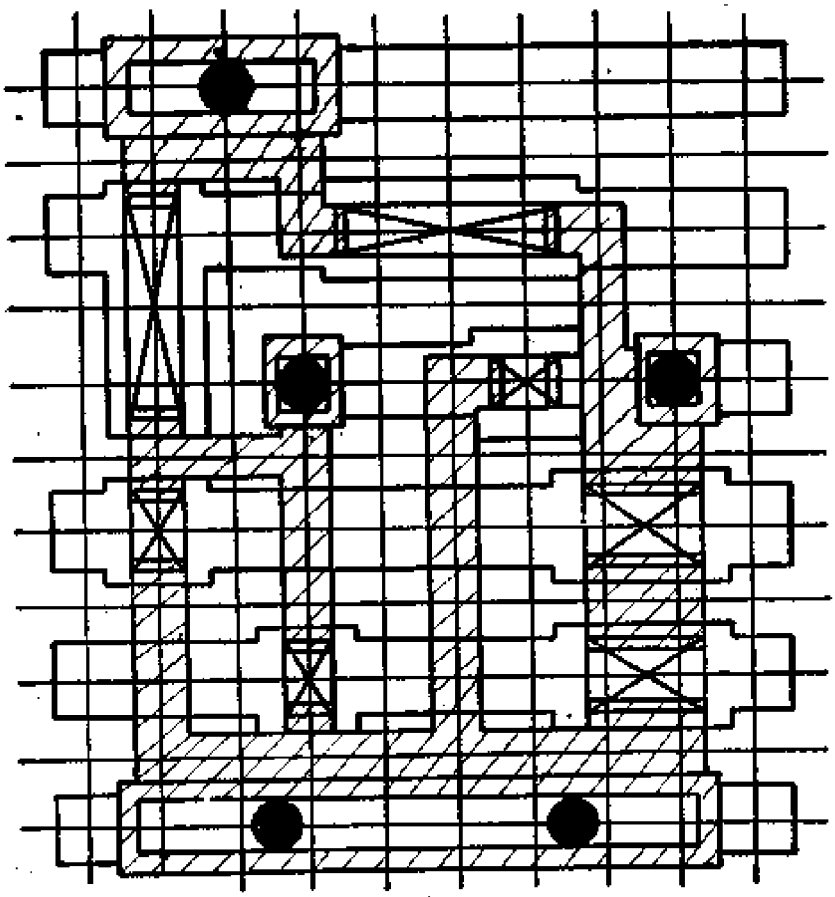
在应用符号法进行布图设计时，首先根据电路和工艺的要求，利用数据库和库定义描述语言，将电路的布图元素定义为符号(当然也可以利用库中已定义的符号)。然后，在依据工艺要求确定的网格场上，再用符号人工地描述电路的布图设计。最后把符号法布图设计的结果由计算机自动地转换成实际的电路布图（见图 2.12)。

符号设计法也可以在相对网格场上进行。当用符号在相对网格场中根据电路布图的相对位置关系画出布图骨架图后，可由计算机自动地进行布图空间的“压缩”(compaction)，并产生实际电路布图(见图 2.13)。

3. 设计方式的优缺点

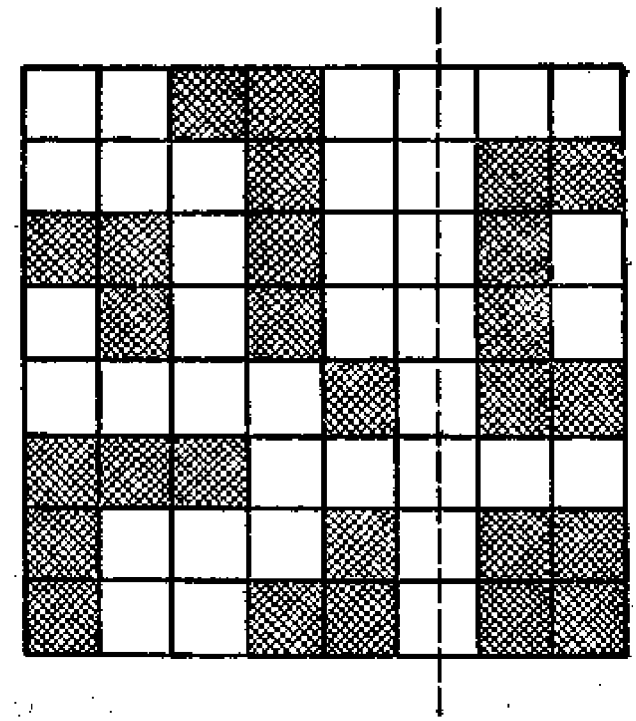
符号法设计方式的主要设计过程仍是人工的，因此可望获得





实际布图结果

图 2.12　符号法布图设计示意图



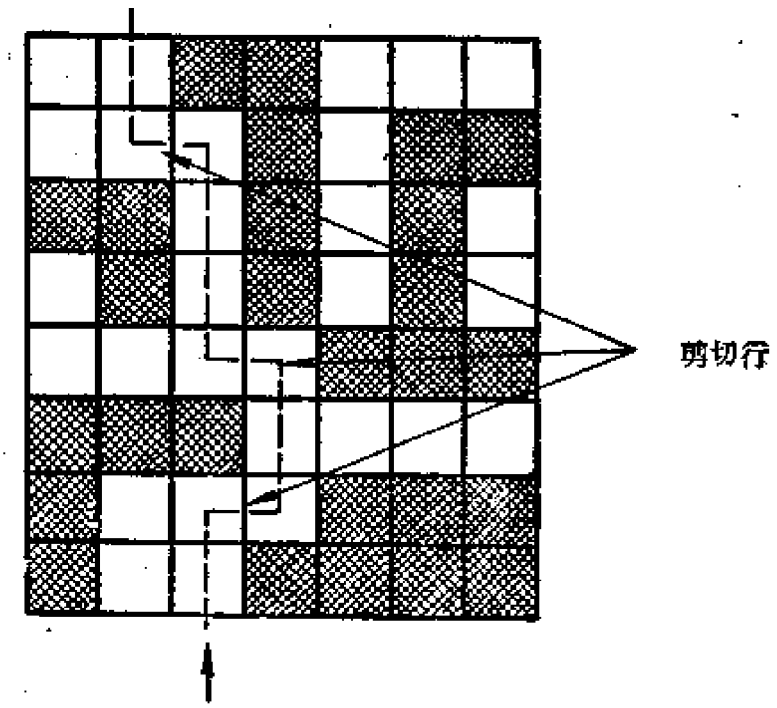
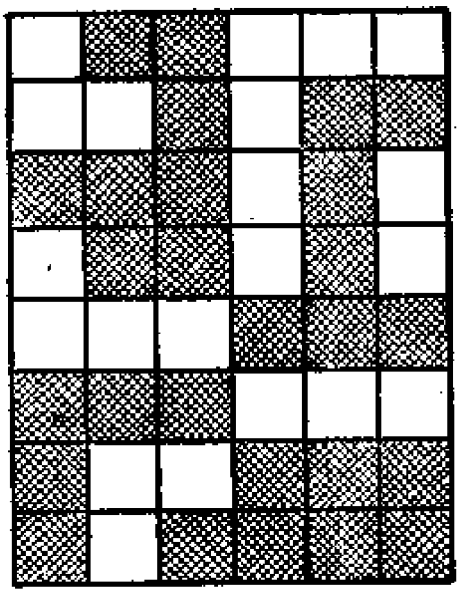
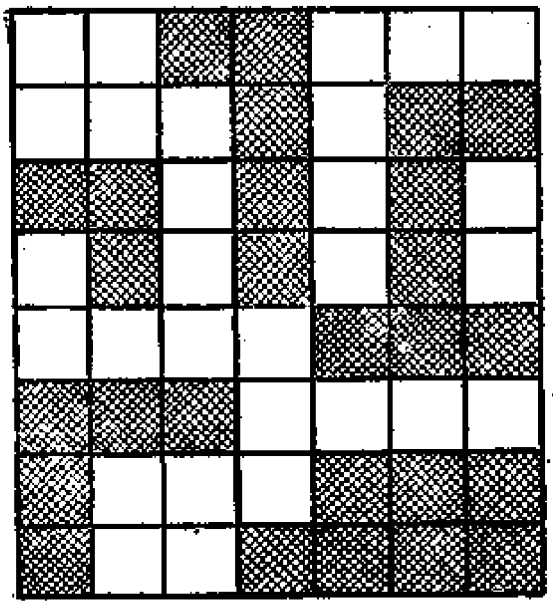


图 2.13　压缩布图空间示意图



压缩的网格布图



压缩的网格布图

图 2.13

较高的布图密度。这种方式适用于中、小规模电路或其它规则电路的布图设计。目前，它是实际生产中的一种重要的设计方式。

二、栅阵列设计方式

1. 设计方式的特点

栅阵列设计方式(gate matrix)[24]，是一种根据 MOS 电路器件工艺特点发展起来的设计方式。例如在硅栅 MOS 电路中，版图上多晶硅与扩散区相交处将形成 MOS 晶体管。同时，多晶硅条又起着连接该晶体管栅极的连线作用。同样，扩散区也同时起着分别连接该晶体管源和漏的连线作用。栅阵列设计方式就是利用上述器件工艺的特点，试图把电路的晶体管"嵌入"到布线区中去，以提高布图密度(见图 2.14)。

2. 设计过程

栅阵列设计方式中，首先把具有公共输入端的晶体管（如图 2.14 中 1、4、5、8 晶体管)安置在同一条多晶硅线上(图 2.14 中第 1 条多晶硅线)，此时这条多晶硅线的一部分就成为 1、4、5、8 晶体管的栅极。同时这条多晶硅线又是 1、4、5、8 晶体管输入端的公共连接线。把这些多晶硅线和对应着电路输出线的多晶硅线（如图 2.14 中第 3、7 条多晶硅线)平行、等距地排列构成栅阵列的"列"。将不同列的而在电路中存在着串联或并联关系的晶体管的扩散区联结起来，以构成栅阵列的"行"（如图 2.14 中 15、16、18 晶体管的扩散区构成了栅阵列的第一行)。需要的时候，行和列可以是断续的。电路所需要的其它连结可以用铝线或扩散条实现。设计时，在画出对应于电路的布图阵列骨架图后，可由计算机根据工艺条件自动生成实际的布图结果。这种设计方式在某种意义上来讲，类似于 PLA 或 ROM 的设计方式，布图结果也具有阵列式的外形。从原则上讲，这种设计方式也适用于其它 MOS 工艺情况。

3. 设计方式的优缺点

这种设计方式的设计质量与人的设计经验有着较大的关系，

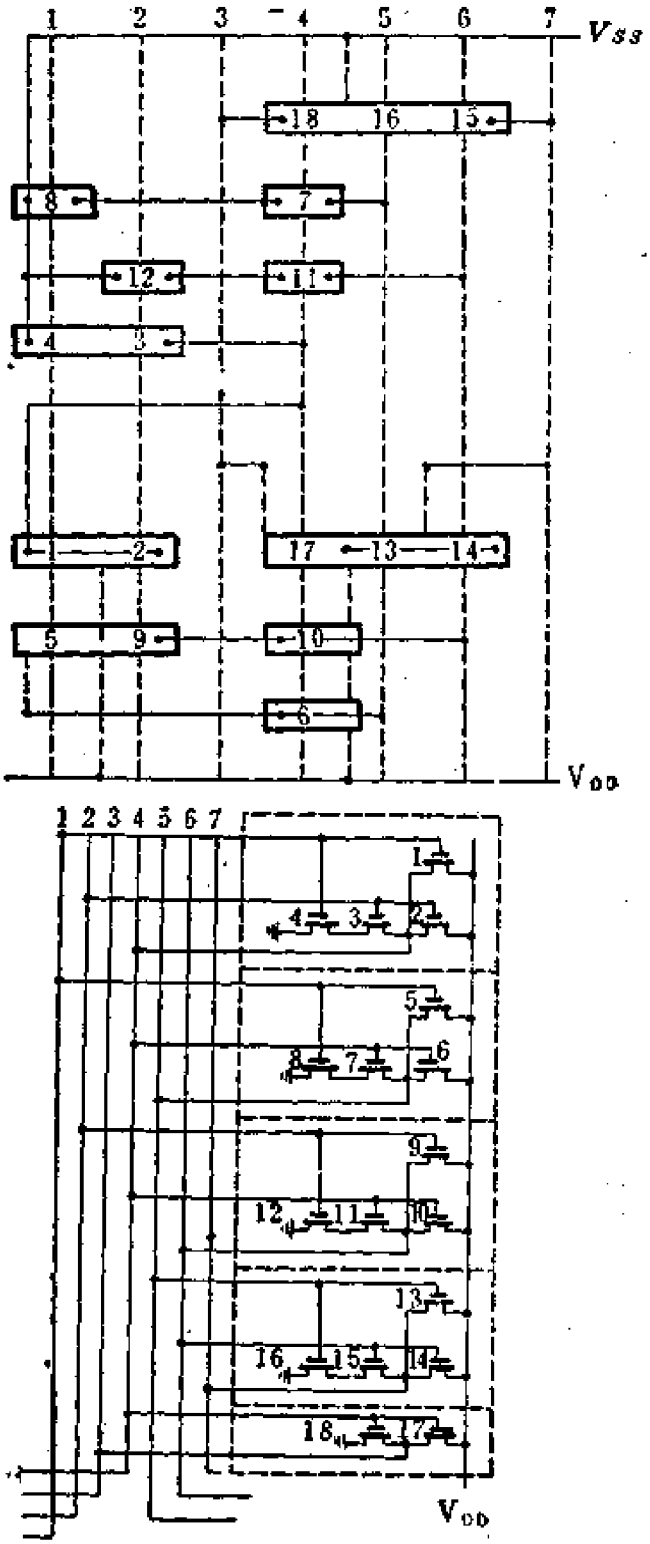


图 2.14　棚阵列方式布图设计示意图

因而存在着如何优化的问题。另外，由于布线通道区兼作元件的安置区域，所以一般来讲可望获得较高的布图密度。这种设计方式尤其适用于面向总线的结构，而不太适合纯随机逻辑的结构。其原因就在于电路中大量的随机连线将占用许多"额外"的布线空间，从而使布图密度下降。鉴于大多数 VLSI 将是面向总线结构的，因此这种设计方式也是一种重要的设计方式。图 1.4 为贝尔实验室 81 年发表的应用 PLA，gate，matrix，polycell 等设计方式完成的一台 CMOS 32 位单片微处理机的布图照片。

三、竖块结构化设计方式

1. 设计方式的特点

竖块结构化(bristle block approach)设计方式[25]，是一种面向计算机结构的分级的结构化设计方式[1]。它可以兼容各种描述方法，例如芯片版图的布图描述、拓扑的布图骨架图描述、晶体管级的晶体管描述、芯片的逻辑描述、芯片文本级的分级描述、用于芯片逻辑模拟的模拟描述以及总线和核心元素排列关系的芯片方块图的描述等。因而它可以使芯片电路的描述和布图编辑易于实现。

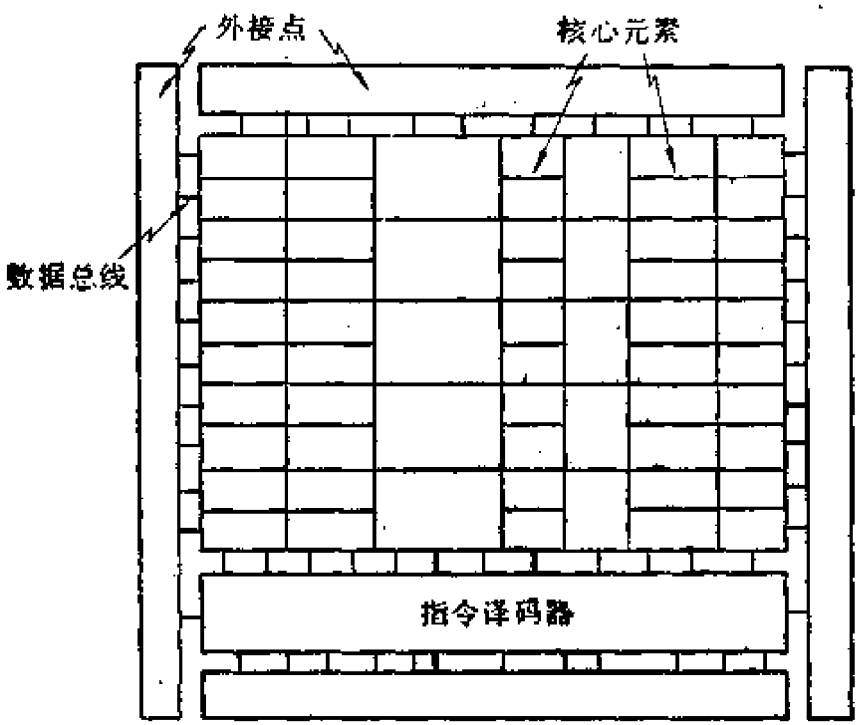
2. 设计过程

在这种设计方式中，电路的位片部分(如算术逻辑单元 ALU、存贮器 RAM、ROM、移位寄存器等)构成竖块结构的核心，数据总线和时钟线设计成核心的水平连线，其控制信号线设计成核心的垂直连线。核心最基本的单元是一种"可伸展的过程性单元"，犹如程序设计者的子程序，可被调用也可根据需要进行修改。其互连接点排列在单元的边上。这些单元被安置在水平线和垂直线的交叉点上。在利用单元设计核心元素时，为使元素中单元的宽度一致化和更好地适应互连的要求，可以调整单元的宽度和相邻单元的互连界面上的接点位置，以便提高布图密度。然后由核心元素组合起来构成结构的核心，其所需要的控制逻辑可用其它方

式(如 PLA) 实现。根据核心的要求决定控制线和外接点的具体位置。

3. 设计方式优点

这种设计方式通过分级设计和结构化设计可望对计算机结构的电路获得较高的布图密度。因而它适用于 VLSI 的布图设计(见图 2.15)。



2.15 竖块式布图设计示意图

2.7 分级设计方式

八十年代，集成电路的发展已迈入了超大规模时代。电路的元器件已达几十万，逻辑门总数已超过一万。集成电路的发展，尤其是 VLSI 的发展，对布图设计提出了新的课题。分级设计方

法就是有效地解决 LSI/VLSI 设计复杂性的方法之一。

一、分级设计方式的概念

对于一个复杂的集成电路芯片的布图设计，其分级设计过程的示意图见图 1.10。

在分级设计方法中，芯片由各级模块组成。从广义来讲，芯片本身也可看作是最高一级的模块。分级设计中的每一级模块由其下级模块组合而成。最低一级的模块就是通常所说的基本单元。

分级设计中模块的划分原则是：

(1) 每个模块完成一定的功能，即模块对应的子电路在电学上、逻辑上是相对独立的。

(2) 使每个模块在内部均有较强的连接性关系，同时使模块间有尽量弱的连接性关系。

二、分级设计中的二种设计方式

在分级设计中通常存在着二种设计方式，自顶向下的设计方式和自底向上的设计方式。

1. 自顶向下的设计方式

自顶向下的设计方式是指先设计最高一级模块，然后设计次一级模块，其设计的过程是逐级向下的。在每一级设计过程中，从上一级模块的布图设计要求出发，在考虑下一级模块布图设计可能性前提下，指定下一级模块的几何形状，尺寸及其在上一级模块中的相对位置，并在考虑各下一级模块的相对位置的基础上，确定便于模块间互连的各模块边界上接点的大致位置。如此逐级向下进行设计，直至下一级模块全部为基本单元为止。

在自顶向下的设计方式中，每一级模块的设计有一个良好的总体考虑，从而为提高总的设计质量带来更大的好处。为此，如何合理地在下一级模块实际设计前，确定对它的要求，以及如何使每

一级的模块布图设计能满足上一级模块提出的要求，这就是设计中要解决的主要问题。

2. 自底向上的设计方式

自底向上的设计方式是指由最低一级模块（基本单元）出发，逐级设计上一级的模块，直至完成整个芯片的布图设计。此时每一级模块的设计结果（形状、大小、接点位置），从原则上来讲，取决于自身设计最佳化目标（如面积最小化，连线总长最短化等）的需要。如何以下级模块为基础求得自身设计的最佳化，是要解决的主要问题。但是由于缺乏总体考虑，往往造成模块间形状和接点位置的不匹配。即使每个模块的设计都趋于"最佳化"，由于上述原因往往也不能保证最后的设计结果会令人满意。因此，在实际的分级设计系统中，往往把自顶向下和自底向上的二种设计方法结合起来，通过多次反复迭代以获得令人满意的结果。

三、分级设计的特点

分级设计方式，原则上讲可使芯片各部分的布图设计，通过若干设计人员的分工并行地进行设计，从而有助于提高设计效率。分级设计方法使各模块的布图设计具有一定的独立性，对于一些模块内的增、删和修改，只要不改变模块的形状、大小和外接点的位置，对其他模块就不会产生影响。而模块的形状、大小和外接点位置的变化也仅涉及到上一级调用到它的模块，因此在每一级的模块设计中，设计所需要的仅是下一级模块的外形、大小和外接点的信息。同时由于每一个下级模块的数据量往往都比较少，从而有利于在人机交互式设计中，设计者对整个设计过程进行有效的控制、评价和修改等。

由于在自顶向下和自底向上设计过程中所设计目标的复杂性，目前分级设计一般需要通过人机交互方式实现。分级设计方式在理论上和实际应用中都还存在着不少令人感兴趣的问题，有待进一步的开发研究[3]。

2.8 各种布图设计方法的应用

一、设计方法的选择

1. 从设计规模考虑

正如其它设计问题一样，在 LSI/VLSI 布图设计中，设计成本和设计效率往往与设计质量是相互制约的。就是说，如果要求得到一个高质量的设计结果，往往需要付出较高的设计成本，同时使设计效率降低。相反，若要求以较低的设计成本和较高的设计效率获得设计结果，则往往不得不在设计结果的质量方面作出某种牺牲。因此，在实际应用时，应从实际问题的具体要求和问题的规模等具体情况出发，对采用的设计方法进行适当的选择。

图 2.16 所示为各种设计方式与问题规模的定性关系。从图上不难看出，当问题的规模比较小时，如设计一些象门或触发器等类的基本单元，传统设计方式的设计成本不高，而且设计质量却相当好。而为了获得较好的设计结果，自动设计方式或交互式设计方式都可能要付出较高的设计代价。因此，一般在小规模集成电路设计中，尤其是基本单元的布图设计中，往往采用传统设计方式或人机交互式设计方式。对随机逻辑电路来说，当问题的规模逐渐增大时，传统设计方式的设计成本将愈来愈迅速的增加，甚至达到令人不能忍受的程度，相比之下，自动设计方式或人机交互式设计方式则愈来愈显示出优越性。从目前的设计水平来看，交互式设计方式虽然由于容易引进差错等原因，而使得设计成本增加，但是一般来讲，由于人机交互系统可以更灵活地体现设计者的思想，因而就有可能获得一个更好的结果。由此看来，自动设计方式和人机交互式设计方式更适合于 LSI/VLSI 的设计任务。

2. 从设计产品的产量考虑

在图 2.17 中显示了各种设计方式与所设计的产品总产量的关系。芯片生产中平均每个晶体管的成本 C 可以大致地用下面的

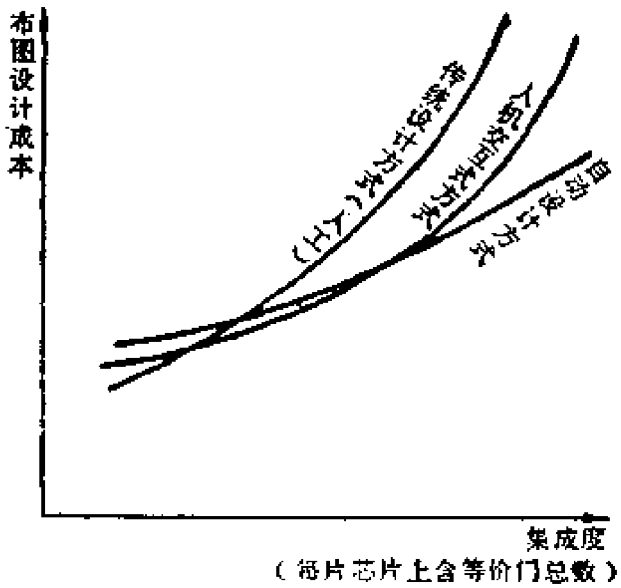


图 2.16　设计方式与设计规模关系图

公式进行计算：

$$C=\frac{(\text{芯片设计成本})+(\text{单个芯片生产成本})\times\text{总产量}}{(\text{芯片上晶体管总数})\times\text{总产量}}$$

当产量很低时，芯片的设计成本将严重影响芯片上每个晶体管的平均成本。当产量很高时，则每个芯片的工艺生产成本将决定芯

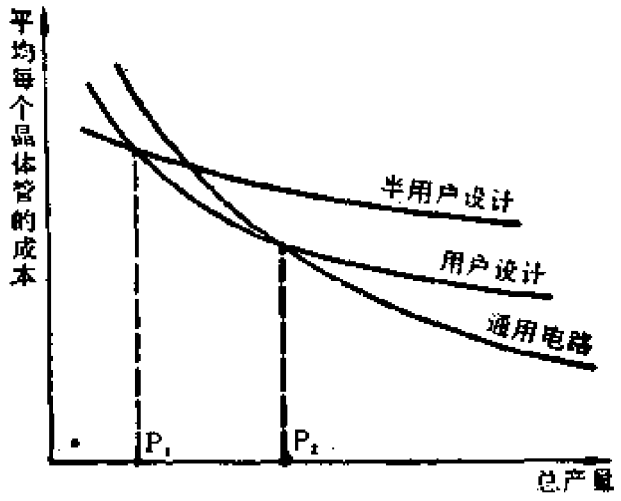


图 2.17　设计方式与产品产量关系图

片上每个晶体管的平均成本，而单个芯片的工艺生产成本与设计质量有着密切的关系。

通用电路由于产量高，因此单个芯片的工艺生产成本将是考虑的主要因素，为了尽可能提高设计质量，通用电路(尤其对于一些"规则"的集成电路，如 RAM，ROM 等)通常采用传统设计方式、符号法设计方式和栅阵列设计方式。

门阵列等设计方式，往往针对采用统一母片的产品，由于母片可以预先进行批量生产，并且当用户提出具体的设计任务时，只需要设计相应的布线及通孔掩膜版便可以生产出用户所需要的芯片，因此有时也称这种设计方式为"半用户设计"。不过，由于这种设计方式要使用统一规格和统一生产的母片去适应多用户的要求，这势必要牺牲一定的设计合理性，然而这种牺牲换得的结果是设计和生产效率的提高以及设计成本的降低，所以这种设计方式特别适用于小批量的芯片生产。

二、"用户设计"方式的特点

完全根据用户的要求进行自动的或半自动的设计方式也称"用户设计"，如多元胞设计方式、任意元胞式设计方式等。它适合于有一定数量要求的芯片设计。从图 2.17 不难看出，这种设计方式的生产量在(P_1, P_2) 之间为最佳设计范围。另外从图中也可看出，对于较大的规模，它的设计合理性优于"半用户设计"，而设计成本低于通用电路设计，并且对于 VLSI 有很好的适应性。因此它是一种有着广阔发展前途的设计方式。随着 DA 技术的发展，它将向着二个方向发展：

① 进一步降低设计成本，提高设计效率。其结果将使图2.17 中 P_1 点进一步向左移动，占领目前"半用户设计"的阵地。

② 进一步提高设计合理性，提高设计质量，其结果将使图 2.17 中 P_2 点进一步向右移动，使愈来愈多的通用电路也适合于用自动和半自动的方式进行设计。

三、提高集成电路生产工艺水平的重要性

值得注意的是，在芯片上每个晶体管的平均成本计算中，当生产量达到一定数量时，单个芯片的生产成本将是一个必需考虑的问题。单个芯片的生产成本取决于设计合理性和工艺水平，以及投入的材料，能源消耗等等，工艺水平愈低下，为了降低单个芯片的生产成本，就必须要对设计质量有较高的要求。换句话说，对布图设计自动化来讲，工艺水平愈低，对设计自动化的要求尤其是对设计质量的要求就愈高。因此，为了发展 LSI/VLSI，一方面要提高设计自动化的水平，另一方面也要提高集成电路生产的工艺水平。

第三章　布图设计自动化对象的定义及描述方法

3.1　单元描述

一、单元的布图描述

在LSI/VLSI布图设计自动化中，单元(cell)通常是设计的基础。单元在电学上对应着LSI/VLSI的功能电路（如各种门、寄存器、全加器、RAM、ROM等）。单元内的布图往往由人工或其它非自动的设计方式完成。这样做，可以获得较精巧的布图密度高的布图设计。因此，单元的布图描述也就是设计结果描述的基础(见图3.1)。

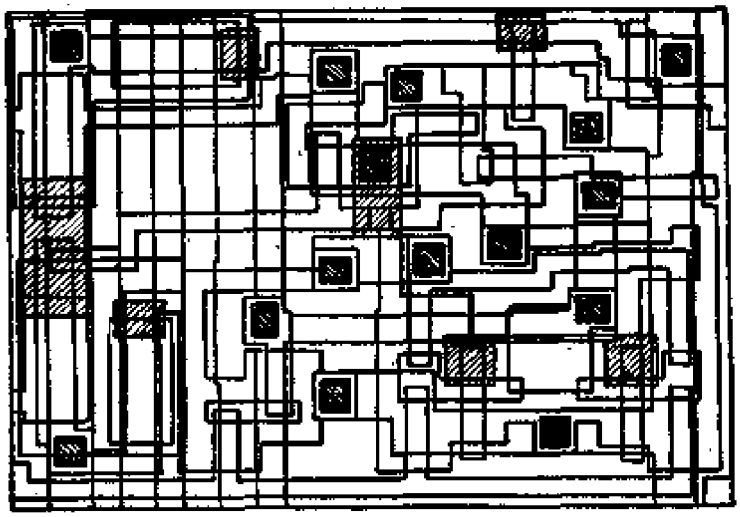


图3.1　由图元构成的掩膜图

我们称构成实际掩膜图案的基本图形为图元，如矩形、斜矩形、平行四边形、圆、环等图形。一般情况下，版图上绝大部分图元是矩形。在有些描述语言中，把各种多边形也可定义为图元。

在布图描述中，对图元的描述一般由三个参量(或参量集)组

成，即图元的形状参数，如矩形的长和宽；图元在版图上的定位参数，如矩形的左下角顶点的 x、y 坐标；图元所在的版图层号说明参数，如该图元在第 2 层。在不同的描述级中，定位参数仅是描述图元相对位置关系的相对坐标，只有当设计完成时才把它们转换成绝对坐标。

例如，对于图 3.2 中所示的矩形图可用如下的描述形式：

层号　REC　x, y, Δx, Δy

其中 REC 是关键字(或称语句标识符，也可以称保留字)，表示描述的图元为矩形，x 和 y 是矩形左下角顶点的坐标，Δx 和 Δy 是矩形的长和宽。具体到图 3.2 可以写成：

2　REC　5, 10, 25, 10

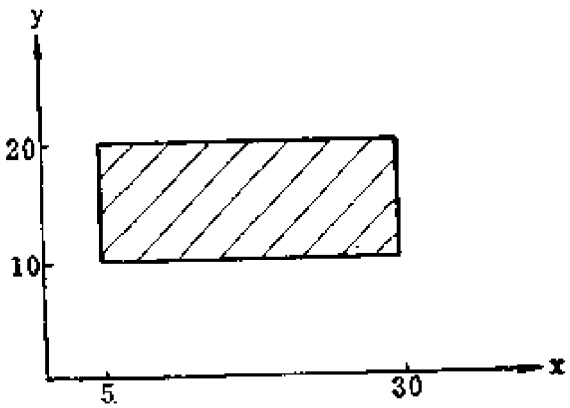


图 3.2　图元描述示意图

单元的布图描述，实际上是通过对其所含图元的逐一描述而得到确定的。

为了描述宏单元(如 RAM、ROM、移位寄存器等单元)的方便，往往还引进一些图元及图元集变换、组合的分级描述手段，以减少描述宏单元所需的数据量和降低差错率。

这些描述手段通常是：

1. 图元集的定义和命名

被命名的图元集将可以在其它描述过程中被引用（类似于一个图元)。

2. 图元及图元集的各种变换

图元及图元集的变换形式很多，如 y 轴镜象变换、x 轴镜象变换、中心对称变换、各种旋转变换(一般为顺时针 90° 旋转变换，逆时针 90° 旋转变换)以及平移等。

例如，图 3.3 中矩形 A 经过两次平移和一次顺时针旋转 90° 到第 3 位置，它的描述方式是：

REC 0, 0, 10, 20; T, 50, 60; T, 80, 60; R_2, 110, 50

其中 T 表示平移，R_2 表示顺时针旋转 90°。

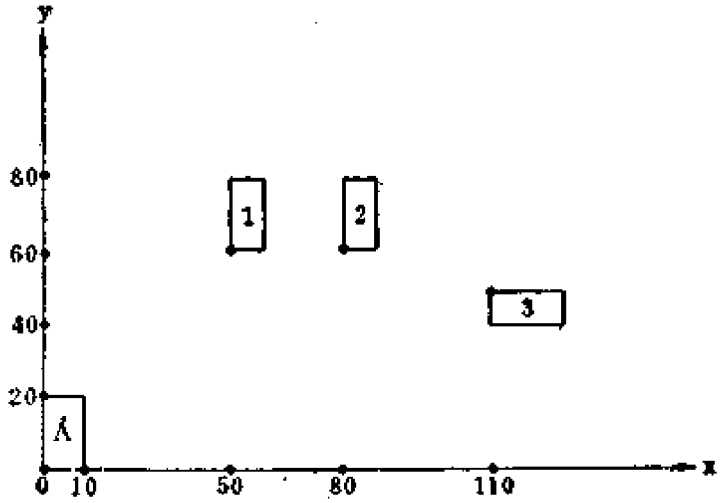


图 3.3 图元变换形式示意图

3. 图元及图元集以一定间距的重复

有了这些描述手段，可以大大简化对一些规则的宏单元的描述。

为了设计资源的共享，基本单元的布图描述可编目装入数据库(单元库)中，以便供其它用户引用。在一些实际系统中，还应该具备对单元库中的单元进行按比例放大或缩小的变换，使库中单元能更好地适应特定用户的布图设计要求。

例如，在描述过程中的任何地方，可以用如下的形式设定比例：

RAT = 整数

此处整数表示比例数值，可以用“+”或“-”表示放大或缩小。若 x 方向与 y 方向比例数值不同，则可用下面的形式描述：

$$\mathrm{RAT}=(Sx, Sy)$$

二、单元的拓扑描述

单元作为 LSI/VLSI 布图设计的基础，设计者所关心的主要不是单元内部的布图情况，而是单元的外形以及单元相互连接点的位置和性质。

单元的外形可由单元的形状参数集加以描述，在大部分情况中，单元的形状为矩形，它的形状参数即为矩形的长和宽。当单元的外形为一个较复杂的直角多边形时（多边形邻边夹角为 $\frac{\pi}{2}$ 或 $\frac{3\pi}{2}$），一种简单的描述方法是：

① 定义直角多边形包络线的逆时针方向为正方向。

② 形状参数集为直角多边形水平边起点的坐标序列。

如图 3.4 即为一个直角多边形单元及其形状参数集。

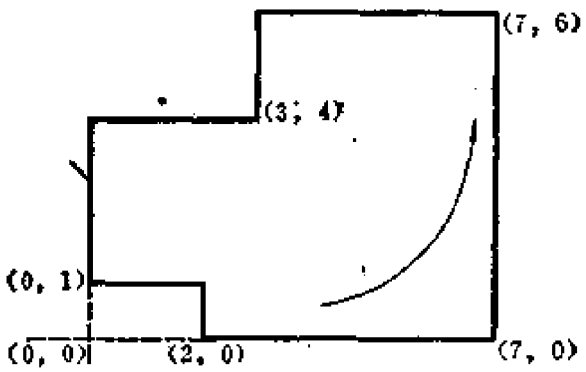


图 3.4 直角多边形单元和形状参数示意图

单元互连接点的位置描述，其中包括接点在平面上的坐标位置（一般为相对单元基点的相对坐标位置）和接点所在的版图层次（如铝线层、多晶硅层等）。互连接点的性质包括各个接点的电学特性（如输入端、输出端、扇入系数、扇出系数等等）和接点间在电

学上、逻辑上的相互关系。这些关系主要有下述三种：

1．逻辑等价关系

如图3.5中接点1和6就存在着逻辑等价关系。具有这种关系的接点在互连时就存在着可置换性。即若原布线要求为：

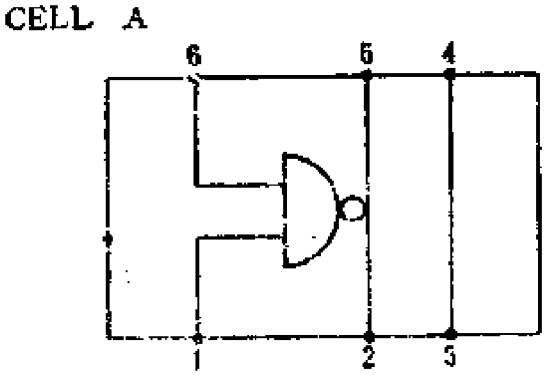


图 3.5　等价接点示意图

1—6　逻辑等价接点(logical equivalence)
2—5　电学等价接点(electrical equivalence)
3—4　过渡通道接点(feedthrough or jumper)

CELL A TER 1——CELL B TER X

CELL A TER 6——CELL C TER Y

则由于CELL A TER 1和CELL A TER 6存在着可置换性，上述布线要求可以改写成：

CELL A TER 6——CELL B TER X

CELL A TER 1——CELL C TER Y

2．电学等价关系

如图3.5中接点2和接点5之间存在着电学等价关系。它们是在单元内部互连的接点，在实现单元互连时，它们存在着可选择性。即：CELL A TER 2——CELL D TER Z 和 CELL A TER 5——CELL D TER Z是等价的。

3．过渡通道接点

这是一种特殊的电学等价关系接点。它们本身在电学上、逻辑上并没有一定的意义，而是为了便于单元间互连线穿越该单元所预先留下的布线通道，如图3.5中接点3和接点4。

3.2 联结关系及其描述方法

联结关系描述是 LSI/VLSI 布图设计对象描述的另一个重要内容。图 3.6 为一个简单的电路逻辑图，并考虑图中 5 条线网：S_1、S_2、S_3、S_4、S_5。在实际问题中，线网的定义是由互连的单元的接点集合来确定的。为了阐述的方便，我们在下文内将把单元的接点编号省略，即将线网定义为与单元相连。因此上述 5 个线网可定义为：

$S_1 = \{A、B、D\}$
$S_2 = \{A、B\}$
$S_3 = \{A、C、D\}$
$S_4 = \{B、C\}$
$S_5 = \{C、E\}$

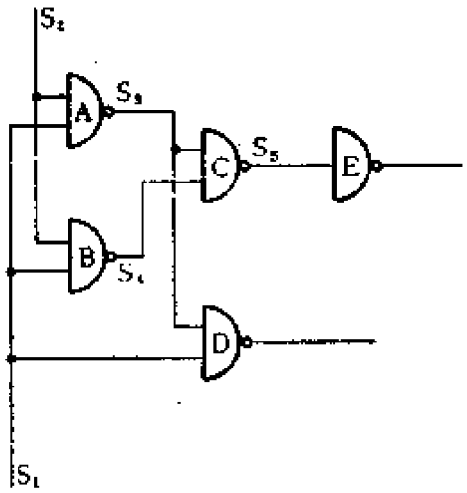


图 3.6 一个简单的电路逻辑图

采用图模型来描述单元的联结关系有二种常用的方法。

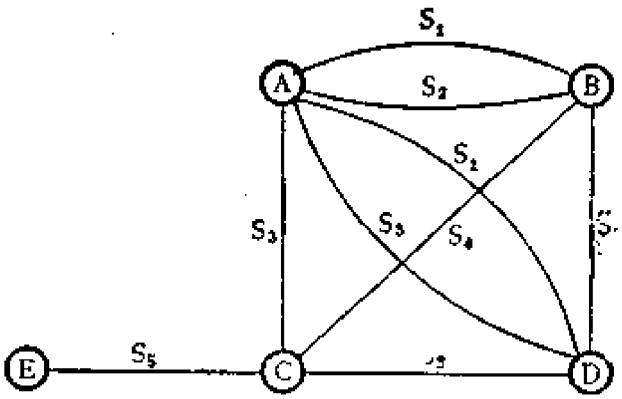
一、线图描述法

1. 描述规则

在线图描述法(linear graph)中，其描述规则是把电路中每一个单元拓扑为图中的一个顶点。当且仅当二个单元同属一条线网时，用一条边联结这二个单元。在实际问题中产生线图的方法，可以通过对每个线网构造其所含单元的完全图来实现，即是使该线网所含各单元间都有一条边相连。然后把这些完全图组合起来而形成所需的线图。

2. 单元邻接矩阵

在图论中，如果图中二个顶点间至少有一条边相连，则称这二个顶点是邻接的。同样，运用线图的概念，如果二个单元同属一条线网，则称这二个单元是相邻的或邻接的。因此线图描述法实质上很方便地描述了单元的邻接关系。如果把电路中的单元与它们的联结关系表示成一个矩阵形式，就得到单元的邻接矩阵。图 3.7 是图 3.6 逻辑图对应的线图和单元的邻接矩阵。



(a) 联结关系线图

	A	B	C	D	E
A	0	2	1	2	0
B	2	0	1	1	0
C	1	1	0	1	1
D	2	1	1	0	0
E	0	0	1	0	0

(b) 单元邻接矩阵

图 3.7 联结关系线图和单元邻接矩阵

3. 线图描述法的特点

由于单元邻接矩阵是对称的，所以在存贮邻接矩阵时仅存贮对角线以上的元素就够了。一般来说，单元邻接矩阵是稀疏的，尤其是对于单元数量足够多的问题更是如此。此时矩阵元素具有非0、非1的其它值，因此存贮空间将较大或存贮手段较复杂。

另外，这种描述方法能很快地检索、判定任何二个单元是否同属一个线网。但是，若要想检索、判定一个线网含几个单元，是哪些单元，每个单元与哪些线网相连，以及相关线网的总数是多少，都不太方便，而这些数据恰恰是很多布局方法所需要的。

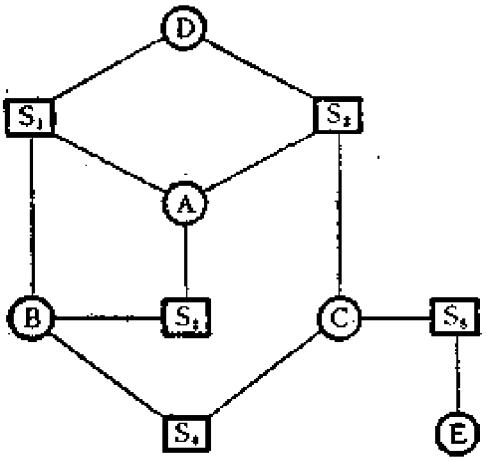
二、偶图描述法

1. 偶图和描述规则

在图论中，如果图 G 能够划分为二个顶点集 V_1 和 V_2，使 V_1

$(i=1, 2)$对应的 G 的子图中不存在联结本集合内顶点的边，则称图 G 为偶图(bipartite graph)，偶图也称二分图。

在偶图描述法中，联结关系的描述规则是把电路中每个单元和每个线网均拓扑为一个顶点，当且仅当单元 x 为线网 S 的一个元素时，才用一条边将顶点 x 和顶点 S 连接起来。把单元和线网与它们间的联结关系用矩阵形式表示，就得到相应的关联矩阵。图 3.8 为图 3.6 表示的逻辑图所对应的偶图描述和关联矩阵。



	A	B	C	D	E
S_1	1	1	0	1	0
S_2	1	1	0	0	0
S_3	1	0	1	1	0
S_4	0	1	1	0	0
S_5	0	0	1	0	1

(a) 联结关系偶图　　(b) 关联矩阵

图 3.8　联结关系偶图和关联矩阵

显然，如果把图 3.8 中所有顶点划分为对应于单元的顶点集和对应于线网的顶点集时，这二个顶点集对应的子图中没有一条边是联结本身集合内的顶点的，因此，该图必是偶图。

2. 偶图描述法的特点

在图论中，图中一顶点所关联的边数称作该顶点的度 (Degree of a vertex)。在描述联结关系的偶图中，对应于单元的顶点的度表示了该单元相关的线网数，而对应于线网的顶点的度表示了该线网所连的单元数。采用偶图描述可方便地检索、判定每个单元与哪些线网相关及相关线网总数，也可方便地检索出每个线网与哪些单元相连及线网所含元素的总和。而且由于关联矩阵的

元素只有0、1二种情况，因此可较方便地采用稀疏矩阵的存贮方法，存贮其关联矩阵，以减少所需要的存贮空间。

三、数据结构的选择

由于LSI/VLSI布图设计的对象规模很大，除了采用分级描述等方法以降低其复杂性之外，适当选择存贮数据的数据结构也是很重要的。通常采用的方法有两种：

1. 邻接编目法

邻接编目法(adjacency vertice listing)使用两个数组，一个是二维数组，它标记与每一个顶点相邻的顶点，另一个是一维数组，它标记每个顶点的度数。参见图3.8，设CN(U、d)是一个$U\times d$的二维数组，其中U是图中的顶点个数($U=10$)；d是图中度数最大的顶点的度数($d=3$)；在CN中，元素CN(I, J)为与顶点I相邻的第J个顶点的标号。在第I行中，与顶点I相邻的顶点可以任意排列（一般以其标号顺序排列）。$VD(I)$是一个一维数组，该数组中第I个元素为第I个顶点的度数。因此图3.8偶图的存贮形式为：

$VD(I)=(3, 3, 3, 2, 1, 3, 2, 3, 2, 2)$

$CN(I, J)=$

S_1	S_2	S_3
S_1	S_2	S_4
S_3	S_4	S_5
S_1	S_3	0
S_5	0	0
A	B	D
A	B	0
A	C	D
B	C	0
C	E	0

单元→线网

线网→单元

这种数据结构，其缺点是存贮空间存在一定的冗余度（二维数组中元素值为零的空间）。当图中各顶点的度数相差较大时，该缺点更明显。

2. 邻接顺序法

邻接顺序法（successor listing）采用二个一维数组 $VN(U)$、$AN(J)$，其中 $VN(I)$ 的维界是顶点总数（$U=10$），每个元素代表图中一个顶点。元素值是一个位置指针，指示在 $AN(J)$数组中存放的和第 I 个顶点邻接的顶点标号集的起始位置。若 $VN(I)=J_i$，$VN(I+1)=J_{i+1}$，那末在一维数组 $AN(J)$中从第 J_i 个元素起存放的是与顶点 J 邻接的顶点，而从第 J_{i+1} 个元素起存放的将是与顶点 $J+1$ 邻接的顶点，于是从 $AN(J_i)$到 $AV(J_{i+1}-1)$的 $AN(J)$中的元素，都是和顶点 I 邻接的顶点。因此图 3.68 中偶图的存贮形式为：$VN(I)=(1, 4, 7, 10, 12, 13, 16, 18, 21, 23)$

$AN(J)=(S_1, S_2, S_3; S_1, S_2, S_4; S_3, S_4, S_5; S_1, S_3; S_5; A, B, D; A, B; A, C, D; B, C; C, E)$

邻接顺序法保持了邻接编目法的邻接关系清楚的优点，又节省了存贮空间，而且易于编写分析图的程序。

3.3 系统描述

一、系统描述的内容

布图设计与电路要求和工艺要求都有着密切的关系，在布图设计中就要反映出它们的要求。这些要求有如下一些方面：某些线网的连线长度要求，线网实际布线时的布线方式要求（最小链接树方式，最小树方式等），某些线网的线宽要求（如地线，电源线等）等。这些要求除可以直接加以描述外，往往还可用赋于线网不同权重的办法加以描述。在有些情况中，尚有一些为了系列化和标准化而提出的要求，如对芯片的外接点位置的一定要求等等。

这些有关设计要求的描述，单元描述，单元联结关系描述，单

元类型描述等，就构成了整个系统的描述。

二、系统描述方法

在分级设计方式中，系统的描述也将采用分级描述的方式。而其基础是模块的描述。每个模块的描述包括：模块的名称、级别、模块的形状参数、模块的外接点信息、模块内部所含下级模块的调用信息、互连结果描述及模块定位信息等(图 3.9)。

在自顶向下设计时，模块的形状参数是对一个预分配模块边界的描述，其外接点位置信息也是一个在一定范围内可调整的信息集。只有基本单元——最低一级模块形状参数和接点信息是确定的，当经过反复迭代后(自顶向下、自底向上)，模块的各种描述信息才能得到最后的确定。

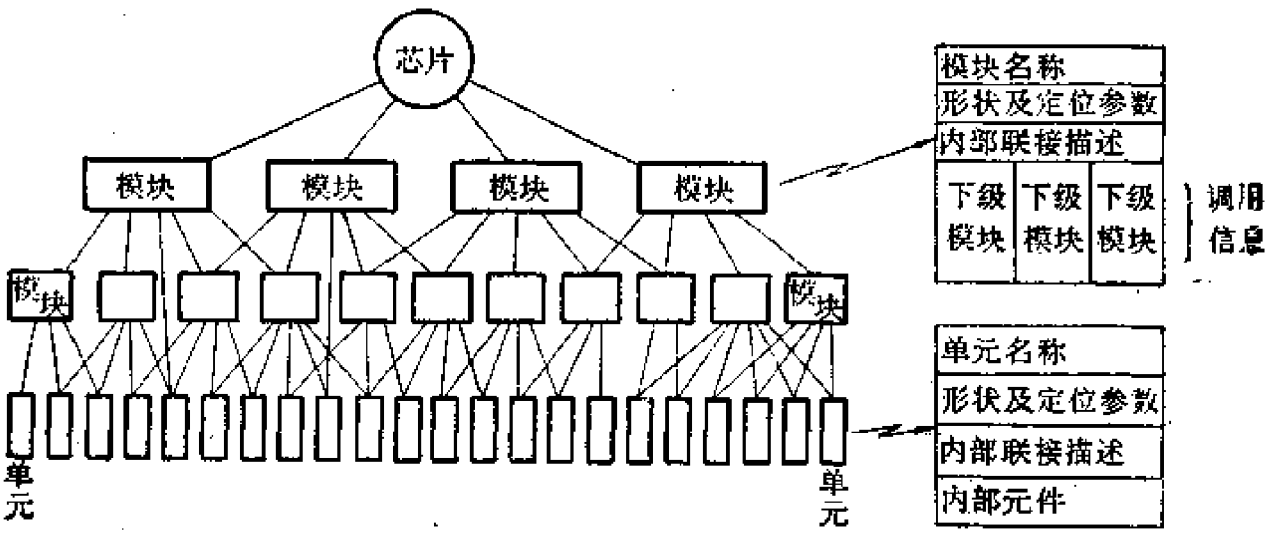


图 3.9　分级描述结构图

最后，我们以一个 16 位移位寄存器的设计为例，使用系统结构描述的层次语言 (SDL) 进行分级结构的描述 [27]，如图 3.10 所示，假定这个 16 位移位寄存器由两个现成的中规模集成电路 (MSI)器件组成，这种器件为一个 8 位移位寄存器 SN7491，那么该 16 位移位寄存器就可以在基本器件功能上进行描述：

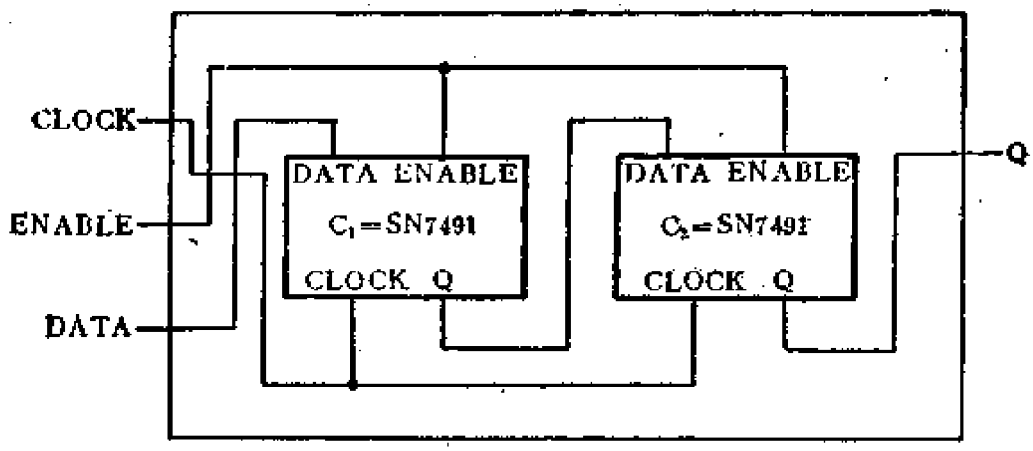


图 3.10 16位移位寄存器

```
NAME:SHIFT16;
PURPOSE: LOGSIM, PCBGEN, CKTANALYSIS;
LEVEL: TTLPACK;
TYPES: SN7491;
EXT:: DATA, ENABLE, CLOCK, Q;
SN7491: C1, C2;
NET1=FROM (.CLOCK) TO (C1.CLOCK, C2.CLO-
     CK);
NET2=FROM (. ENABLE) TO (C1. ENABLE, C2
     .ENABLE);
NET3=FROM(.DATA) TO (C1. DATA);
NET4=FROM(C1.Q) TO (C2.DATA);
NET5=FROM(C2.Q) TO (.Q);
END;
```

说明：

① 无论是系统、子系统或具体的器件等都应该加以命名，以便被高一层的描述引用。如：

```
NAME: SHIFT16;
```

就是把要设计的16位移位寄存器命名为SHIFT16。

② 对每一种描述都应该指出它的用途，例如用于寄存器传输

级模拟、印刷电路板布局、集成电路掩膜布局、门级逻辑模拟等。如：

PURPOSE, LOGSIM, PCBGEN, CKTANALYSIS;

就指明了用两个8位移位寄存器所描述的16位寄存器可以用于逻辑模拟、印刷电路板的产生及电路分析。

③ LEVEL: TTLPACK; 定义了该结构描述的层次。层次与用途有关，它由用户决定。典型的如门级、电路级、寄存器级、系统级等。对于每个由设计者指定的目的与层次的组合PURPOSE·LEVEL，数据库中都有以它为名的信息块，其中含有下一层的详细信息，这些信息就是前层所用的全部类型。

④ 必须明确说明描述中所用到的器件类型和它的对外连接点的名称。如：

```
TYPES: SN7491;
EXT:: DATA, ENABLE, CLOCK, Q;
```

⑤ 此外，还要指出每种类型所含的器件以及反映它们内部连接关系的线网。如：

```
SN7491: C1, C2;
NET1 = FROM (.CLOCK) TO (C1.CLOCK, C2.CLO-
       CK);
.....
NET5 = FROM(C2. Q) TO (.Q);
```

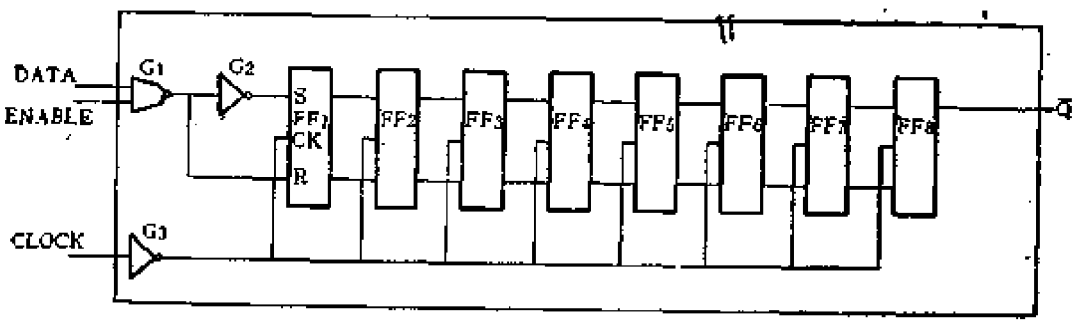


图 3.11　8位移位寄存器示意图

如果设计者要进行门级逻辑模拟，那就需要把它的逻辑翻译成逻辑模拟器能够识别的基本元件，比如与非门、反向器、简单触发器等，这样就可以通过 LOGSIM·TTLPACK 调用 SN7491 型器件的信息。SN7491 型的描述(对照图 3.11)如下：

```
NAME: SN7491;
PURPOSE: LOGSIM, CKTANALYSIS;
LEVEL: GATE;
TYPES: NAND, INV, RS;
EXT::DATA, ENABLE, CLOCK, Q;
INPUTS: .DATA, .ENABLE, .CLOCK;
OUTPUTS: .Q;
NAND: G1;
INV: G2, G3;
RS: FF1, FF2, FF3, FF4, FF5, FF6, FF7, FF8;
NET1 = FROM(.DATA) TO (G1.IN1);
NET2 = FROM(.ENABLE) TO (G1. IN2);
NET3 = FROM(.CLOCK) TO (G3. IN);
NET4 = FROM (G3.OUT) TO (FF1. CK, FF2. CK,
       FF3.CK, FF4.CK, FF5.CK, FF6.CK, FF7.
       CK, FF8.CK);
NET5 = FROM(G1.OUT) TO (G2.IN, FF1.R);
NET6 = FROM(G2.OUT) TO (FF1.S);
NET7 = FROM(FF1.Q) TO (FF2.S);
NET8 = FROM(FF1.QBAR) TO (FF2.R);
NET9 = FROM(FF2.Q) TO (FF3.S);
NET10 = FROM (FF2.QBAR) TO (FF3.R);
NET11 = FROM(FF3.Q) TO (FF4.S);
NET12 = FROM( FF3.QBAR) TO (FF4.R);
NET13 = FROM(FF4.Q) TO (FF5.S);
```

```
NET14=FROM (FF4. QBAR) TO (FF5.R);
NET15=FROM (FF5.Q) TO (FF6.S);
NET16=FROM(FF5.QBAR) TO (FF6.R);
NET17=FROM (FF6.Q) TO (FF7.S);
NET18=FROM(FF6·QBAR) TO (FF7.R);
NET19=FROM (FF7.Q) TO (FF8.S);
NET20=FROM (FF7.QBAR) TO (FF8.R);
NET21=FROM(FF8.Q) TO (.Q);
END;
```

该描述中，把 SN7491 这个器件类型的输入输出脚名分开并加以描述，并指明了该类器件包含一个“与非”门，两个反向器和八个 RS 触发器。

若调用 LOGSIM·GATE，数据库中还应该提供如下的结构信息：

```
NAME: RS;
PURPOSE: LOGSIM;
LEVEL: END;
EXT:: R, S, CK, Q, QBAR;
INPUTS: .R, .S, .CK;
OUTPUTS: .Q, .QBAR;
END;

NAME: NAND;
PURPOSE: LOGSIM;
LEVEL : END;
EXT:: IN1, IN2, OUT;
INPUTS: .IN1, IN2;
OUTPUTS: .OUT;
END;
```

NAME: INV;
PURPOSE: LOGSIM;
LEVEL: END;
EXT :: IN, OUT;
INPUTS: .IN;
OUTPUTS: .OUT;
END;

这里 LEVEL: END;表示最低级别。

运用这样的宏定义，对一个门级逻辑模拟器的输入便可以产生一个描述。

如果要作电路模拟，只需要使用

CKTANALYSIS·GATE

调用相应电路级模块(假定是一个 TTL 反向器）的信息即可。图 3.12 所示的 TTL 反向器的描述如下：

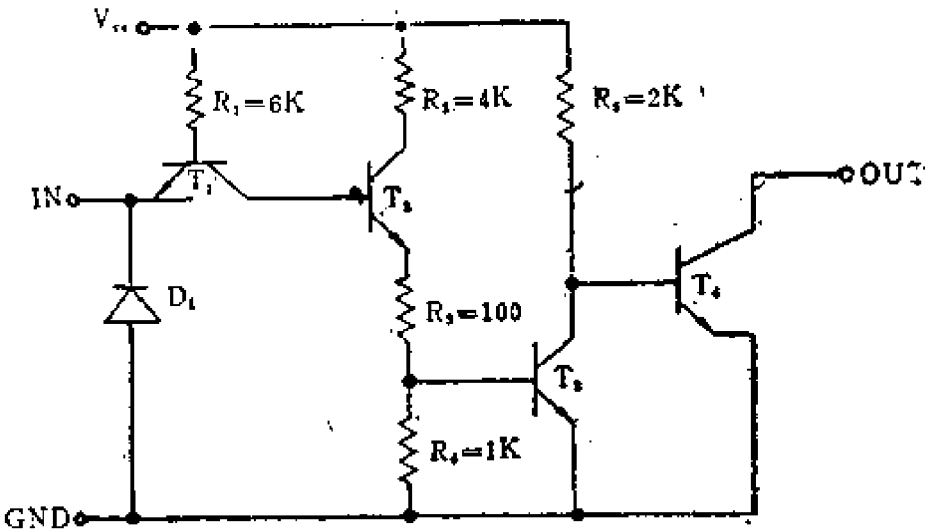


图 3.12 TTL 反向器电路图

NAME: INV;
PURPOSE: CKTANALYSIS;
LEVEL: COMPONENT;
TYPES: TRANSISTOR1, DIODE2, R6K, R4K, R2K,

```
        R1K, R100;
EXT :: IN, OUT, VCC, GND;
INPUTS: .IN;
OUTPUTS: .OUT;
TRANSISTOR1: T1, T2, T3, T4;
DIODE: D1;
R6K: R1;
R4K: R2;
R2K: R5;
R1K: R4;
R100: R3;
NET1 = .IN, T1, .E, D1.A;
NET2 = D1.B, .GND, R4.B, T3.E, T4.E;
NET3 = .VCC, R1.A, R2.A, R5.A;
NET4 = R1.B, T1.B;
NET5 = T1.C, T2.B;
NET6 = R2.B, T2.C;
NET7 = T2.E, R3.A;
NET8 = R3.B, R4.A, T3.B;
NET9 = R5.B, T3.C, T4.B;
NET10 = T4.C, .OUT;
END;
```

假如在电路分析程序中指定了晶体管的一种模型，则使用CKTANALYSIS·COMPONENT调用如下的信息描述：

```
NAME: TRANSISTOR1;
PURPOSE: CKTANALYSIS;
LEVEL: END;
EXT :: B, E, C;
END;
```

从上面的例子已大致可以看出，在分级设计方式中各级模块及其调用关系的描述情况。当然，若从布图设计的角度来看，在各模块的描述中还应该加入模块形状参数、定位信息以及其它工艺要求等方面的描述，并且与各级模块的信息一起存入数据库，以备各种不同设计要求的调用。

第四章　初始布局方法

4.1　引　　言

正如在第一章中对布图设计的定义所阐述的，布图设计可描述为在给定区域内(或尽可能小的区域内)互不重叠地安置电路所需要的全部元、器件或单元并完成其所有必需的互连。在分级设计中，还必需考虑模块的划分和定义。显然，模块的划分、单元的安置和互连都是密切有关、相互影响的，但由于布图问题的复杂性，通常采用分过程进行处理的方法来实现电路的布图设计。

元、器件或单元在芯片上的安置过程称作为布局过程。

一、布局设计的目标

布局设计的实际目标是多方面的：如尽可能保证100%完成所需要的互连；尽量使所需要的芯片面积最小(除母片式，即门阵列模式外)；尽量使一些关键性的信号线网连线长度最短化；尽量使产生的寄生参数影响限制在电路允许的偏差范围之内等等。在研究布局算法时，若要直接去考虑上述目标，实际上是很困难的或者是不可能的，其中有些目标在设计过程中甚至往往是互相冲突的。因此，在实际的布局算法中，往往采用一些容易描述和度量的、而且认为与上述目标有较好的对应关系的目标函数。例如，采用使带权的线总长最短化为目标函数。一般认为，带权线总长最短化能较好地综合反映"使芯片面积最小化"、"100%完成布线"、"减少寄生干扰和连线产生的延时"等目标。对不少实际问题，这是一个相当有效的目标函数，也是一种应用相当广泛的目标函数。我们将在后面几节中分别介绍其它几种布局目标函数。在这里我们要

着重指出，实际布局算法中的目标函数往往只是，而且也只能是实际布局问题设计目标的一种近似。因此，它必须易于度量，而且应该和算法本身相匹配；此外，所有布局问题的设计目标都是与布线可行性联系在一起的。所以也可以讲，布局设计最重要的目标就是为了满足布线设计的需要。

二、初始布局概述

正如第一章中所述，由于布图设计问题的复杂性，设计过程一般采用分过程设计的方法。在需要的时候，每一过程又可以继续分成若干个子过程。在布局设计中，一般可分为初始布局和迭代改善布局二个子过程。本章将着重讨论初始布局子过程。

初始布局的作用主要是：

(1) 求得一个较好的布局初始构形，以减少迭代改善的次数，从而提高整个布局设计的处理效率。一般来讲，初始布局子过程应该是一个处理效率相当高的过程。其所耗费的机时仅为整个布局设计的 $\frac{1}{10}$ 左右。

(2) 由于一般迭代改善布局算法的求解精度与布局的初始构形有相当大的关系，因此期望通过初始布局获得一个"好"的布局初始构形，从而最终获得一个令人满意的布局结果。

需要注意的是，初始布局的对象是设计要求和电路描述，而迭代改善的对象是初始布局的结果——布局的初始构成。对象的不同使它们的目标函数，处理方法也有所不同。为提高处理效率，则必须注意它们的目标函数的一致性。

在初始布局过程中，由于总存在着一些单元，其具体位置无法严格确定，因此前述的布局目标函数(如带权线总长最短化等)实际上都难以度量。所以，对于初始布局来讲，要解决的问题是：每次从当前未布单元集中选取哪个（或哪几个）单元进行安置？或者讲，单元安置的次序应如何决定？对选定的单元（或单元集）在芯

片上应如何加以安置？因此，初始布局算法一般采用快速的构造性算法，并且相应地确定其所采用的选择函数和安置规则。这些选择函数和安置规则往往是启发性的和易于度量的。由于单元的选择函数和安置规则的不同也就产生了不同的初始布局方法。

三、布局设计中的符号和术语

为了讨论的方便，在本节中我们首先对一些符号和术语给出其明确的定义[28]。

ρ_k：第 k 个线网的权。

J_k：第 k 个线网中的单元数。

S_k：第 k 个线网。

$S(x)$：一个线网集合，单元 x 是这些线网的一个元素。

$M(x)$：单元 x 的邻接单元集，即与单元 x 存在公共线网的所有单元的集合。

A^j：在第 j 次选择安置后，已安置单元集。

B^j：在第 j 次选择安置后，未安置单元集。

P_k^j：在第 j 次选择安置后，S_k 中已安置单元数，并有：

$$\alpha_k^j=\begin{cases}1 & 0<P_k^j<J_k\\ 0 & P_k^j=0,\ J_k\end{cases}$$

（当且仅当在第 j 次选择、安置后，S_k 中至少有一个单元已安置，且至少有一个单元未安置时，$\alpha_k^j=1$）

$$R^j(x)=\{S_k/S_k\in S(x)\text{且 }\alpha_k^j=1\}$$

$P(x,\ y)$：二个不同单元 x 与 y 之间的联结度。

$$P(x,\ y)=\begin{cases}\sum\limits_{S_k\in S(x)\cap S(y)}\rho_k & x\neq y\\ 0 & x=y\end{cases}$$

当考虑线网元素数的影响时，可定义为：

$$P_\lambda(x,\ y)=\begin{cases}\sum\limits_{S_k\in S(x)\cap S(y)}\left(\dfrac{J_k+\lambda}{J_k}\right)\rho_k & x\neq y\\ 0 & x=y\end{cases}$$

其中，λ参数反映了线网元素数对联结度的影响。λ越大，元素数少的线网对联结度的影响愈大。而负的λ则使元素数多的线网的影响变大。由于J_k的最小值为2，所以λ的下界为-1。当λ为0时，$P_0(x, y) = P(x, y)$。

在文献[29]中，线网元素数为J_k的多点线网，当用链接树方式互连时，可用J_k-1二元素线网来表示，则单元间的联结度可定义为：

$$P_0(x, y) = \sum_{gk \in g(x) \cap g(y)} \rho_k/(J_k-1)$$

从另外的角度来看，我们用多点线网的完全图来描述线网，这时图中每条边的联结度为$P_0(x, y)$。

联结度的实际意义是描述了单元x和y之间的加权公共线网数。当$\rho_k=1$，$J_k=2$时，即为单元x和y之间的公共线网数，也即单元x和y之间联接的紧密程度。

在本书中，为了描述方便，在几种定义都可使用时，我们将都采用$P(x, y)$来表示单元x和y之间的联结度。

四、布局设计中连线长度的计算

布图设计中的连线长度通常采用所谓曼哈坦(Manhattan)距

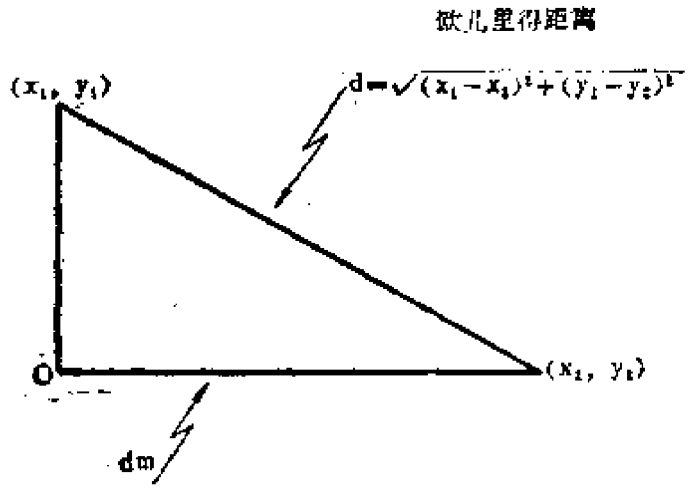


图4.1 曼哈坦距离示意图

离的方法进行计算。在直角坐标系中，两点(x_1, y_1)，(x_2, y_2)间的曼哈坦距离dm定义为：

$$dm = |x_1 - x_2| + |y_1 - y_2|$$

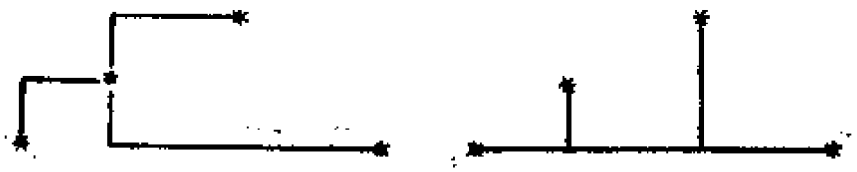
图4.1是曼哈坦距离示意图，采用这种定义的好处之一是可使距离（两点间连线长）的计算比计算欧几里得距离快一个数量级。值得注意的是，采用这种计算方法后，两点间的最短路径一般不是唯一的。

在布局设计中往往需要估算完成布线后的连线长度。我们知道，在布线实际完成前，要算出实际连线长度，对于复杂的问题是根本不可能的，因此通常采用近似计算的方法。对于一条m个接点的线网，设覆盖这些接点的最小矩形的长和宽分别为L和W，则该线网连线长的近似值$l_k = L + W$（即矩形半周长）。

l_k实际上是线网k在不考虑其它线网的影响下，以直角最小斯坦尼(steiner)树形式连接时连线长的下界值[30]。所谓直角最小斯坦尼树就是在互连时允许有非接点的连线交点存在的最小树（见图4.2(b)）。由线网的统计可知，大约75%的线网只有2～3个接点，此时l_k等于直角最小斯坦尼树的长度。对于4～5个接点的线网，文献[30]证明了其最小斯坦尼树连线长的上界为$2L + 4W$。l_k的误差是有限的。对于具有很多接点的线网（如地线、时钟线），由于其几乎和每个单元相连，它的计算误差对设计过程几乎没有什么影响，同时对于已确定接点位置的线网，计算l_k也比较简单，因此，我们一般使用l_k作为线网实际布线后连线长的近似值。

对于接点数大于3的线网，若为了求得一个误差更小一些的近似值，可采用其最小树（如图4.2(a)）连线长度作为近似值l'_k，l'_k是一个比l_k更好的实际连线长的界。它一定小于线网以最小斯坦尼树方式连接时的长度的1.5倍。而且l'_k的计算速度将比l_k快[31]。

在初始布局中，当有些单元尚未安置时，为了确定当前的被选



(a) 斯坦尼树　　(b) 最小树

图 4.2　斯坦尼树和最小树

单元的安置，就需要计算相关的连线长度。这时往往只考虑已安置单元和被选单元的互连情况，每个线网的连线长 L_P 为覆盖该线网中已安置单元接点的最小矩形的半周长。

4.2 对联结法

一、对联结法的基本思想

对联结法 (pair——linking method)[32] 的思想是非常直观的，它的选择原则是每次从未安置单元集 B' 中，选择与已安置单元集 A' 中某一单元联结度最大的单元来进行安置。文献[11]中就采用了这种方法(联结度定义采用 $P_2(x, y)$)。其选择函数为：

$$f_1(x) = \max_{y \in A'} \{P(x, y)\} \quad x \in B'$$

单元的安置采取所谓的"核心生长法"。即在全部单元开始安置前，选择具有：

$$\max\{P(x, y)\} \quad x, y \in B' \text{ 且 } x \neq y$$

值的一对单元安置在芯片的中心位置。在有些实际系统中，也可由设计者人工地在芯片中心位置预先安置一个单元或单元子集。然后以这些单元为"核心"，根据选择函数从未安置单元集中选择具有最大 $f_1(x)$ 值的单元。每次选出的单元将围绕核心选择适当的邻近已安置了单元的位置进行安置。按照这种办法，逐渐向芯

片的四周扩展和生长，直至单元全部安置完毕。

二、对联结法的主要步骤

STEP 1：在所有单元未安置前，根据用线图描述的联结关系，构造 $n\times n$ 的对称的 P 矩阵（其中 n 为单元总数），矩阵中每一个元素 $P(i, j)$ 为单元 i 和单元 j 的联结度。

STEP 2：选择初始单元时。选择 P 矩阵中最大元素值所对应的单元对作为初始单元对。

当 P 矩阵中最大元素有好几个时，可采取向前看 μ 步的选择策略，其思想可简述为：分别假定选中其中一对为初始单元对，然后以它为基础，选择下一个与它们之中的一个单元具有最大联结度的单元，并累计 $P(x、y)$ 值。若 k 次选择后（$k\leqslant\mu$），累计 $P(x、y)$ 值的最大者为唯一时，则选择其对应的单元对为初始单元对。若 μ 次选择后，累计 $P(x、y)$ 值的最大者仍不唯一，则从它们对应的初始单元对中任选一对进行安置。

STEP 3：初始单元对的安置，采用核心生长法，即将单元对安置在芯片的中央相邻的位置上。在 $P\times q$ 的门阵列模式中，可使单元对的安置位置与长轴方向一致。

STEP 4：未安置单元的选择，其选择函数为 $f_1(x)$。当选出的单元不唯一时，除可采用前述的向前看 μ 步的策略外，也可采用另一个选择函数来选定（参看 4.3，4.4 节）。

STEP 5：选定单元的安置。其安置规则为，使选定单元安置在尽量靠近与它相关的已安置单元邻近的位置上。

在芯片面积、形状固定的设计问题中，显然，选定单元的安置位置不允许超出芯片的范围。在边界浮动的设计问题中，将以期望的芯片形状、和芯片利用率来限制选定单元的安置位置。

当出现与已安置单元有两个或更多的位置是等距的情况时，则计算与选定单元存在联结关系的已安置单元位置坐标的加权平均值：

$$\bar{u} \approx \frac{\sum_{x \in A'} P_\lambda(x, y)(u_x + k)}{\sum_{x \in A'} P_\lambda(x, y)} - k$$

$$\bar{v} \approx \frac{\sum_{x \in A'} P_\lambda(x, y)(v_x + k)}{\sum_{x \in A'} P_\lambda(x, y)} - k$$

其中 y 为选定单元，(u_x, v_x)为单元 x 坐标，常数 k 是用来对于 u，v 从 0 开始计数的补偿因子。

选定单元选择最靠近$(\bar{u}, \bar{v})$点的候选位置进行安置。重复 STEP4 和 STEP5 直至 B' 为空集。

三、对联结法的特点和问题

对联结法的处理效率与单元总数 n、线网总数及线网元素数的频率分布有关，但是其主要影响因素是单元总数，文献[34]已证明对联结法的计算复杂性是 $O(n^2)$，因此对联结法的处理效率是较高的。

对联结法的主要问题是选择函数过于粗糙，仅考虑每二个单元之间的联结度关系，因而在不少情况下，它与布局总的目标函数——带权连线总长最短化的一致性较差。其示意图如图 4.3 所示。

在对联结法中，根据 f_1 选择函数，B' 中下一个被选单元应

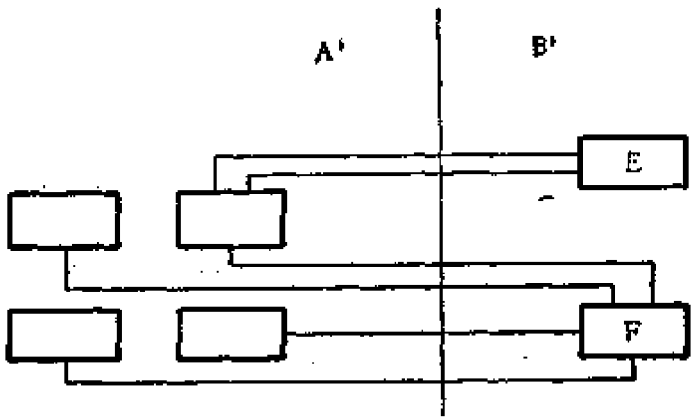


图 4.3 对联结法示意图

是单元 E。但若从布局总的目标来看，在大部份情况中，选择单元 F 将显然比选择单元 E 更合理。

4.3 成群展开法

一、成群展开法的基本思想

和对联结法不同，在成群展开法(cluster—development method)[32]中，以与已安置单元集 A' 具有最紧密的联结关系的单元作为下一个需要安置的单元。其选择函数为：

$$f_2(x)=\sum_{y\in A'}P(x,y)\quad x\in B'$$

在实际问题中，若线网 S_k 的元素数 $J_k>2$，且线网中至少有一个元素已被安置，则当采用链式联结把该线网所有元素连起来时，在 A' 和 B' 间至少存在一条连线，最多存在 T_k 条连线。

$$T_k=\begin{cases}\min[2P_k,\ 2(J_k-P_k)] & P_k\neq\dfrac{J_k}{2}\ \text{且}\ P_k\neq J_k\\ 2P_k-1 & P_k=\dfrac{J_k}{2}\end{cases}$$

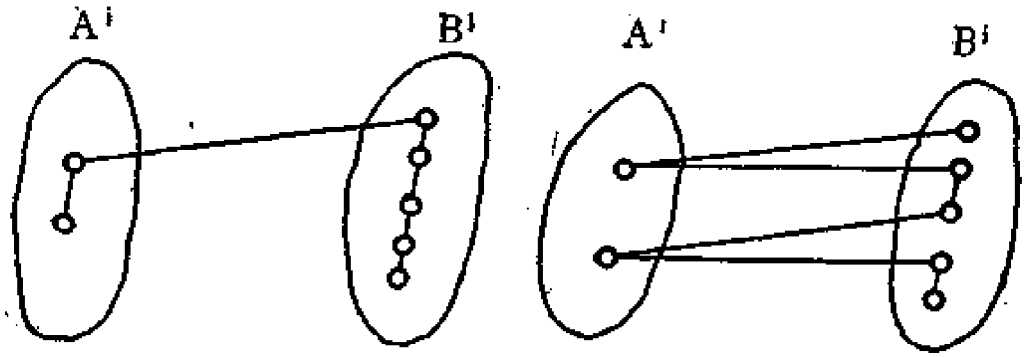


图 4.4 当 $J_k=7$, $P_k=2$ 时线网二种链式联结方式图

图 4.4 为 $J_k=7$, $P_k=2$ 的线网 S_k 的二种极端的链式联结方式。

当采取最小生成树方式联结时，$T_k=J_k-1$。可见对于 $R'(x)$

来讲，线网元素数与已安置单元数的不同，在实际布线时所反映的联结关系也有所不同。考虑到这种情况和计算的方便，实际采用的选择函数可为：

$$F(x)=\sum_{R'(x)}\frac{T_k+1}{T_k}\qquad x\in B'$$

式中 $R'(x)$为含未安置单元 x 的线网集合，且这些线网中都至少有一个单元为已安置单元。

值得注意的是，在 $F(x)$ 的计算中，不再直接以未安置单元 x 与已安置单元之间的联结关系为基础，而是以相关的线网为基础。对属于 $R'(x)$的各线网，在考虑其实际布线可能性的前提下，计算其相应的联结度关系。显然，对于多接点线网的问题，$F(x)$作为选择函数将比 $f_2(x)$具有更多的优点。

$F(x)$的物理意义是，在其它条件相同时，认为一个线网如果其所含元素(单元)愈少，则对应的联结关系就愈重要一些。同时还认为，当其它条件相同时，一个线网所含的元素在未安置单元集 B' 和已安置单元集 A' 中的数量相差愈多，则该线网对应的联结关系就愈重要一些。

当线网的元素数都等于 2，而且线网权重也定义为 2 时，则可以看到，$F(x)$和 $f_2(x)$是完全等价的。

成群展开法的安置规则，也是采取核心生长法。其主要步骤如下所述：

二、成群展开法的主要步骤

STEP 1：在所有单元未安置前，根据用线图描述的联接关系，构造 P 矩阵。

STEP 2：选择初始单元对。选择 P 矩阵中最大元素值所对应的单元对，首先安置它们。当这样的单元对不唯一时，可按照对联结法中 STEP 2 一样处理。

STEP 3：初始单元对的安置。其安置法与对联结法相同。

STEP 4：未安置单元的选择。当其选择函数为 $f_2(x)$ 或 $F(x)$，即选择当前 B' 集中具有 $\max F(x)$ 值的单元 x 进行安置。当选出的单元不唯一时，除可采取向前看 μ 步的策略外，还可以采取下述复合的选择函数：

$$f_3(x) = f_2(x) * Q + f_1(x) \qquad Q \gg f_1(x)$$

$f_3(x)$的物理意义是，当 $f_2(x)$不同时，因 $Q \gg f_1(x)$，选择具有最大 $f_2(x)$的单元为选中单元。当 $f_2(x)$相同时，选具有最大 $f_1(x)$ 的单元为选中单元。

STEP 5：选定单元的安置。其安置规则如同对联结法的安置规则。通常以与选定单元存在联结关系的已安置单元的位置坐标加权平均值，来作为实际安置位置的参考点。重复 STEP 4，和 STEP 5 直至安置完所有的单元。

三、成群展开法的特点和问题

成群展开法也是一种处理效率较高的方法。文献 [34] 证明，其计算复杂性为 $O(n^2)$。而其实际的处理效率将高于对联结法。

成群展开法和对联结法一个共同的问题是，它们的选择函数都仅仅考虑了待安置单元与已安置单元的联结度关系，而根本没有考虑待安置单元与其它未安置单元的关系，如图 4.5 所示。

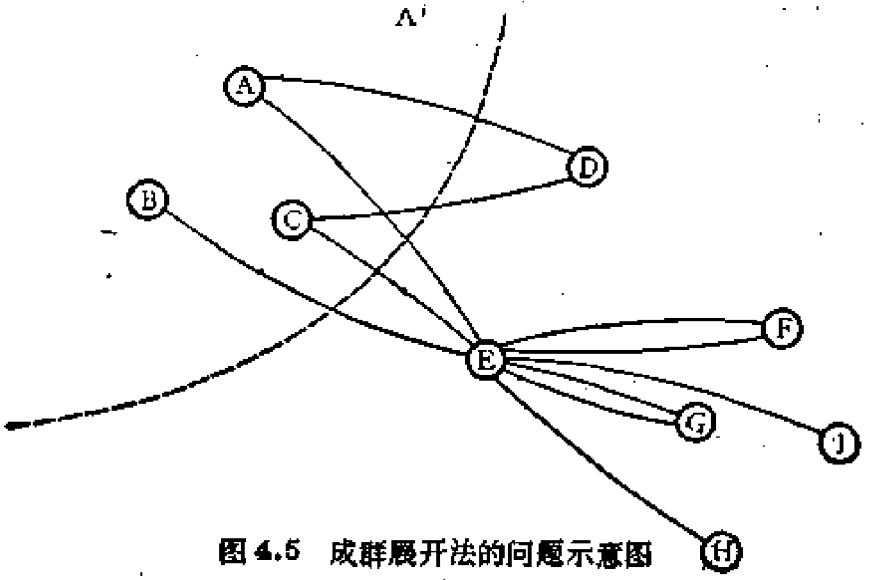


图 4.5　成群展开法的问题示意图

根据成群展开法，下一个待安置单元应为单元 E；对于实际问题，这样往往不尽合理。而选取单元 D 则较为合理。需要说明的是，我们在这里所讲的“合理”是从一种“启发性”的观点出发的，即更多的是从设计者的经验出发的，并不排除在一些情况下可能出现反例。

4.4 “内—外”联结度法

一、“内—外”联结度法的基本思想

“内—外”联结度法 (inside—outside connectivity)是针对“对联结法”和“成群展开法”的不足，以“内—外”联结度作为选择函数，即不仅考虑未安置单元与已安置单元的联结度关系，而且考虑它与其它未安置单元的联结度关系的一种布局方法[35]。

对每一个未安置单元 x，它的“内—外”联结度 IOC_x 的计算方法如下所述：

开始时，令 IOC_x 等于 0；

对于 $S(x)$ 集中各线网元素 S_k，如果 S_k 中除单元 x 外的其它单元都属于已安置单元集，则 $IOC_x = IOC_x + \rho_k$；

如果 S_k 中所有元素都属于未安置单元集，则 $IOC_x = IOC_x - \rho_k$；

显然，当 ρ_k 为 1，IOC_x 为正数时，表示单元 x 上和且仅和已安置单元相连的线网数多于单元 x 上和且仅和未安置单元相连的线网数。

在有的文章中也有采用下述函数作为“内—外”联结度选择函数的：

$$f_3(x) = \sum_{y \in A'} P(x, y) - \sum_{y \in B' \text{且} y \neq x} P(x, y) \quad x \in B'$$

当所有的线网元素数都等于 2 时，$f_3(x)$ 和上述 IOC_x 的计算方式是等价的。

二、"内—外"联结度法的主要步骤

STEP 1：初始单元对的选择和安置基本方法与对联结法同。

STEP 2：未安置单元的选择。按 *IOC* 算法求出各未安置单元的 *IOC* 值，并选择具有最大 *IOC* 值的单元首先进行安置，当候选单元不唯一时，除可采用向前看几步策略外，也可与其它联结度函数构成复合选择函数(如 $f_4(x)=f_3(x)*Q+f_2(x)$，$Q \gg f_2(x)$等)。

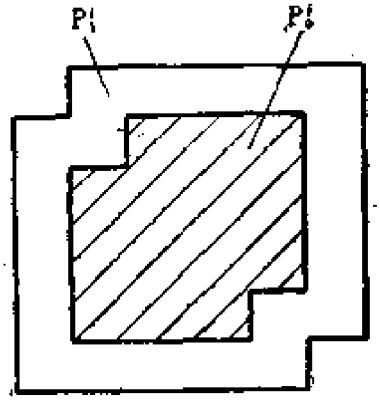


图 4.6 邻接位置集

STEP 3：选定单元的安置。令已安置单元在芯片上的位置集 $P_0^I=\{P_0(M, N)\}$。设 P_1^I 集为 $P_1^I=\{P_1(i, J)\}$满足：

$$P_1^I=\{P_1(i, j)\mid(i, j),\ i=M-1, M+1,$$
$$j=N-1, N+1 \text{ 且 } P_1(i, j)\notin P_0^I\}$$

称 P_1^I 集为 P_0^I 集的邻接位置集，如图 4.6 所示。当计算选定单元若安置在 P_1^I 中的一个位置时，该单元与已安置单元相关的线网连线总长，从中选取能使连线总长最短的位置作为选定单元的安置位置。重复上述 STEP2 和 STEP3 直至全部单元安置完毕。

三、"内—外"联结度法的特点

从这种方法的主要处理过程可以看到，它依据的处理原则也是十分简捷的。对实际问题具有足够高的处理效率。

由于"内—外"联结度法比前述的二种方法具有更多一些的全局考虑，其初始布局结果一般比较好。文献[36]认为这种算法比[32]中所介绍的各种初始布局方法（对联结法，成群展开法等）都好。

四、初始单元对的选择和安置

设计实践表明，在很多问题中，初始单元对的选择和安置对于

初始布局的结果往往有很大的影响。在上述三种算法中，尽管在初始单元对的选择上采取了一些相应的措施，但在实际问题中，尤其是对于规模比较大的问题，这种单一的初始单元对的选择和安置方法都将使布局结果具有较大的随机性。因此，在有些实际算法中，往往采用"多初始单元对"和分区安置的方法。此时芯片的初始布局犹如从多个"核"出发，同时地向四周"生长"，这种方法将可使结果得到一定的改善。多初始单元对的选择可以是人工确定的，也可按下述原则确定：即选择 n 个单元对，使各单元对具有尽可能大的联结，同时使各单元对相互间的联结度尽可能小。安置时，首先把芯片划分为 n 个面积、形状大致相同的部分，然后将选定的单元对分别安置在各部分的中心位置上。

4.5 群 法

一、群法的基本思想

在前三种算法中，单元的选择和安置是同时进行的，而且在构造过程中一个单元一旦被安置在某一位置上，则在构造全部完成前这个单元的位置将不再改变。由于在安置时，往往考虑的仅是其与已安置单元的关系，因此在一定程度上存在着盲目性。群法(clustering)[37]，是一种组合构造方法，它将尽量推迟每一个单元在芯片上具体位置的确定，而是首先将所有单元依其联结关系分为若干群，使群内有紧密的内部联结关系而群间只有相对弱的联结关系，最后在结群的基础上进行单元位置的确定。

二、单元的结群

结群是成对地实现的，每一个结群元素可以是一个单元也可以是一个单元集。结群元素 i 的总联结度为：

$$T_i = \sum_{x \in i, y \in i} P(x, y)$$

结群元素 i 和结群元素 j 的联结度为：

$$C_{ij} = \sum_{x \in i, y \in j} P(x, y)$$

而结群元素 i 和 j 的结群值 CV_{ij} 可由下式定义：

$$CV_{ij} = f(i) \frac{C_{ij}}{(T_i - C_{ij})} + f(j) \frac{C_{ji}}{(T_j - C_{ij})}$$

其中 $f(i)$、$f(j)$分别为元素 i 和 j 的尺寸函数。

1．结群值计算的例子

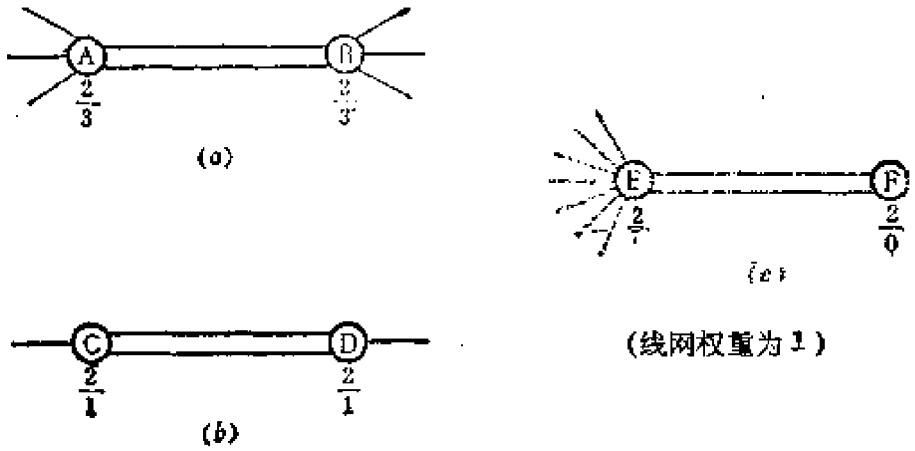


图 4.7　结群值计算的例子

结群值计算的一些例子可见图 4.7，图中 E、F 之间有最大的结群值，在这里值得注意的是每个元素对结群值的贡献是不同的。在图 4.7(c)中，这表示了元素 E 并不十分企求与元素 F 结群，而元素 F 却非常需要与 E 结群，它没有其他对象可供选择。

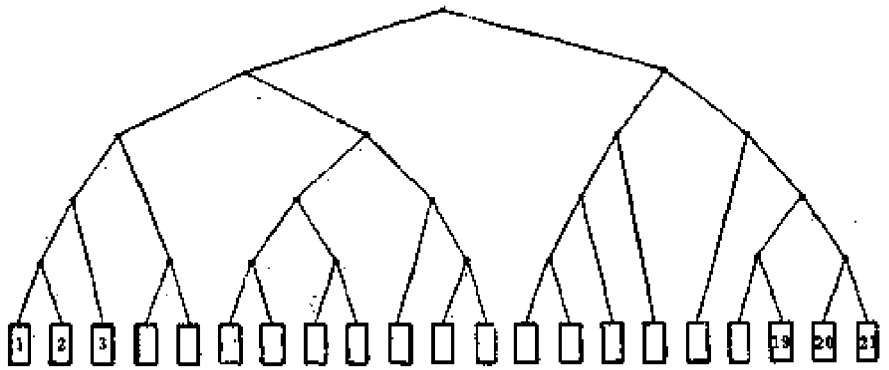


图 4.8　结群二元树(21 个单元)

计算各对元素的结群值并从中选择具有最大结群值的元素对结成群。结群的元素对成为一个新的元素，参加下一步可能的结群。这个过程一直进行到所有的单元结成了一个群，或结成了若干互不相连的群。结群的过程及其结果可用二元树来描述（图4.8）。

2. 结群过程中需要注意的问题

(1) 结群元素尺寸的影响。经验结果表明，当一个元素的尺寸与其它元素相比非常大时，如仅考虑联结关系，则这些大的元素往往很容易被选来进行新的结群，将出现如图 4.9(a) 所示的二元树结构，较小的元素间的结群关系将无法描述。结群值计算中的$f(i)$、$f(j)$的作用就是防止过早构成大的元素，因此 $f(i)$是与元素 i 的尺寸成反递增关系的函数，用于补偿由于元素尺寸不同所出现的不平衡情况(图 4.9(a))。自然，如果这种补偿作用太大了，造成所有的群的尺寸都以同样的速率增长，也不是算法所希望的。

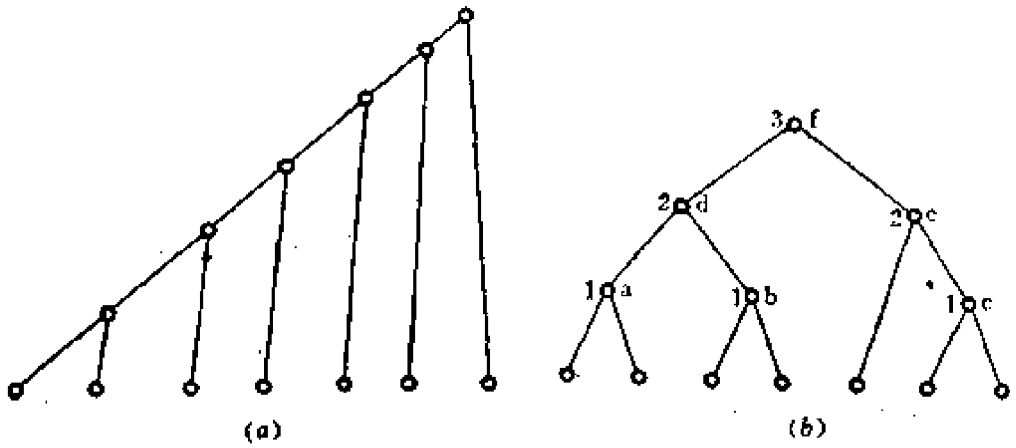


图 4.9　结群元素尺寸的影响

(2) "圈"的影响。成对地结群在处理时效率较高，但在实际问题中，结群的情况要复杂得多。如图 4.10 为一个三元素的"圈"。

从成对结群的观点来看 $CV_{AB}=CV_{AC}=\frac{5}{3}$，$CV_{BC}=\frac{2}{3}$，但实际上，因为图中任二个元素通过第三个元素存在着间接的联结，因此它们的结群值应更大一些。

一种较好的描述方法是定义有效联结度 $C'_{(i)}$，$C'_{(i)}=\sum_{x\in i,y\in j}P(x,y)$ 其中 j 为被定义的含元素 i 的圈中的元素。则图 4.10 中 $C'_A=4$，$C'_B=C'_C=3$。

$$CV'_{ij}=f(i)\frac{C'_i}{(T_i-C'_i)}+f(j)\frac{C'_j}{(T_j-C'_j)}$$

因此，当 $f(i)=f(j)=1$ 时，$CV'_{AB}=CV'_{AC}=4+3=7$，$CV'_{BC}=3+3=6$

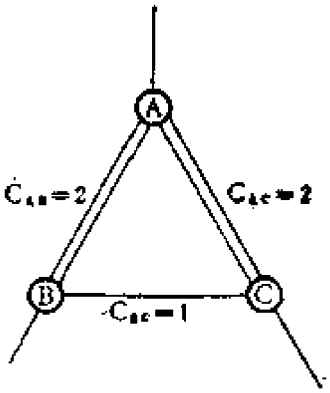


图 4.10 三元素的"圈"

$CV_{AB}=CV_{AC}=\frac{2}{3}+\frac{2}{2}=\frac{5}{3}$

$CV_{BC}=\frac{1}{3}+\frac{1}{3}=\frac{2}{3}$

实际结群处理时，在成对结群前，先定义圈，使它们预先结群。应该注意的是，在其他条件相同时，圈的尺寸愈大愈是相对弱的群。

(3) 结群过程中单元联结度的修正。当采用线图模型描述电路的联结关系并采用 Charneg and Plato 方法[29]描述元素间联结度(图 3.7)时，在同一线网的二个单元结群后将使描述该线网的完全图的顶点数减少一个，因此必须随之修改该线网中元素间的联结度。这样虽然增加了处理的复杂性，但这种联结度定义方法对结群处理是有利的。

三、结群基础上的单元布局

1. 行式布局

当考虑单元成行地进行安置时，一个结群二元树实际上对应着一种行式的单元初始布局。在结群二元树中，与每一个顶点相连的二个子群表示了它们彼此间的联结关系将比它们与其它元素(子群)的联结关系紧密，对应于布局中，则表示它们的安置位置应该相邻。而二个子群绕其顶点转动(互换位置)将在不破坏结群关系的条件下对应一个新的行式布局。对于 n 个单元的二元树，树

的顶点数为 $(n-1)$，不破坏结群关系的行式布局的可能个数为 $2^{(n-2)}$ 个。当 n 足够大时，我们将不可能穷尽所有的可能以寻求一个最佳的布局。在实际处理时，往往采用一些近似的解法(参见第五章)。

2. 分区布局

结群基础上布局的另一种方式是在单元结群时赋于一定的限制条件(如群的大小限制等)，然后将所有单元结成符合限制条件的若干群，再考虑群间的联结关系，决定各群在芯片上的安置位置。在考虑了各群在芯片上的安置位置以后，下一步再决定群内各单元的安置位置，或根据结群关系拆群，先大群，后小群，边拆群，边安置，直到决定了每个单元的位置时为止[35]。

4.6 多元胞模式的初始布局方法

一、多元胞模式初始布局概述

布局问题就其本质来讲是构成电路的单元在二维空间中的安置问题。对于采用等高不等宽单元模型的多元胞模式，为了得到以较高的处理效率求得一个满意的初始布局结果，往往用二个一维的安置问题去代替二维的安置问题。即把芯片上的单元成行地排列构成块(Block)，首先在 y 方向决定每个单元应安置到哪个块中，在确定了每个块的组成后，对每一块决定其块内单元在 x 方向的排列顺序。

二、块和子块(SUBBLOCK)的定义

1. 单边单元多元胞模式中块和子块的定义

在单边单元多元胞模式中，单元"背对背"排列成行。水平通道区上边和下边二行单元构成一个块。这些块又被垂直通道区分割为子块(图 4.11)。

2. 双边单元多元胞模式中块和子块的定义

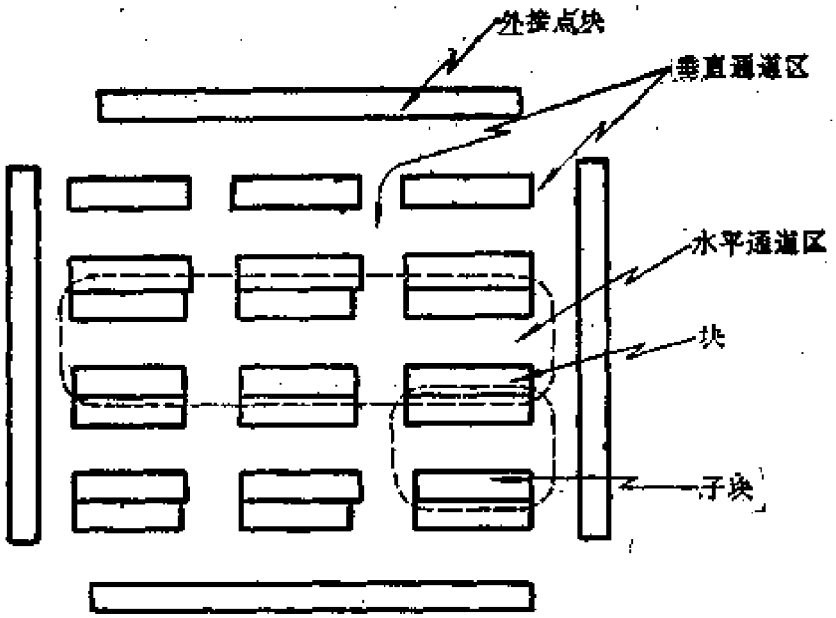


图 4.11　单边单元多元胞模式中块的划分

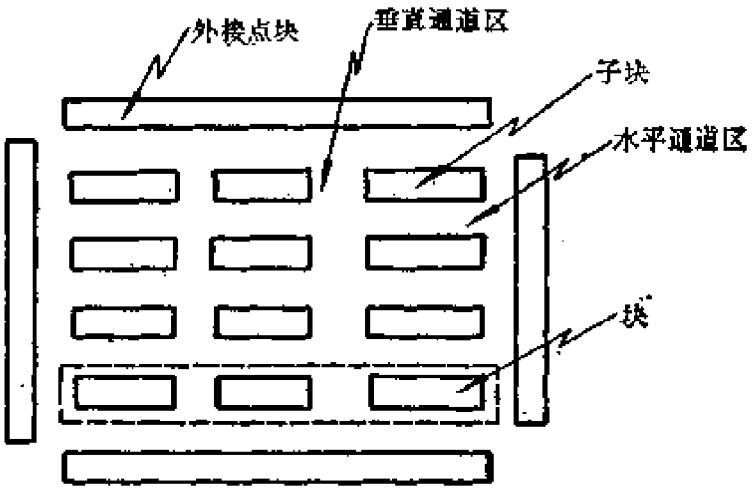


图 4.12　双边单元多元胞模式中块的划分

在双边单元多元胞模式中，成行排列的单元构成块，这些块又被垂直通道区分割为子块(如图 4.12)。

三、块的构造

1. 块的数量的确定

在多元胞模式的初始布局中，首先需要确定芯片上的块的数

量。

设计实践表明，设计得到的芯片面积与单元面积总和之间存在下述关系：

$$S = 2\psi \sum_{i=1}^{n} S_i。$$

其中 S_i 为第 i 个单元的面积，n 为单元总数，ψ 是与电路互连复杂性、单元类型有关的一个参量。一般情况下认为 ψ 等于1。因此当期望设计结果芯片的形状趋于正方时：单边单元模型的块数

$$N_B = \frac{\sqrt{2}}{4} \times \left(\sum_{i=1}^{n} L_i \Big/ \sqrt{\sum_{i=1}^{n} S_i} \right),$$

双边单元模型的块数

$$N_B = \frac{\sqrt{2}}{2} \times \left(\sum_{i=1}^{n} L_i \Big/ \sqrt{\sum_{i=1}^{n} S_i} \right)$$

其中 L_i 为第 i 个单元 x 方向的长度，($\psi = 1$ 时)。

2. 块的构造方法

确定每个单元应安置到哪个块中的设计目标是使各块中单元间有较紧密的联结关系，而使块之间只有尽可能少的联结，实际上这也就是电路的分划过程。为了使设计得到的芯片面积尽量小，还应使各块的长度大致相同，块的平均长度为：

$$\overline{BL} = \sqrt{2\psi \sum_{i=1}^{n} S_i}。$$

通常块的构造可采用下述二种方法。

(1) 逐一构造法。首先从当前未安置单元集中任取一个单元，或选择其中联结度最大的一对单元放入块 B_k 中，应用成群展开法（或其它方法，如"内—外"联结度法）从未安置单元集中选择与 B_k 中各单元联结度和最大的单元置入 B_k 中，直至 B_k 的单元长度和与 $\overline{BL}$ 大致相同，再进行 B_{k+1} 块的构造。这种构造方法实际上每次是将未安置单元划分为二个子集，并使二个子集间的联结度尽可能小，并且使其中一个子集所含单元总长大致与 $\overline{BL}$ 相同。这样构造过程一直进行到余下的未安置单元总长与 $\overline{BL}$ 基本相等为止，

当余下的未安置单元总长偏离 $\overline{BL}$ 较大时，则需调整预计的 $\overline{BL}$ 值重新构造各块。

当基本确定了各块的组成后，可采用行式布局确定芯片上各块的排列顺序.这样也就确定了芯片上各块的相对 y 坐标。

SEMINS—AVESTA 系统 [38] 在块的构造中采用的就是经过修改的逐一构造法，它在应用成群展开法从未安置单元集中选择待安置单元时，考虑到单元大小的不同以及多点线网的情况，采用了下述函数作为选择函数：

$$f(x)=\frac{\sum_{y\in A'} P_C(x,\ y)}{\sum_{s_k\in S(x)} \rho_k}$$

在实际选择时，将使较小的单元(接点较少的单元)容易先被选来进行安置。

逐一构造法简捷而快速，为了获得较好的块的划分结果，则需进行迭代来改善构造得到的结果。

(2) 并行构造法。与逐一构造法不同，并行构造法在通过计算或人工指定块的个数后，对各块同时进行构造，直至所有单元都被分配到块中去为止。

在并行构造法中，通常由设计者确定各块的初始单元或单元集，这样虽然将使设计结果在一定程度上依赖于人的设计经验，但另一方面也使设计者可通过初始单元集的选择来控制设计进程和调整设计结果。然后应用"内—外"联结度法从未安置单元中选择当前具有最大 IOC 值的单元进行安置。当候选单元不唯一时，选择其中单元长度最长的优先进行安置，在安置时，计算该单元放入哪一块内，它与已安置单元的相关连线总长最短。当候选的位置不唯一时，选择其中已安置单元总长较短的块作为该单元的安置位置，也即选择当前容量最大的块作安置位置，直至所有的单元都依上法安置完毕止。

通常为获得较好的块的划分结果，需进行迭代来改善初始构造得到的结果。

贝尔实验室 LTX 系统[61] 多元胞模式块的构造采用的是上述并行构造法。从上述块的构造过程可以看到，实际上该过程是一个在一定限制条件下(群的大小、群的安置位置、初始单元集的确定)的结群过程。通过块的构造(一般还需进行迭代改善)，芯片上单元的相对 y 坐标得到了确定，下一步的任务是确定单元在芯片上的相对 x 坐标。

四、子块的划分

在规模比较大的电路布图设计中，由于各块的单元数比较多，因而相应的水平通道区也比较长。根据通道区的布线规律，随着通道区的长度增长，通道区的利用率将下降。为了提高布图密度，可采用"行列式"结构的多元胞模式，从而使各水平通道区的长度不致太长。这样就相应地引入了"子块"的概念，如图4.11、4.12所示。

在行列式结构的多元胞模式中，在确定了块的构成后，需进行子块的划分。子块的数量可根据电路的规模和系统的要求决定，一般各块中子块的数量大致相同。子块的划分原则是使块内各子块的大小(单元长度和)基本相同，同时使各子块内单元间有尽可能紧密的联结关系，而使各子块间有尽可能少的联结关系。对块 B_i，令其所含单元集为 G_i，子块分划时首先从 G_i 中根据上述目标划分出 G_{i1} 子块，使 G_{i1} 与 $G_i - G_{i1}$ 二子集间有尽可能少的联结关系。然后从 $G_i - G_{i1}$ 中划分出 G_{i2}……。这个子块的划分过程非常类似于一个 0－1 问题的求解，它的精确解可用一种直接搜索的隐算法求得[39]。

在有些系统中，也可首先解决各块中单元的排列顺序，然后将各块等分为大小基本相同的若干子块。

五、块内(子块内)单元排列顺序的初始构造

1. 块内单元一维布局的设计目标

在多元胞模式中，单元排列成行，设计得到的芯片面积除与块

的最大长度有关外，还与块的数量及单元行之间的通道区高度有关。为了获得足够高的布图密度，块内单元一维布局（即块内单元排列顺序的确定）的设计目标应是使与其相关的水平通道区的高度尽可能小。在初始构造布局中，由于单元在块内的实际位置尚未确定（或尚未完全确定），因此计算单元间互连所需的通道区高度实际上是不可能的。从人们的实际设计经验可知，单元间互连线在水平方向的总长度和完成布线所需通道区高度之间存在着一定的联系，因此在实际的初始构造算法中，往往采用单元互连线水平方向总长度最小化作为块内一维布局的设计目标。

2．块内单元安置顺序的确定

块内一维布局也可象块的划分一样，采用对联结法、成群展开法和“内—外”联结度法进行布局构造。在并行构造法进行块的构造的基础上，还可采用“目标坐标顺序法”(target coordinate)。

定义：所谓线网中点是指那样的点，如果设线网 S_k 中接点的 x 和 y 坐标的最小值分别为 x_{min} 和 y_{min}，其最大值分别为 x_{max} 和 y_{max}，则称点

$$\left(\frac{1}{2}(x_{max}+x_{min}),\ \frac{1}{2}(y_{max}+y_{min})\right)$$

为 S_k 的线网中点。

在实际的初始布局中，接点的精确位置往往是难以计算的，因此一般用接点所在单元的中心点坐标作为它的近似值。

定义：所谓单元的目标坐标 TC_i，如果令 $S(i)$ 为含单元 i 的线网集合，其元素数为 m_i，$S_k \in S(i)$，S_k 的中点 x 坐标为 MPX_k。则单元 i 的目标坐标 TC_i 为

$$TC_i = \frac{\sum_{k=1}^{m_i} MPX_k}{m_i}$$

在求出每个单元的目标坐标后，在每一块中可按目标坐标的大小从小到大排队，并按此顺序把块内的单元从左到右排列，完成块内单元的一维布局[61]。

上述初始构造方法处理效率较高，因此可提供一个较好的初始布局结果。

六、单元安置方位的选择

单元在一个具体位置上往往还存在着一个“方位”的选择问题，图 4.13 给出了一个单元的四个不同的位置“方位”。单元的方位选择在块的构造和块内一维布局时进行，一般选择其相关连线长度最短的方位进行安置。

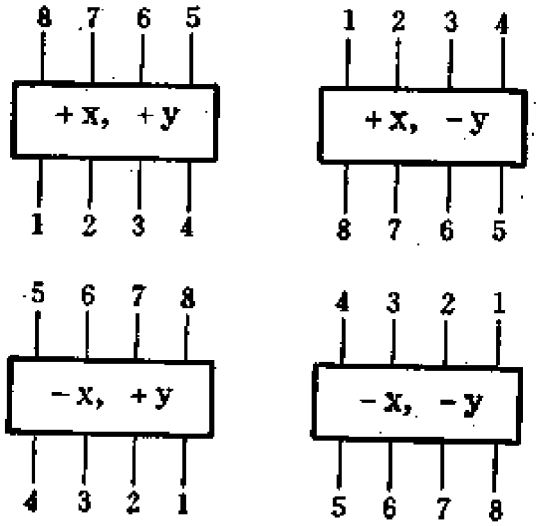


图 4.13 单元的方位

4.7 门阵列的二维初始布局

在门阵列模式的初始布局中，由于它的单元模型具有高度的规则性，因此可方便地应用对联结法，成群展开法，内外联结度法来进行初始构造布局。

实验和设计经验表明，一个好的初始布局在经过迭代改善后，可望得到一个更理想的解，并且可以使迭代过程的处理效率得到提高。本节介绍一种基于二分图最大匹配算法的门阵列二维初始布局算法[40]、[41]、[42]。

一、二维初始布局的图模型

设母片上排列 $P\times q$ 的单元矩阵，我们可以把全部单元定义为 P 个垂直块，每个垂直块含 q 个单元和 q 个水平块，每个水平块含 P 个单元，从另一个角度来讲，每个单元属于一个水平块和一个垂直块。我们分别用 b^V 和 b^H 表示所有垂直块和水平块的集合。即：

$$b^H=\{b_1^H,\ b_2^H\cdots b_q^H\}$$

$$b^V=\{b_1^V,\ b_2^V\cdots b_p^V\}$$

并可相应地定义一个含 q 个顶点的 H 图和一个含 P 个顶点的 V 图。图中每个顶点分别对应着一个水平块或一个垂直块。如果一条线网的元素(单元)分别属于 b_i^H 和 b_k^H 块，则在对应 b_i^H 和 b_k^H 的顶点间连接一条无向边。如果一条线网的元素分属 r 个水平块($r>2$)，则在对应的 r 个顶点存在着 $r-1$ 条边，形成一个对应的链接树(linear tree)。在 V 图中边的连接原则也是相同的。这样，二维初始布局的目标可定义为求得一个单元集的安置方案，使它们在对应的 H 图和 V 图中的总边数 $E=E^H+E^V$ 最少化。

这个问题很类似于图的划分问题 [43]、[44]。它是一个 NP 完备问题[45]。因此在求解上述问题时，将采用一些有效的启发式算法。

二、二维初始布局的基本思想

首先可应用内外联结度法(或成群展开法、对联结法)得到一个单元的安置结果，即把每个单元分配到一个水平块和一个垂直块中去。由 4.2、4.3、4.4 节可知，初始单元集的选择将对结果有较大的影响；此外在这些方法中由于在安置中仅考虑了与已布单元的联结关系，所以将影响结果的精度。因而需要进一步地改善上述结果。

1. 块划分结果的改进

我们来考虑一下 H 图和 V 图中边数与块中单元组成的关系，

可以发现，当我们把属于一个垂直块 b_i^V 的单元重新分配到不同的水平块中去时，由于 b_i^V 的组成并没有改变，因此 V 图中的边数 E^V 不变，但由于 H 图各水平块的组成发生了变化，而使 H 图的边数 E^H 可能发生变化。因此，如果我们找到一个 b_i^V 中单元重新分配的方案，可使 E^H 下降，则由于 E^V 不变，布局的结果将得到改善。设 $b_i^V=\{C_1, C_2, \cdots C_q\}$，其中 C_i 是 b_i^H 的一个元素，则可构成一个二分图。左边的顶点为 $\{C_1, C_2, C_3, \cdots C_q\}$，右边的顶点为 $\{b_1^H-C_1, b_2^H-C_2, \cdots b_q^H-C_q\}$。左顶点与右顶点间的边权为 C_i 与 $b_j^H-C_j$ 互连的边数，即 C_i 与 $b_j^H-C_j$ 之间的公共线网数(图 4.14)，因此上述问题就归结为一个二分图的最大匹配问题。与此同时，我们可以建立一个 $q\times q$ 的价格矩阵 W，价格矩阵的行对应着 b_i^V 中的每一个单元，列则对应着水平块和相应单元的差集 $b_j^H-C_j$。矩阵元素 $W(i, j)$表示 C_i 和 $b_j^H-C_i$ 间的联结度。应用最大匹配算法可求得一个 b_i^V 中单元的重新分配，使 E^V 不变，而使 E^H 下降最多。

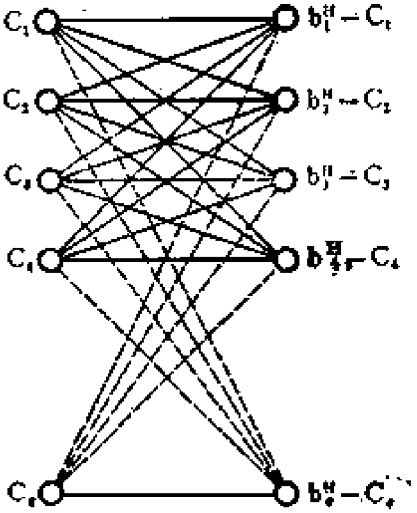


图 4.14 C_i 与 $b_j^H-C_j$ 之间的公共线网

同理，这个处理过程也可用于 b_i^H 中单元的重新分配。实际处理时，可按 b_1^V、b_1^H，b_2^V、b_2^H，…… 的顺序交替地进行块内单元的重新分配，直至 $E=E^V+E^H$ 不再下降为止。

图 4.15 为通过应用“内—外”联结度法等得到的 8 个单元及 11 条线网的预处理安置结果。其中 $E^H=11, E^V=4$。其联结矩阵为 $A=\{a_{ij}\}$(偶图描述)。

设该母片上单元排列为 2 行 4 列。预处理结果为：

$$b_1^H=\{1, 8\}$$

$$b_2^H=\{3, 6\}$$

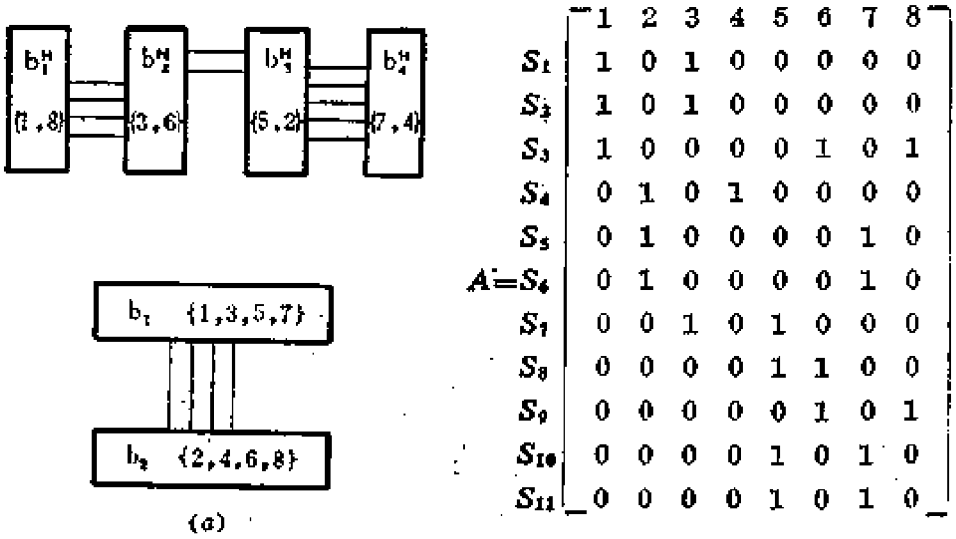


图 4.15 "内—外"联结度法的预处理安置结果

$$b_3^H = \{5, 2\}$$

$$b_4^H = \{7, 4\}$$

$$b_1^V = \{1, 3, 5, 7\}$$

$$b_2^V = \{2, 4, 6, 8\}$$

然后我们对每个垂直块中的单元进行重新分配，则 $b_1^V = \{1、3、5、7\}$，$b_1^H - C_1 = \{8\}$，$b_2^H - C_2 = \{6\}$，$b_3^H - C_3 = \{2\}$，$b_4^H - C_4 = \{4\}$，其价格矩阵W为：

$$W = \begin{array}{c} \\ 1 \\ 3 \\ 5 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccc} \{8\} & \{6\} & \{2\} & \{4\} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \end{bmatrix} \end{array}$$

其联结图如图 4.16 所示。应用最大匹配算法对第一行(垂直块 b_1^V)的单元重新分配，我们可得 W' 为：

$$W' = \begin{array}{c} \\ 1 \\ 5 \\ 7 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} \{8\} & \{6\} & \{2\} & \{4\} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

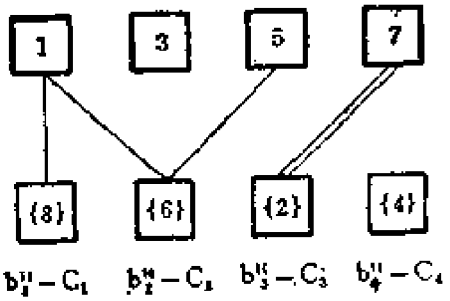


图 4.16 价格矩阵的联结图

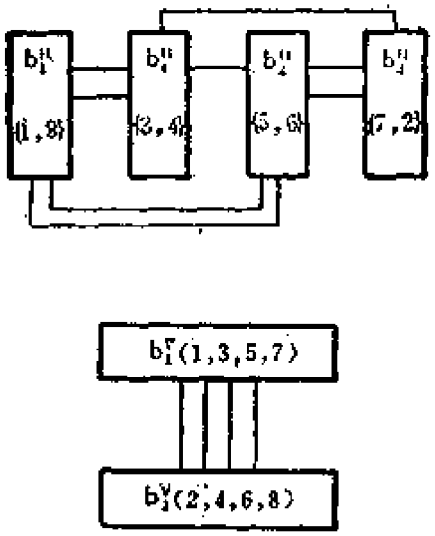


图 4.17 初始布局的块划分结果

对应的水平块组成为 $b_1^H = \{1, 8\}, b_2^H = \{5,6\}, b_3^H = \{7, 2\}, b_4^H = \{3, 4\}$。在 E^V 不变的情况下，使 E^H 减少了 3。当继续对其它水平块和垂直块进行处理时，发现这已经是最少的 E 值了，因此初始布局结束。上述结果 $b_1^V = \{1, 3, 5, 7\}$, $b_2^V = \{2, 4, 6, 8\}$, $b_1^H = \{1, 8\}$, $b_2^H = \{5, 6\}, b_3^H = \{7, 2\}, b_4^H = \{3, 4\}$ 为初始布局的块划分结果（图 4.17）。

2. 块的安置

块的划分完成后，需要进一步决定块在芯片上的安置，使连线的总长度最小化。

水平块的安置方法和垂直块的安置方法是相同的，因此我们只需讨论水平块的安置就可以了。

首先我们介绍一种连线总长的计算方法，这种方法计算得到的连线总长实际上是完成实际布线后，连线总长的一个下限界。

令水平块以某一顺序在芯片上排列，可记作 $O^h = (b_1^H, b_2^H \cdots b_q^H)$，设 $S^H(K^H) \subset S$ 是与水平块一个有序子集 $K^H = \{b_1^H, b_2^H \cdots b_k^H\}$ 相连的线网子集，则 $S^H(b^H - k^H)$ 是与余下的水平块相连的线网子集。

定义：$\rho(b_1^H, b_2^H \cdots b_k^H)$ 为 k^H 和 $b^H - k^H$ 二水平块子集间的

互连密度：

$$\rho(b_1^H, b_2^H \cdots b_q^H) = |S^H(k^H) \cap S^H(b^H - k^H)|$$

则该安置方式的水平连线总长 $L^H(O^H)$ 为：

$$L^H(O^H) = \sum_{i=1}^{q-1} \rho(b_1^H, b_2^H \cdots b_i^H)$$

水平块安置位置的决定就是使其对应着具有 $\min\{L_i^H(O_i^H)\}$ 的排列顺序。这是一个典型的行式布局问题(可参看第五章)。

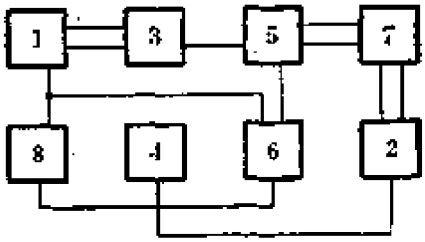


图 4.18 行式布局的初始布局结果

图 4.18 为前例经过行式布局的初始布局结果。$O^h = (b_1^H, b_4^H, b_2^H, b_3^H)$, $O^V = (b_1^V, b_2^V)$

从一些实验结果来看，该方法可获得比构造初始布局更好一些的结果($E^H + E^V$ 可减少 5%～13%)，而计算时间约为构造初始布局的几倍。

本节所述二分图最大匹配算法也即分配算法，为布图设计中一个基本算法。本章附录中介绍了几种具体的实用算法。

4.8 边生长的等分接点法

一、边生长法的主要思想

在初始布局构造中，初始单元(或单元集)的选择往往对初始布局的结果有较大的影响，但它们的选择往往带有一定的随机性。另一方面，在布图设计中由于标准化和系统化的需要，或分级设计时对模块设计的需要，对模块设计结果的引出接点位置或顺序有一定的要求。当采用"核心生长法"构造初始布局时，从设计逻辑来讲，引出接点的合理位置应取决于芯片"内部"单元的安置结果。这样往往就难于满足设计中对引出接点位置和顺序的要求，或为了满足这些要求而使相应的布局、布线变得不合理，这是核心生长

法的一个缺点。此时可采用边生长法。所谓边生长法是指把引出接点定义为“单元”，而且把这些引出接点单元（或一部份引出接点单元）作为构造的初始单元。然后运用对联结法、成群展开法或“内—外”联结度法进行未安置单元的选择和安置，犹如从芯片的边上（一条边或几条边）开始生长，随着单元的逐步安置，向芯片中央扩展，直至全部单元安置完毕。

边生长法可应用于多元胞模式和门阵列模式的初始布局构造。这种方式还将有助于改善芯片上各区域布线密度的均匀性，提高可布性。GAELIC—COMPEDA 系统[46]就是采用单边生长法来进行任意元的布局构造。

二、等分接点法的主要思想

在构造布局中另一个问题是如何决定选定单元的安置位置。前几节介绍的方法基本思想是类似的，即把选定单元安置在与已安置单元（或单元集）相关连线总长最短的位置上。这种方法的缺点在于仅考虑了选定单元与已安置单元的位置关系，而没有考虑与未安置单元之间的关系。

1．等分接点法的定义

若在 x 轴上有序地排列着 m 个定点，其坐标为 $x_1, x_2, \cdots x_i, x_{i+1}, \cdots x_m$。现有一动点 x，欲使 $\sum_{i=1}^{m} \overline{xx_i}$ 最小，则 x 动点的最佳位置（或区间）应为：

(1) x_k 处；其中 $k=\frac{m+1}{2}$（当 m 为奇数时）。

(2) $[x_k, x_{k+1}]$ 中；其中 $k=\frac{m}{2}$（当 m 为偶数时）。

2．等分接点法动点最佳位置的证明

(1) 当 m 为奇数情况时：

用反证法：设动点 x 在 x' 处，$\sum_{i=1}^{m} \overline{x'x_i}$ 最小。x' 在 x_k 点右侧距

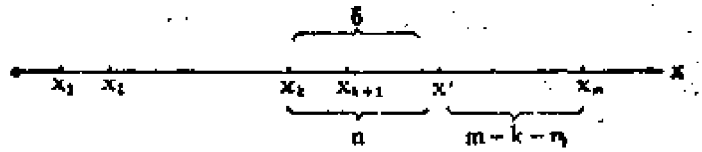


图 4.19　m 为奇数情况时证明示意图

离x_k为 δ 处，如图 4.19 所示，(x_k, x')间有 n 个点，$\frac{m+1}{2} \geqslant n \geqslant 0$。

令动点 x 原在 x_k 处，当 x 由 x_k 处移动到 x' 处时，则$[x_1, x_k]$区间内各点与动点 x 的距离 $\sum_{i=1}^{k} \overline{x'x_i} = \sum_{i=1}^{k} \overline{x_k x_i} + k \times \delta$。$(x', x_m]$区间内各点与动点 x 的距离 $\sum_{i=k+n+1}^{m} \overline{x'x_i} = \sum_{i=k+n+1}^{m} \overline{x_k x_i} - (m-k-n) \times \delta$。$(x_k, x']$区间内各点与动点 x 的距离最多只能减少 $n \times \delta$(当 n 个点重迭于 x' 处时的特例)，即 $\sum_{i=k+1}^{k+n} \overline{x'x_i} \geqslant \sum_{i=k+1}^{k+n} \overline{x_k x_i} - n \times \delta$。因此：

$$\sum_{i=1}^{m} \overline{x'x_i} \geqslant \sum_{i=1}^{m} \overline{x_k x_i} + k \times \delta - (m-k-n) \times \delta - n \times \delta$$

$$= \sum_{i=1}^{m} \overline{x_k x_i} + 2k \times \delta - m \times \delta$$

$$\sum_{i=1}^{m} \overline{x'x_i} - \sum_{i=1}^{m} \overline{x_k x_i} \geqslant (m+1) \times \delta - m \times \delta = \delta$$

因此动点在 x' 时 $\sum_{i=1}^{m} \overline{x'x_i}$ 不是最小。同理可证 x' 在 x_k 左侧时的情况。所以当 m 为奇数时，动点在 $x_k\left(k = \frac{m+1}{2}\right)$处 $\sum_{i=1}^{m} \overline{xx_i}$ 最小。

(2) 当 m 为偶数情况时：

① “在 $[x_k, x_{k+1}]$ 区间内各点上 $\sum_{i=1}^{m} \overline{xx_i}$ 都是等值的”。

见图 4.20，因 m 为偶数，$k = \frac{m}{2}$，当动点由 x_k 处移向(x_k, x_{k+1})

中任一点 x 时，其左侧定点数和右侧定点数相同（为 k 点），左侧定点与动点距离增加 $k\times\delta'$，则右侧定点与动点距离减少 $k\times\delta'$，因此 $\sum_{i=1}^{m}\overline{x_kx_i}=\sum_{i=1}^{m}\overline{xx_i}$。同理，当动点由 x_{k+1} 处移向 $(x_k,\ x_{k+1})$ 中任一点 x 时，$\sum_{i=1}^{m}\overline{x_{k+1}x_i}=\sum_{i=1}^{m}\overline{xx_i}$。因此，在 $[x_k,x_{k+1}]$ 区间内各点上 $\sum_{i=1}^{m}\overline{xx_i}$ 都是等值的。

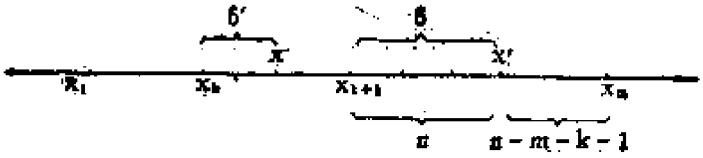


图 4.20　m 为偶数情况时证明示意图

② “在 $[x_k、x_{k+1}]$ 内各点 $\sum_{i=1}^{m}\overline{xx_i}$ 最小”。

用反证法：设动点在 x' 处时，$\sum_{i=1}^{m}\overline{x'x_i}$ 最小，x' 在 x_{k+1} 右侧距离为 δ 处。$(x_{k+1},\ x']$ 间有 n 个定点。$m-k-1\geqslant n\geqslant 0$。类似①的证明过程，可得：

$$\sum_{i=1}^{m}\overline{x'x_i}-\sum_{i=1}^{m}\overline{xx_i}\geqslant(k+1)\times\delta-(m-n-k-1)\times\delta-n\times\delta$$
$$=2(k+1)\times\delta-m\times\delta=2\delta$$

因此，动点在 x' 处时，$\sum_{i=1}^{m}\overline{x'x_i}$ 不是最小。同理可证 x' 在 x_k 左侧的情况。所以当 m 为偶数时，动点在 $[x_k,\ x_{k+1}]$ 区间内 $\sum_{i=1}^{m}\overline{xx_i}$ 最小。

由等分接点法可知，当已知 x 轴上定点总数和位置 x 对应的 $(0,\ x]$ 区间内定点的分布情况，即可判定位置 x 对动点来讲是否是 $\sum_{i=1}^{m}\overline{xx_i}$ 最小位置，而与 $(x,\ x_m)$ 区间内定点位置无关。

三、边生长等分接点法

1. 边生长等分接点法的单元选择函数

当把等分接点法用于边生长(如芯片的上边)的一维初始构造布局时，对未安置单元 x，设 $S(x)$元素数为 m，各线网权重为 1。

计算各未安置单元的"内—外"关联度 IOC_x。在边生长法中，IOC_x 值意味着：当单元 x 安置在当前位置上时，单元 x 与已安置单元相连的线网数与它和未安置单元相连的线网数之差。边生长等分接点法的选择函数 $f_4(x)$可为：

$$f_4(x) = P \times IOC_x + m \quad P \gg m$$

上述复合选择函数中主项的权重 P和辅项 m的作用是，当具有最大的 IOC 值的候选单元不唯一时，选择接点多的（单元尺寸大的）单元先进行安置。在边生长法中，当从 B' 中选择具有最大的 $f_4(x)$ 值的单元首先进行安置时，一方面从联结关系来看，意味着所选单元与已安置单元的联结关系强于（相对于其他待选单元而言）与未安置单元的联结关系。另一方面从当前安置位置来看，意味着所选单元可望得到一个偏离其最佳位置相对较小的安置位置，即在当前各候选单元中，将该单元安置在当前位置，可望使由于偏离各单元自身最佳位置所引起的连线总长的增加量最小化。因此，$f_4(x)$选择函数与边生长法相结合，从联结关系来看，具有与内外联结度相同的优点。从边生长的安置规则来看，它使安置规则和选择规则一致化，可望使单元的安置更加合理。

2. 边生长等分接点法的特点

边生长等分接点法在布局构造时还可方便地给出一些重要的布局结果评价参数。

在多元胞模式中，各单元行的实际长度 BL_i 的均匀性将影响布图密度，$\max\{BL_i\}$将决定芯片所需的实际长度 L。这些都与布局时各单元行实际长度 BL_i 的计算有关。我们知道：

$$BL_i = \sum_{j=1}^{m_i} L_{ij} + N_i \times W$$

其中：L_{ij} 为 i 行中单元 j 的长度；

m_i 为 i 行中单元的总数；

N_i 为穿过 i 行的线网总数；

W 为一条布线通道的宽度。

显然，这里重要的是 N_i 的计算。

在门阵列模式中，布局能满足布线要求的必要条件是使每一个单元行都满足：

$$N_i \leqslant \left(L - \sum_{j=1}^{m_i} L_{ij} \right) / W$$

显然，N_i 的计算对布局的可行性评价也是很重要的。

边生长等分接点法可以方便地在每一行构造完成时，精确地求得 N_i 值。这将有利于及时地对部分布局进行适当的调整和对整个构造过程的控制。

我们以双边单元的上边生长为例。为了描述方便，令单元的接点集为 $T = \{t_1, t_2, \cdots t_m\}$

T 集中与 A' 元素相连的接点子集为 T_A 集

T 集中与 B' 元素相连的接点子集为 T_B 集

T 集中属于单元上边的接点子集为 U 集

T 集中属于单元下边的接点子集为 D 集

并记 $|S|$ 为 S 集的元素数。

在每次选定单元安置时，都根据单元方位选择函数 $f_5(x)$ 进行单元安置方位的选择。

$$f_5(x) = (|U \cap (T_A - T_B)| - |U \cap (T_B - T_A)|) + (|D \cap (T_B - T_A)| - |D \cap (T_A - T_B)|)$$

当 $f_5(x)$ 大于等于零时，单元取 $+y$ 方位，否则单元取 $-y$ 方位（参见图 4.13）。已安置单元的方位确定后，各单元行的 N_i 值的计算将是十分方便的。设由上至下对各单元行进行编号，$i-1$ 行将表示其位置高于 i 行的上一个单元行。各线网可用其接点集来定义，$S_q = \{t_{\rho p}\}$，$t_{\rho p}$ 表示 ρ 单元的 p 接点。则穿越第 i 行的线网 S_q 必满足下述条件：

$$S_q \cap \left(\sum_{k=1}^{i-1} \sum_{j=1}^{m_k} T_{kj} + \sum_{j=1}^{m_i} U_{ij} \right) \neq 0$$

且 $$S_q \cap \left(\sum_{k=1}^{i-1} \sum_{j=1}^{m_k} T_{kj} + \sum_{j=1}^{m_i} U_{ij} \right) \neq S_q$$

其中：T_{kj} 为 k 单元行中 j 单元的接点集；

U_{ij} 为 i 单元行中 j 单元上边接点集；

m_k 为 k 单元行中单元总数。

因此，穿越第 i 行的线网总数 $N_i = |\{S_q\}|$。并由此可方便地计算得到各单元行的实际长度 BL_i。从而在初始构造中，可更精确地控制各单元行的构造结果，以获得更为满意的初始布局结果。

在实际应用时，也可在一维构造中采用对边同时生长的方法。如在多元胞的块构造中采用上、下边同时生长，而在块内单元位置的构造中采用左、右边同时生长，并行构造的方法。引文[47]介绍了一种单边生长的等分接点法实际布局算法。

4.9 初始布局方法评价

一、初始布局过程的必要性

我们在本章内介绍了一些初始布局方法，对于初始布局的方法及其必要性在从事布图设计自动化的科技工作者之中存在着不同的看法。相信随机初始布局的人们认为，在迭代改善布局时将对布局的结果进行如此多的调整和修改，以致于任何初始布局在开始时实际上都可认为是好的。另一种观点认为，要评价一个初始布局结果是否是一个"好"的解，实际上是很困难的。因此，为了避免在迭代改善过程中失去找到一个更好的解的机会，至少应有几个初始布局的结果可被用来进行探索。目前绝大多数实际系统的设计者认为初始布局过程还是有必要的，对提高整个布局设计的精度和处理效率都是有利的[48]。文献[11]对 4 个设计问题分别经过构造初始布局再进行邻接交换迭代改善求解，和用十个随机初始布局再进行成对交换迭代改善取得的结果进行了对照比较（见表 4.1）。结果表明，初始构造布局后可在更短的计算时间内

表 4.1 初始构造布局和随机初始布局经改善后的结果对照表

问题	项目	构造+邻接交换	随机+成对交换	改进的比率
A	试布次数	1	10	
	连线总长(平均值)	485	535	9%
	平均机时/每次	186	315	41%
	连线总长分布范围	485	515—565	6%—14%
	求得最佳解总时间	186	3150	94%
B	试布次数	1	10	
	连线总长(平均值)	463	524	11%
	平均机时/每次	180	295	39%
	连线总长分布范围	463	480—575	3%—19%
	求得最佳解总时间	180	2950	94%
C	试布次数	1	10	
	连线总长(平均值)	325	343	7%
	平均机时/每次	120	200	40%
	连线总长分布范围	325	315—363	−3%—12%
	求得最佳解总时间	120	2000	94%
D	试布次数	1	10	
	连线总长(平均值)	262	273	4%
	平均机时/每次	130	142	7%
	连线总长分布范围	262	240—287	−9%—10%
	求得最佳解总时间	130	1420	90%

注：采用的机器为 IBM System/360 Mode/85，机时计算单位为秒。初始构造过程机时约为 4～6 秒。

经过迭代改善得到更好的布局结果(平均结果)。

文献[42]在门阵列的二维初始布局中，给出一个含 151 个单元(其中 15 个引出接点单元) 419 条线网问题的布局实验结果，并

与五个随机初始安置后经二分图最大匹配算法优化的结果进行了比较，结果表明，初始构造比随机安置可以更短的时间经过优化得到更好的结果(见表 4.2)。

表 4.2　151 个单元的布局实验结果

	方法	初始安置时的边数			优化后结果的边数			计算时间
		$E=E^H+E^V$	E^H	E^V	$E=E^H+E^V$	E^H	E^V	(秒)
1	随机	898	520	378	764	387	377	18.0
2	随机	902	524	378	769	413	356	17.7
3	随机	912	518	394	775	398	377	16.9
4	随机	886	501	383	741	383	358	19.3
5	随机	931	528	403	793	403	395	17.8
	五次平均	906	518	388	769	397	372	17.9
6	初始构造	765	371	394	701	310	391	1.8+9.3

注：采用的机器为 CDC—6400。

这些实验结果表明，在大部分的实际设计中，初始布局过程是必要的。

文献[36]通过对同一问题 25 个不同的初始布局结果和它们

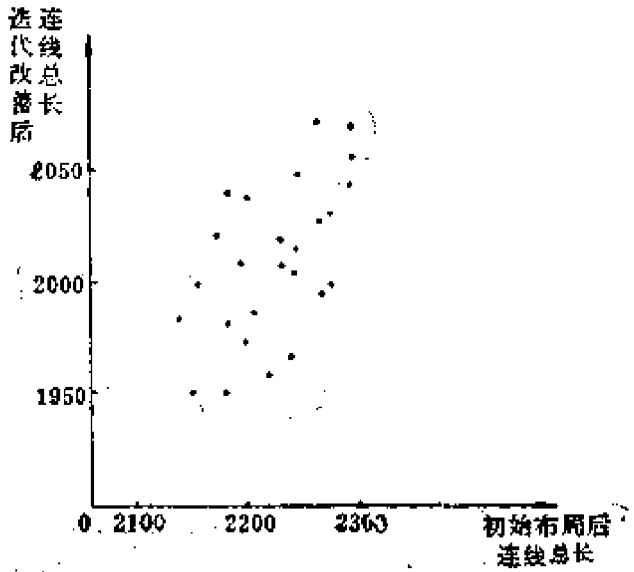


图 4.21　初始布局和通过迭代改善后的结果分析

通过迭代改善后得到的结果的关系进行了分析（图 4.21）。分析结果表明：当用某些参数(如连线总长)来评价初始布局的质量时。一般地讲，一个“好”的初始布局将可望经过迭代改善后得到一个较好的布局结果。但是由于评价参数往往很难真正地对布局的结果给出严格的评价以及迭代改善方法所具有的一些限制，一个较“好”的初始布局结果将不一定总对应着一个更好的布局结果。因此，从若干初始布局中选择一个最“好”的去进行迭代改善并不能保证获得一个可能的最佳解。在规模比较大、要求比较高的问题中，选择若干较好的初始布局分别进行迭代改善，然后从中选择一个最佳的结果作为实际解是提高布局精度的一个有效方法[36]。自然，这种方法的代价是要付出多几倍的机时。

二、初始布局方法的评价

在评价一种初始布局方法时，它的处理效率往往是容易评价的，难于评价的是它的解的质量。由于目前的迭代改善布局算法往往对于结群性弱的单元的位置调整比较有效，换句话讲，对于一个有强的结群性的单元集，一旦在初始布局时被安置在某些不合理位置，则在迭代改善时，它们的位置调整将比较困难。因此，结群性弱的单元在初始布局安置位置不合理时，解的质量影响较小，而结群性强的单元集安置位置不合理将严重影响解的质量。此外，布局质量的评价中很重要的一点是，该结果是否能很好地满足布线的需要。要精确地评价这一点也是很困难的，一般认为，在很多问题中，若能使芯片上各区域布线密度均匀化，即能较好地满足布线的需要。值得注意的是这个期望目标在不少情况下与连线总长最短化的目标是相悖的。

当我们把芯片划分成若干相同的区域 L，如分成 25 个矩形区域，见图 4.22，当采用最短曼哈坦路径实现互连时，对于区域间的连线（不含本区与其它区域间的连线），芯片上各区域的布线可利用率是不同的。设芯片上任二区域内有相同的连线数，则将有

F	E	D	E	F
E	C	B	C	E
D	B	A	B	D
E	C	B	C	E
F	E	D	E	F

图 4.22　把芯片划分成 25 个矩形区域

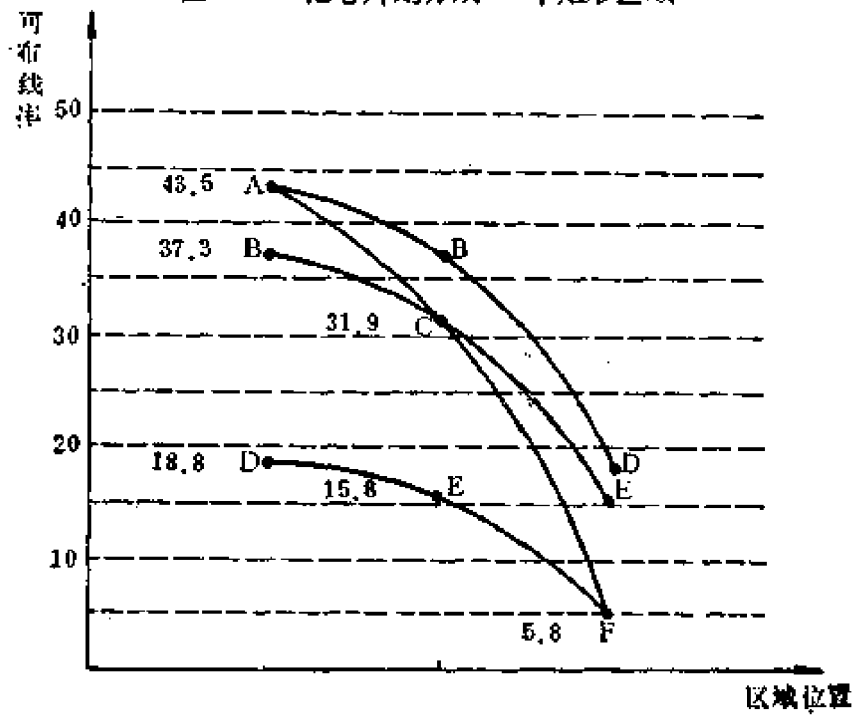


图 4.23　各域布线可利用率与位置变化的情况

43.5% 的两域间的连线可利用 A 域。而仅有 5.8% 的两域间连线可利用 F 域。图 4.23 给出了各域布线可利用率随其位置变化的情况。由此可见，为了使布局结果能较好地满足布线的要求，即使实际的各域布线密度均匀化，在区域其它条件（如面积、域内单元密度等）相同时，应使 A 区域内（包括 A 区与其它区的连线）的连线数（或所需的布线区面积）低于 F 区域内的连线数，即应使各区域内的连线密度分布与其布线可利用率形成某种反递增关系。考虑到一般迭代改善方法的限制，在初始布局时，应使构造方法较好地适应上述要求，以获得一个更好地满足布线要求的布

局结果。

此外，初始布局时还应考虑到设计中的其它要求，如外接点位置、顺序要求、特殊线网的连线长度要求等。

三、各类方法简评

文献[36]认为内外联结度法优于对联结法和成群展开法。从单元联结性关系出发，内外联结度法的优点是明显的。从单元安置位置最佳化出发的边生长等分接点法，除具有内外联结度法相同的优点外，对于安置位置的确定、适应引出接点的要求及适应可布性的要求等方面也有明显的优点。

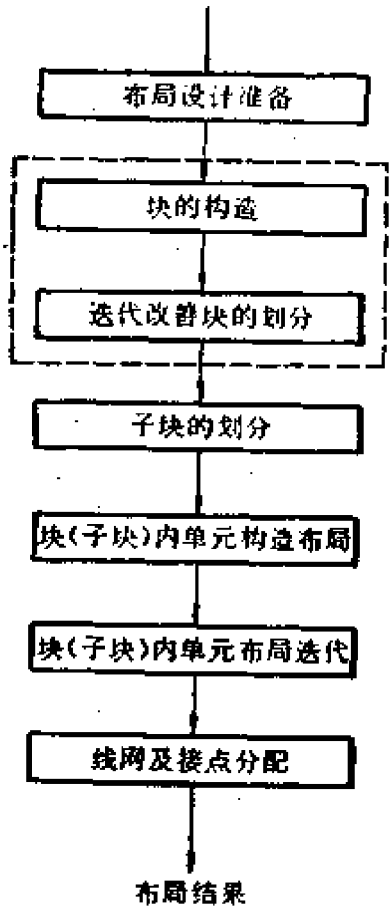


图 4.24 多元胞模式中二维简化为一维的布局方法

在多元胞模式中，把二维的布局问题简化为二个一维的布局问题并分别进行构造和迭代改善的方法（图4.24)是一种行之有效的方法，它可以相当高的效率对规模很大的电路进行布局设计。

通常，块的构造和迭代改善是顺序执行的。这样在迭代改善时，有些不同大小的单元的合理交换可能因单元的尺寸(长度)与块的长度的限制而不能实现。贝尔实验室 LTX 系统在多元胞块的构造过程中穿插进一些可控的迭代改善过程，以克服上述问题，提高最终结果的精度。设未布单元总数为 N，K 为穿插的迭代改善过程次数，在并行地进行块的构造时，每当构造了 N/K 个单元后，即停止构造过程进入迭代改善过程，对当前的构造结果进行优化，然后再继续构造，经过 K 个迭代改善过程后，完成块的

划分。实验表明，一般随着 K 值的增加，将使解的质量（以连线总长来度量）随之得到进一步的改善。当然，与此同时所费机时也将相应地增加。对不同的问题，合理的 K 值的选择具有一定的随机性。

原则上，建立在联结关系基础上的初始构造方法也可应用于任意元模式的初始布局。文献[36]介绍了一种从联结度出发的任意元初始构造布局。首先由电路所含单元的总面积出发估算设计结果芯片的大致面积，并预先确定一个大致的外形（一般为矩形，即确定其长和宽），然后把所有引出的 I/O 接点安置到边界上的适当位置。然后每次选择与已安置引出接点和已安置单元具有最高联结度的单元先进行安置，并使其安置位置尽可能靠近与其相关的已安置接点。如果可能的话，将其装入已安置单元间留下的“洞”中去。在安置中并注意尽可能使已安置单元的轮廓线不大于预定的外形。

这些方法的缺点是在选择、安置过程中没有考虑单元外形的影响，或只是具有一些极其简单的启发性原则（如面积大的单元适当先安置等等），文献［3］认为对于任意元模式的布局，形状的匹配将比联结关系更重要，这将使任意元的布局问题更类似于一个“下料”问题。

从我们上述讨论可以看到，布局方法研究（包括初始布局方法研究）是一个正在发展、还需进行大量的试验和研究的领域。

4.10 本章附录——关于分配问题(二分图最大匹配问题)的算法

一、问题的描述

一个典型的实际问题为：有 n 个人做 n 件工作，已知每人做各件工作的效率(或时间)，求一个最优的分配方案，使总的工作效率最高(或使花费的总时间最少)。

当用一个具有二分划(X，Y)的赋权二分图来描述上述问题时，X 顶点集表示人的集合$\{x_1, x_2, \cdots x_n\}$，Y 顶点集表示工作的集合$\{y_1, y_2, \cdots y_n\}$，边 x_iy_j 的权重为人 x_i 做工作 y_j 时的效率(或时间)，则上述问题即为求二分图最大匹配问题[49]。该问题的条件也可用一个费用矩阵 $W(n, m)$来描述，其中行表示人，列表示工作，矩阵元素 $W(i, j)$ 为人 i 做工作 j 的“费用”(效率或时间)，则该问题即求列标 j 的一个排列 $P^1, P^2, \cdots P^n$ 使

$$Z = \sum_{i=1}^{n} W(i, P^i)$$

最小(称最小费用分配)或最大(称最大费用分配)。

显然最大费用分配问题可用一个简单的方法化为最小费用分配问题。令 $\mathrm{Max}C = \max\limits_{1\le i,j\le n}\{W(i, j)\}$；对原费用矩阵 $W(n, m)$构造 $W^*(n, m)$使

$$W^*(i, j) = \mathrm{Max}C - W(i, j)$$

从 $W^*(n, m)$求出的最小费用分配解即为 $W(n, m)$最大费用分配的解。其最大费用总值 Z 为

$$Z = n \times \mathrm{Max}C - Z^*$$

因此在下文内我们仅讨论最小费用分配。

事实上，分配问题可推广到人数和工作数不等的情况，即费用

矩阵 $W(n, m)$为一长方阵。故分配问题的一般提法为：

当 $n>m$ 时，求行标 i 的一个排列 q^1，$q^2\cdots q^m$ 使 $Z=\sum_{i=1}^{m} W(q^i, j)$最小；

当 $n\leqslant m$ 时，求列标 j 的一个排列 P^1，$P^2\cdots P^n$ 使 $Z=\sum_{i=1}^{n} W(i, P^i)$最小。

二、MUNKRES 算法

MUNKRES 算法[49]为 Munkres 首先提出[47]，是 KUHN [40]算法(匈牙利算法)的变形。原用于方阵情况，由 Bourgeois 和 Lassalle[53]将其推广到长方阵情况。

1. 基本思想

已知矩阵 A，若矩阵 B 满足：

$b(i, j)=a(i, j)-C_i-r_j(i=1, 2, \cdots n, j=1, 2, \cdots m)$

其中 C_i，r_j 为任何实数，则 B 和 A 具有相同的分配问题最优解。记作：

$$TA=B。$$

若找出变换 $T_1, T_2, \cdots T_s$，使 $T_sT_{s-1}\cdots T_1A=A^{(s+1)}$

满足：ⓐ $\forall a_{(i,j)}^{(s+1)}\geqslant 0$

ⓑ $A^{(S+1)}$中不同行，不同列(称独立的)的零元素最大个数 $k=\min\{n, m\}$

则 $A^{(S+1)}$ 中独立的零元素的行(或列)标就是分配问题的解。

2. 算法步骤

(1) 令 $K=\min\{n, m\}$，$L=\max\{n, m\}$，$A^{(0)}=A$

(2) 对矩阵 $A^{(0)}$施行变换 T_0：$T_0A^{(0)}=A^{(1)}$，其元素为：

$$a^{(1)}(i, j)=\begin{cases} a^{(0)}(i, j)-h_i \quad (i=1, 2, \cdots n); \\ h_i=\min\limits_{\forall j}\{a^{(0)}(i, j)\}; \quad \text{当 } m\geqslant n \text{ 时} \\ a^{(0)}(i, j)-h_j \quad (j=1, 2, \cdots m); \\ h_j=\min\limits_{\forall i}\{a^{(0)}(i, j)\}; \quad \text{当 } m<n \text{ 时} \end{cases}$$

(3) 对 $A^{(S)}$中的零元素标以"*"号

逐个检查 $A^{(S)}$中的元素。若元素值为零且同行、同列中没有已经标以"*"的零元素(以下简称"*")时，则对该元素标以"*"，直至 $A^{(S)}$中全部元素都检查完毕为止。

(4) 覆盖行

对 $A^{(S)}$中"*"所在行进行覆盖。若覆盖行数 k 等于 K，则"*"的列(或行)标即为分配问题的解，否则继续处理。

(5) 对 $A^{(S)}$中的零元素标以"△"号

逐个检查$A^{(S)}$中未被覆盖的零元素，并标以"△"号(以下简称"△")。考察它所在的列，若同列中无"*"，则立即转(6)；若有"*"则取消"*"对应的覆盖行，覆盖包含"△"和"*"的列，继续检查。若所有的零元素都已被覆盖，则转(7)。

(6) 产生新的"*"

令 a_0 表示未被覆盖的"△"，a_1 表示与 a_0 ("△")同行的"*"，a_2 表示与 a_1("*")同列的"△"，直至由 a_r 找不到对应的 a_{r+1} 为止(r 必为偶数)，可形成二个序列：

$$\text{"△"}: a_0,\ a_2,\ a_4,\ \cdots a_r$$

$$\text{"*"}: a_1,\ a_3,\ a_5,\ \cdots a_{r-1}$$

取消 $a_1, a_3 \cdots a_{r-1}$ 的"*"号标志，对 $a_0, a_2, a_4, \cdots a_r$ 标以"*"，显然，经上述处理后，"*"数将增加 1 个。取消原有的覆盖(行和列)，转(4)。

(7) 作 $T_5 A^{(S)} = A^{(S+1)}$的变换

令 $R = \{(i, j) \mid a^{(S)}(i, j)$未被覆盖$\}$

$$h^{(S)} = \min_{(i,j)\in R} \{a^{(S)}(i, j)\}$$

$J^{(S)} = \{j \mid j$ 列未被覆盖$\}$

$I^{(S)} = \{i \mid i$ 行被覆盖$\}$

$$\bar{a}^{(S+1)}(i, j) = a^{(S)}(i, j) + h^{(S)} \quad (i \in I^{(S)})$$

$$a^{(S+1)}(i, j) = \bar{a}^{(S+1)}(i, j) - h^{(S)} \quad (j \in J^{(S)})$$

经过变换后，$A^{(S+1)}$保持了$A^{(S)}$中原有的标以"*"的零元素，并产生了新的零元素。然后，保留原有的"*"，"△"和覆盖，转(5)。

三、分支限界法

1. 分支限界法(branch and bound method)[52]的基本思想

将所有的可行解的集合进行分划。对每个分划出的可行解子集来计算其最优解的下界。如果这个下界大于当前最优解（第一个当前最优解可用某种近似算法求得或由分支限界法本身求得），则这个可行解子集就不用继续搜索了。如果这个下界值小于当前最优解，则需将该可行解子集继续分划、搜索。当找到一个可行解优于当前最优解时，把该解作为当前最优解。继续搜索直至所有的可行解都被搜索到或被判定不用搜索为止。该方法得到的解一定是最优解。图 4.25 为一个 4 阶分配问题的判定树，树上的顶点是费用矩阵的一个元素，求解分配问题即为求该树上一条由根到叶的路径 S，使

$$\sum_{(i,j)\in S} W(i,j)\text{最小}$$

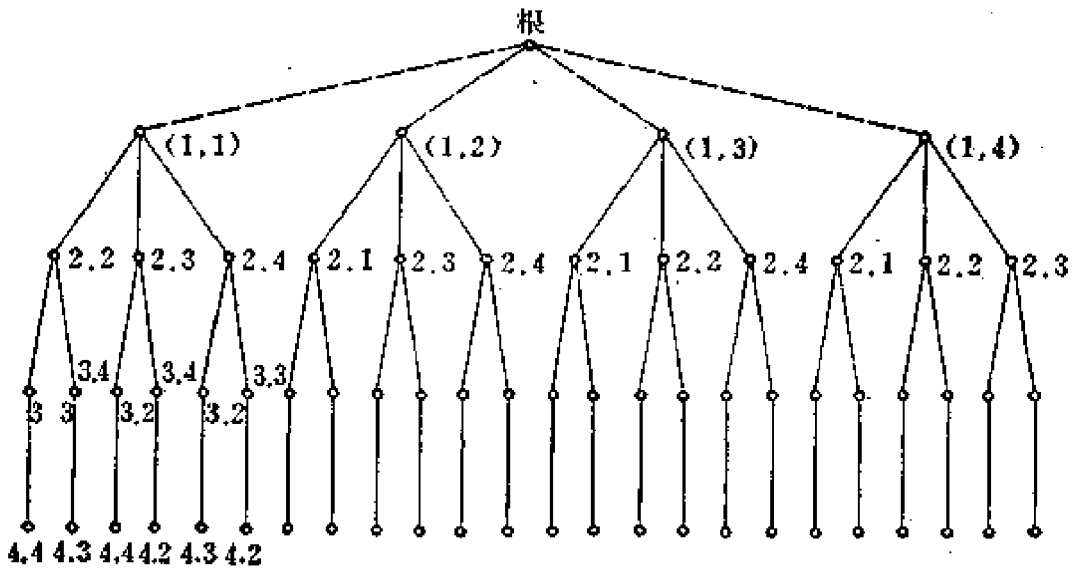


图 4.25　4 阶分配问题的判定树

2. 算法步骤

令 ρ 为树上顶点到根的最短距离（边权为 1）。ρ 称作顶点的级，U 为当前最优解值，S 为结果顶点集，S^* 为局部解顶点集，I、J 分别为 S^* 元素在费用矩阵中行号和列号的集合。令当前被检测顶点为 V。

(1) $L=0$，以树根顶点为 V

(2) 寻找 V 顶点没有检测过的关连边

① 存在上述性质的边。任取其中一条，作已检测标志，设该边相连的 $L+1$ 级顶点为 x，转(3)。

② $L=n$，转(5)。

③ 不存在未检测过的边，转(4)。

(3) 计算与 x 点相连的各子路径的最优部分解下界 B_x：

$$B_x=\sum_{i\notin I}h_i \qquad h_i=\min_{j\notin J}\{W(i,j)\}$$

① 如果

$$\sum_{(i,j)\in S^*}W(i,j)+B_x\geqslant U \quad \text{转(2)。}$$

② 如果

$$\sum_{(i,j)\in S^*}W(i,j)+B_x<U,\ L=L+1,$$

将 x 顶点装入 S^*，用顶点 x 作为当前检测顶点 V。转(2)。

(4) 删去 S^* 中 L 级顶点

$L=L-1$，在 L 级顶点中找与顶点 V 有关连边的顶点 P。

① 找到顶点 P，用 P 代替顶点 V，转(2)。

② $L<0$，过程全部结束，已获得最优解，即 S，U。

否则，重复(4)。

(5) 找到一个新的当前最优解

$$U=\sum_{(i,j)\in S^*}W(i,j);$$

$$S=S^* \qquad \text{转(4)。}$$

四、分配问题的近似算法

对于大型的分配问题，考虑到处理效率通常需要采用近似方法。

1．行扫描法

行扫描法[50]是顺序检查所有的行，并选出每一行中未覆盖的最小元素 h_{i1}，然后把这一行分配给所选出的元素所在的那一列，且覆盖该列。当所有的行处理完后，每一行和列都唯一地分配给了对应的列和行，即构成了一个分配问题的解。$\sum_{i=1}^{n} h_{i1}$ 即为这个分配问题解的费用值。

行扫描法的计算时间随 n^2 增长。当费用矩阵的元素在 0 和 1 之间均匀分布时，行扫描法在寻找最大费用分配解时，它的期望相对误差的界为 $[\rho_n(n+1)]/n$，即当 n 变大时趋于 0。

2．损益分析法

损益分析法[57][47]是对行扫描法的一个改进。首先对费用矩阵的行进行扫描，求出每行中未被覆盖的最小元素 h_{i1} 和第二个最小的元素 h_{i2}（当 h_{i2} 不存在时，可令 h_{i2} 等于 h_{i1}）。并计算它们的差 $S_i = h_{i2} - h_{i1}$。然后以 $F_{(i)}$ 函数进行选择：

$$F_{(i)} = S_i * P - h_i,\ P \gg h_i,$$

选择具有最大 $F_{(i)}$ 值的行首先进行分配，即将该行的最小元素 h_{i1} 分配给 h_{i1} 所在的列，且覆盖 h_{i1} 所在的行与列。当所有行都被覆盖后，即求得了一个分配问题的解。$\sum_{i=1}^{n} h_{i1}$ 即为这个分配问题解的费用值。

损益分析法的计算时间随 n^3 增长。它的解的精度总是等于和高于行扫描法的求解精度。

第五章　改善布局的方法

5.1　引　　言

一、迭代改善布局概述

迭代改善布局过程是决定布局质量的关键过程，也是在布图设计中需要计算时间最多、设计代价最大的步骤之一。由于布局问题的复杂性，改善布局的过程通常是一个逐步优化的过程。与构造布局不同，改善布局是从现有的布局结果出发，按特定的选择规则，选择一个单元子集进行某种单元位置和方位的变换，从而形成一个新的布局。对新的布局用一个相应的确定的目标函数给以定量的评价。如果结果得到了改善，就用新的布局代替原来的布局，否则保留原布局结果。反复使用上述过程，直至找不到一个新的更好的(以特定的目标函数为标准)布局为止；或者以其它的系统控制条件(如使用的计算时间、改善率等)决定是否结束该布局的迭代改善过程。

二、构成改善布局方法的要素

从上述对迭代改善布局过程的大致描述可以看到，构成一个迭代改善布局方法的要素是：

① 确定待变换单元子集——迭代对象的确定及迭代变换的方式。

② 布局结果质量的评价标准——迭代改善的目标函数。

不同的迭代方式和不同的迭代改善目标函数形成不同的迭代改善布局方法。实际应用时，其组合变化很多，为了阐述的方便，本书将以迭代方式为分类标准，介绍各种改善布局方法的主要思

想，并在相应的章节中对一些特定的目标函数作较详尽的讨论和介绍。

5.2 对交换方式的迭代改善布局方法

对交换方式(pairwise interchange)在概念上是极为简单的。它的一般形式是，通过芯片上一对单元的互换位置而产生一个新的布局，这是一种用得最广泛的迭代改善方式。结合不同的目标函数就形成了不同的迭代改善方法。

一、成对交换法

1. 基本思想

在门阵列模式中，由于单元的大小、形状都相同，在对交换时不存在单元特性上的限制。因此，迭代对象的确定可以是随机的。当选定一个单元后，可依次与其它单元逐个地交换位置，每次交换后，计算与该二单元相关的线网连线总长(通常以覆盖线网所有接点的最小矩形半周长来近似)之和，看其是否下降，若改善，就用新的布局代替原布局。值得注意的是，这里讲的其它单元包括母片上的闲置单元，在这种情况下，实际上可理解为将选定单元移到某一空位上去。

2. 迭代对象的选择

为了得到最高的优化速度，较好的方法是根据单元的联结性关系或它的连线费用的大小来确定迭代对象。

文献[54]把具有"最多联结"的单元 x 选作迭代对象、即单元 x 具有 $Max\Sigma P(x, y)$ 值（其中 y 为除单元 x 外的芯片上的所有单元），并优先与具有较小的 $\Sigma P(u, y)$ 值的单元 u 作试交换。以此原则，反复对全部单元进行交换。直至每次循环中连线长的改善值低于某给定界限为止，这时改善过程结束。

文献[55]对现有的布局结果，计算每个单元 i 的连线长费用

对总费用的贡献：

$$C(i)=\sum_{j}P(i,j)\cdot d_{p(i)p(j)}$$

其中：$d_{p(i)p(j)}$ 是单元 i 和单元 j 的距离。$C(i)$ 值愈大，表明单元 i 的连线长度在连线总长中所占份额愈多，因而变换 i 单元的位置，有可能得到更大的布局目标函数(连线长函数)的改善值，即得到较快的改善率。在该文献中，将单元 M(具有 $\max\{C(i)\}$ 值)与所有其他单元试交换，对具有最大改善值的单元对进行实际的交换，形成新布局，如果最大改善值为 0，则不交换。

对于所有 $i\neq M$ 的单元，重新计算 $C(i)$。重复上述过程，直至满足过程终止条件为止。

在这些早期的对交换方法中，试交换次数非常多，实际上其中大约 90% 的试交换结果是不可行的。

二、力矢量成对松弛法

1．方法的物理模型及算法思想

力矢量成对松弛法(force—directed relaxation)[32]，采用一种力学模型来描述布局问题和定义相应的目标函数。在这种方法中，可把每一个单元看作是一个用若干弹簧或橡皮带连着的模块。当二个单元间存在公共的线网时，就设想其存在着一条橡皮带把这二个单元联结了起来。二个单元间受的拉力，由虎克定律 $F=kS$ 确定。这里 S 为单元间的距离向量，k 是弹性系数，它的数值等于二个单元之间的联结度 $P(i,j)$。这样，对一个布局来讲，单元 x 上的总力为所有与单元 x 有关单元对的合力，可用 F_i 表示：

$$F_i=\sum_{\forall j}P(i,j)S_{ij}$$

其中，j 为与单元 i 间存在公共线网的单元。

如果单元 i 处于自由状态，显然当 F_i 不等于 0 时，单元 i 将开始移动，直至到达某一合力 $F_i=0$ 的稳定位置为止。我们称这个稳定位置为单元 i 的目标点 T。有时也称作单元 i 的零张力位

置。我们把以单元 i 的原位置为原点，原点与目标点 T 的矢量称作单元 i 的目标矢量 $F_t(i)$：

$$F_t(i)=\frac{F_i}{\sum_{\forall j}P(i,j)}=\frac{\sum_{\forall j}P(i,j)S_{ij}}{\sum_{\forall j}P(i,j)}$$

因此，目标点 T 的坐标 (x_t, y_t) 可由下式得出：

$$x_t=\frac{\sum_{\forall j}P(i,j)x_j}{\sum_{\forall j}P(i,j)}$$

$$y_t=\frac{\sum_{\forall j}P(i,j)y_j}{\sum_{\forall j}P(i,j)}$$

其中，x_j, y_j 是单元 i 的位置为原点时，单元 j 所在位置的 x、y 坐标。

目标点 T 的位置，实际上就是与单元 i 相关的所有单元位置坐标的加权平均值。它和单元 i 有关连线总长最短的位置存在一定的对应关系（注意，这里讲的仅是一定的对应关系，实际上，它们并不等价）。

显然，如果所有的单元都处于自由状态，那末该单元系统到达稳态后应当处于最小张力状态，即最小能量状态。理想的情况是所有单元都处在其零张力位置，总张力为零。一般地讲，即处于 $\sum_{\forall i}F_i$ 为最小的布局位置，这也就是力矢量松弛法由自己的模型所得到的布局目标函数。

2．力矢量成对松弛法处理要点

在力矢量成对松弛法中，其处理的要点如下述：

① 首先对现有布局计算每个单元的零张力目标位置。

② 选择一对单元 (i, j)，使单元 i 的目标位置在单元 j 所在位置的一个给定的邻域内，或者单元 j 的目标位置是在 x 的位置的给定的邻域内。

③ 将这一对单元进行试交换，如果布局有所改善（即 $\sum_{\forall i}F_i$

减少或连线总长减少)，则实行交换，并形成新布局。重复上述②③的处理，把那些在当前循环中未被交换过的满足②所述条件的所有单元对都处理一遍。

④ 如果再也找不到任何满足②所述条件的，且使布局得到改善的单元对时，迭代改善过程结束。否则从①开始，进入又一个新的循环。

一般认为，采用力矢量成对松弛法，可减少试交换的随机性，从而可缩短处理时间，提高迭代改善效率。

三、多元胞模式中的对交换法

1．多元胞模式布局设计的目标

在多元胞模式设计中，布局的主要设计目标是：在满足布线需要的前提下，如何在尽可能小的芯片面积内实现单元的安置。我们知道，芯片的面积 S 取决于布局设计结果的长度和宽度。对于一个典型的多元胞芯片来讲，它的长度 L 和宽度 W（有时也称高度 H）与单元安置及通道区有如下关系(见图 5.1)：

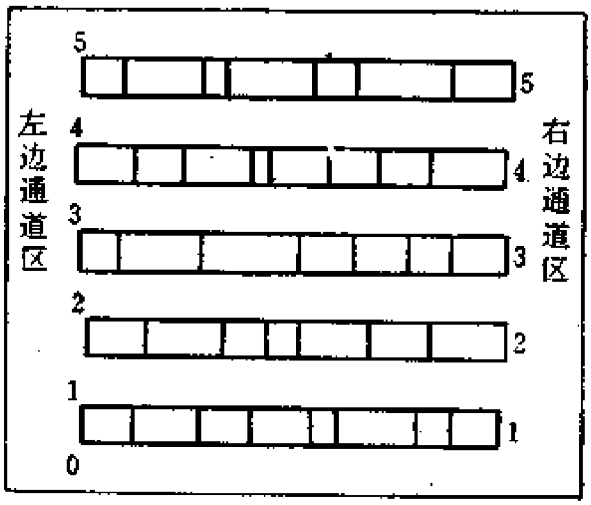


图 5.1　多元胞模式

$$\text{宽度 } W(\text{或高度 } H) = \sum_{i=1}^{m}(\text{单元行高度})i$$

$$+\sum_{i=0}^{m}(\text{第 } i \text{ 通道区的最大布线密度})\times(\text{每条布线通道宽度})$$

长度 L = Max{第 i 块的实际长度}

+(左边通道区最大通道密度

+右边通道区最大通道密度)

×(每条布线通道宽度)

通常在多元胞布局中把二维的布局问题分成二个一维问题，分别构造和迭代改善。

在块的迭代改善时，常常采用成对交换法。

此时的目标函数是一个分级的目标函数。首先考虑的是如何通过迭代改善使块的实际长度最大值在块的总数不变的前提下最小化。

由 4.8 节可知，在双边单元多元胞模式中块的实际长度 BL_i 为(当不考虑单元中的冗余通道时)：

$$BL_i = \sum_{j=1}^{n_i} L_{ij} + N_i \times W$$

其中：L_{ij} 为第 i 块中单元 j 的长度；N_i 为穿越 i 块的线网总数；

W 为一条布线通道的宽度。

由于目标函数的主项是 $\{BL_i\}$，因此布局的主要目标就是使 Max$\{BL_i\}$ 最小化；其次，即在 Max$\{BL_i\}$ 至少不增加的前提下，使 y 轴方向的连线总长最小化。在多元胞模式中，y 轴方向的连线总长与 $\sum_{V_i} N_i$ 有着极好的对应关系，因此，通常用 $\sum_{V_i} N_i$ 最小化作为目标函数的次项。可以看到，$\sum_{V_i} N_i$ 和 Max$\{BL_i\}$ 既存在一定的联系，又具有独立变化的可能。

2. 迭代改善的方法

(1) 单元目标位置的确定

为了提高迭代效率，我们可利用等分接点法计算每一个单元的目标位置(应放在哪一块中)和应取的方位。

(2) 迭代对象的选择

在迭代对象的选择时，也遵循一个分级优选的原则：

① 首先选择处在当前实际长度最长的单元行内的单元作迭代对象。

② 优先选择目标位置离当前位置最远的单元作迭代对象。严格地讲，这里的距离概念是一个以单元位置改变后 $\sum_{Vi} N_i$ 减量来度量的值。

③ 优先选择单元长度值大的单元作迭代对象。

(3) 交换对象的选择

在选择交换的对象时，可优先在选定单元的目标单元行中进行试交换。

(4) 迭代方式

在实际系统中[47]存在着三种可能的迭代功能：

① "自身迭代"，即单元 i 并不与任何其他单元交换位置，而仅仅是改变自身的位置方位。这似乎是一个单元与它自身进行了一次迭代。此时，迭代成立的条件是 $\sum_{Vi} N_i$ 下降，显然，此时单元 i 所在行的实际长度也必然下降。

② "一对零"迭代，实际上即单元 i 移入另一单元行并取适当的方位。这似乎与其交换位置的另一单元是一个"空"单元。此时，迭代成立的条件是：

$\mathrm{Max}\{BL_i\}$ 下降；或

$\mathrm{Max}\{BL_i\}$ 不变而 ΣN_i 下降。

③ "一对一"迭代，即单元 i 和另一单元行(块)的单元 j 交换位置，并进行相应的方位选择。其迭代成立条件为：

$\mathrm{Max}\{BL_i\}$ 下降；或

$\mathrm{Max}\{BL_i\}$ 不变而 ΣN_i 下降。

引文 [56] 在单边单元多元胞模式中，还允许在二个块之间进行"一对 n"的迭代，其中 n 可等于 1、2、3、4，实际上这是一种广义的"一对一"迭代。

反复进行上述的迭代对象选择(各种形式的迭代试探)及修改布局的过程，直至已找不到任何满足迭代成立条件的迭代对象或满足其它终止条件(如花费的总机时控制条件，迭代改善率控制条件等)为止，这时迭代改善过程就结束。

由于多元胞模式块间的单元迭代受单元长度不同的限制和单元所在块的实际长度的制约，在构造布局后再进行迭代改善将可能使一些合理的交换因上述限制而无法实现。贝尔实验室LTX系统把块的初始构造和块间单元迭代穿插进行，可在一定程度上较好地解决上述问题(见4.9节)，其条件是必须采用块的并行构造法。

多元胞模式中，当块间单元迭代改善完毕后，单元间相对的 y 坐标就被确定了，各单元行的实际长度也被确定了，芯片的长度也得到了确定。下一步将进行各块中单元位置的初始构造和迭代改善(见5.3节)。

四、最小割算法

本节主要阐述最小割算法(MIN—CUT)[57]。我们知道，布局的最主要目标是满足布线的需要，并在此前提下追求其它设计目标。传统的布局方法从连线总长最短着手，设计经验表明，它确实在许多情况下都是相当有效的，但另一方面却也表明它与布局的实际目标存在不对应性。如在门阵列模式中，布线完成率与连线总长间存在着图5.2所示的近似关系。即在比较差的布局中(用连线总长来度量)，随着迭代改善，连线总长减少了，布线完成率随之上升。但是，当连线总长减少到某一值后，随着连线长的下降布线完成率反而下降。在另一些问题中，也往往会出现这样的情况，二种布局，如果从连线总长来看，它们相差不大；但布线完成率却可能有相当大的差别。

1. 最小割算法的目标函数

与传统的连线总长最短目标函数不同，最小割算法试图用与

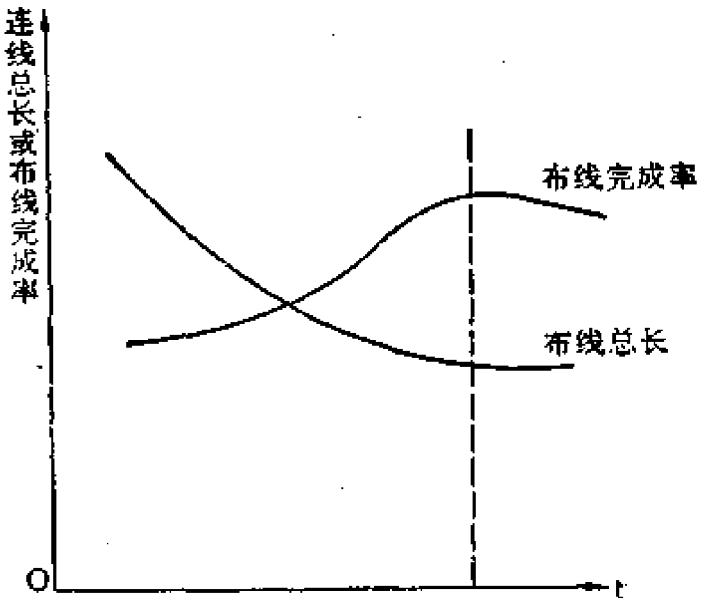


图 5.2　布线完成率与连线总长间关系

可布性有更好的对应关系的割值函数来作为目标函数。

定义：直线 C 对线网集的割值 N_C。

设直线 C 为 $y=y_i$ 的一条直线且横贯芯片。若线网接点集 $S_k=\{V_1, V_2, \cdots V_m\}$ 满足：$\exists V_i \in S_k$ 且 $y_{Vi} \geqslant y_i$，同时 $\exists V_j \in S_k$ 且 $y_{Vj} \leqslant y_i$。则称直线 C 对线网 S_k 进行了切割。令线网集中被直线 C 切割的线网子集为 s，则直线 C 对线网集的割值 N_C 等于 s 子集的元素数，即 $N_C=|s|$。

N_C 也可称作为 $y=y_i$ 处的布线密度。其实际意义为完成布线时在 $y=y_i$ 处所需布线通道的下界值。

由上述评价参数 N_C 出发，可以引出一些新的布局目标。

① 在母片式中，使每一处布线密度小于该处实际存在的布线通道容量 Q。设芯片长度为 L，通道宽度为 W，$y=y_i$ 处单元总长为 SCL，则 $y=y_i$ 处芯片上实际通道容量 $Q=(L-SCL)/W$。上述目标可描述为使在任一处 $y=y_i$（或 $x=x_i$），有 $N_C<Q$。

② 在母片式上，上述①的目标与使芯片上各处布线密度均匀化有着极好的对应关系，就是说，与使 $\mathrm{Max}\{N_C | C \in \forall$ 水平线和 $\forall$

垂直线）最小化的目标有很好的对应关系。

③ 在多元胞模式中，使通道区布线密度最大值最小化，从而使芯片面积实现最小化。

在实际使用最小割算法时，存在着三种具体的目标函数：

(1) 线网总割值 $N_C(SUM)$ 最小化目标函数

$$N_C(SUM)=\Sigma N_C \quad C\in H, V$$

其中，H，V 为所有沿 x 轴，y 轴间隔为单位距离的平行线。

显然，上述目标函数与用最小半周长估算的连线总长最短的目标函数是等价的。满足该设计目标的布局设计问题是一个 NP 完备问题，在有限时间内取得最佳解是不可能的。

(2) 最大割值最小化目标函数

$$N_C(mM)=\min\{\mathrm{Max}\{N_C|C\in\psi\}\}$$

其中，ψ 为预先定义的一个割线集。

对多元胞模式的每个通道区来讲，$N_C(mM)_i$ 对应着该通道区所需要高度的下界。值得注意的是，对多元胞模式来讲，垂直割线集的 $N_C(mM)$ 仅是芯片上所需水平通道总数的一个松的下界，其更严格的下界值应为 $\sum_{i=1}^{n+1}N_C(mM)_i$（$N_C(mM)_i$ 为第 i 个水平通道区的下界）。

上述目标函数与连线总长最短化目标函数并无一定的对应关系。这种目标函数下的最佳化的实现也是相当困难的。

(3) 序列割线的割值最小化目标函数

$$N_C(s)=\min N_{C_r,r}|\min N_{C_r,r-1}|\min N_{C_r,r-2}|\cdots|\min N_{C_r,1}$$

其中：C_i 为第 i 个割线序列；

$C_r=C_1, C_2, C_3, \cdots C_r$ 为一个有序集合；

“|”可读作“以……为条件”，如 $\min N_{C2}|\min N_{C1}$ 即表示在以 C_1 割线处达到割值最小化的条件下使 C_2 割线处的割值最小化。

$N_C(s)$ 函数描述了对一个有序割线集求得序列割值最小化的过程。实际算法中一般以 $N_C(s)$ 为目标函数。

2. 最小割算法

(1) 块和块的划分(block and block division)

设在一个矩形域 $R(A)$ 内，存在一个属于该域的可移动的元素集 $E(A)$，每个属于 $E(A)$ 的一个元素都分配了一个在 $R(A)$ 中的位置。

设割线 C 将 $R(A)$ 域分为 $R(A_1)$，$R(A_2)$ 二个子域，将元素 $E(A)$ 分成 $E(A_1)E(A_2)$ 二个子集。其中，$E(A_1)\cup E(A_2)=E(A)$，$E(A_1)\cap E(A_2)=\phi$ 且 $|E(A_i)|\leqslant|R(A_i)|\ i=1,2$。上述过程称为块的划分过程，$R(A)$ 称为第一级块，$R(A_1)$ 称为第二级块，也即定义第 j 级块分划后得到的为第 $j+1$ 级块。最小割算法的目标即为实现一个分划过程并使割线 C 处的割值 N_C 最小化。一般讲来，这

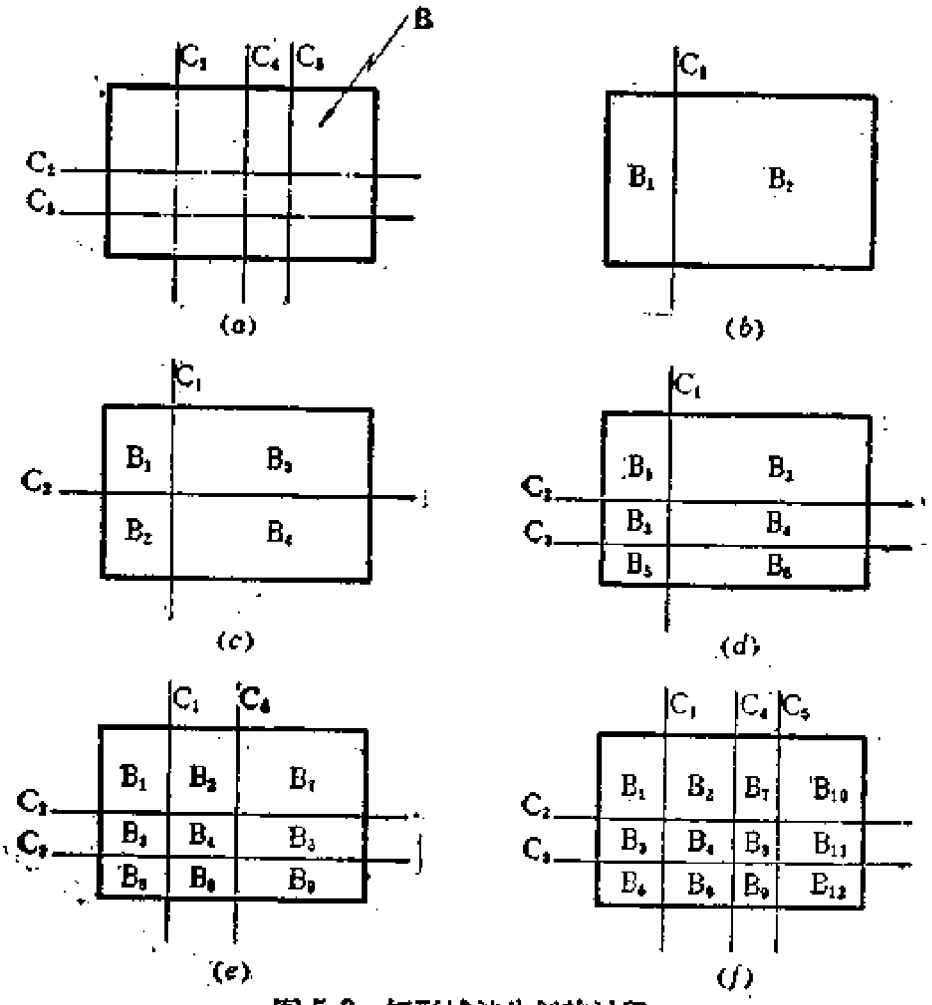


图 5.3 矩形域被分划的过程

是一个相当困难的问题。

(2) 面向割线的最小割布局算法 (cut oriented min-cut placement)

这种算法以割线为基础，即对一条割线来讲，所有与其交叉的块都将被其分划成二部分。图 5.3 所示为矩形域 B(一级块 B)被割线序列 C_1, C_2, C_3, C_4, C_5, 分划的过程。

在这种算法中以 $N_C(s)$ 为目标函数，以图中 C_3 割线为例，C_3 切割 B_3, B_4, 它是 C_1, C_2 的后续切割。N_{C3} 的最小化是指如何把 B_3, B_4 中的元素重新安置，各分为二部分而使 C_3 处的割值最小化，显然这将不会改变原 N_{C2}、N_{C1} 的值。这时为提高处理效率，对于被切割的块的子集也不是同时进行处理，而是逐个地处理。如对割线 C_3，可先处理 B_3，求得 B_3 的最佳分划，然而利用 B_3 的分划结果考虑 B_4 并求得 B_4 对 C_3 的最佳分划，然后再处理 B_3, …直至不能继续改善为止。这时，每次求出的不是割值的最小值而是割值的局部最小值。

在求块 B_i 对 C_j 的最佳分划时，通常利用对交换法。

在门阵列模式中，设 B_i 被 C_j 分划为 B_{i1}, B_{i2} 二个子集。$B_{i1} \cup B_{i2} = B_i$, $B_{i1} \cap B_{i2} = \phi$。迭代对象的选择原则如下述：

在 B_{i1} 中选择一个单元 X，使 $f(x)$满足

$$f(x) = \mathrm{Max}\Big[\sum_{\substack{x \in B'_{i2} \\ z \in B_{i1}}} P(x, z) - \sum_{\substack{x \in B'_{i1} \\ z \in B_{i2} \\ x \neq z}} P(x, z)\Big]$$

其中 B'_{i1}, B'_{i2} 分别为割线含 B_{i1}, B_{i2} 一侧的全部单元的集合。

在 B_{i2} 中选择一个单元 y 使 $f(y)$满足

$$f(y) = \mathrm{Max}\Big[\sum_{\substack{x \in B'_{i1} \\ y \in B_{i2}}} P(y, z) - \sum_{\substack{x \in B'_{i2} \\ y \in B_{i1} \\ y \neq z}} P(y, z)\Big]$$

并令 B_{i1}, B_{i2} 中的空位之选择函数值 $f(x)$或 $f(y)$ 等于 0。

当 $f(x) + f(y) > 0$ 时，将对应的 X, Y 单元交换位置(或是移动其中一个单元到另一位置)。

当 $f(x) + f(y) \leqslant 0$ 时，B_i 被 C_j 分划过程结束。

当所有线网都为二接点线网且线网权重都为1时，上述迭代改善过程结束时实现的是割值最小化目标。在一般情况下，上述迭代过程实际上实现的是一个带权的割值最小化目标。

整个算法过程描述如下：

STEP 1：选定一个割线的处理序列。

STEP 2：在割线序列中选下一条割线进行处理。

STEP 3：设割线 C_j 通过块的子集 $\beta=\{B_1, B_2\cdots B_t\}$，求 β 的一个划分，使 N_{Cj} 最小。

STEP 4：通过 C_j 的处理，β 中各块都分划成了二个新的块。

STEP 5：如果割线序列已处理完毕，或每个块至多只有一个可移动单元时，过程结束，否则转 STEP 2。

(3) 面向块的最小割算法 (block oriented min-cut placement)

此时，算法以块为基础（或以块的一个子集为基础）。这种方式也称之为对分法(bisecting)，即对于指定的块或块的子集用一条(或一个序列的)割线来进行切割。此割线对非指定的块不起切割-分划作用。如在图 5.3(c) 中，如果 B_4 中元素远比 B_2 中元素多。我们可以首先选择 B_4 块，用 C_3 割线对其进行切割。在面向块的最小割算法中，C_3 将仅仅对块 B_4 进行切割并将其划分为 B_4，B_6。虽然这时 C_3 也与 B_2 交叉，但不考虑 C_3 对块 B_2 的切割。

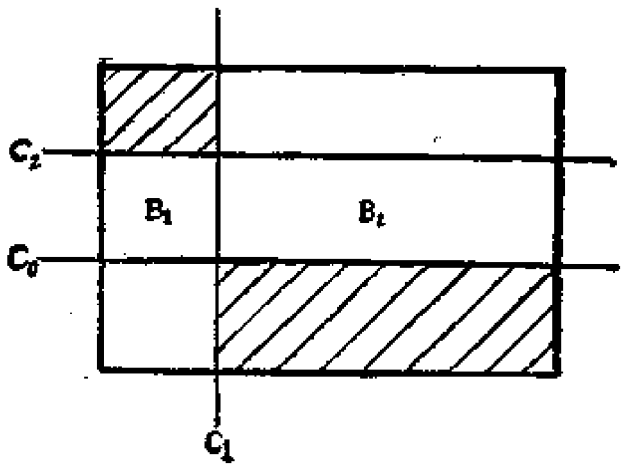


图5 4 一个面向块最小割示意图

图 5.4 为一个面向块最小割的例子，C_2 切割 B_1，C_3 切割 B_2。我们可以首先在 B_1 中对割线 C_2 求最佳分划，然后在 B_2 中对 C_3 求最佳分划。此时，可利用 B_1 中阴影部分（C_1 以上部分）元素的信息，然后在 B_1 中对 C_2 迭代改善原分划时，可利用 B_2 中阴影部分的元素信息。

值得注意的是在面向块的最小割算法中，目标函数已不再是 $N_C(s)$了。这样做的优点在于每次可根据块的具体情况，动态地决定割线的位置，有针对性地调整、改善某一部分的布局。但这时，它的目标函数已再不能用简明的方式予以描述了。

算法过程要述如下：

STEP 1：选择下一个块或块的子集 β。

STEP 2：对这些块选择一条割线 C_j。

STEP 3：求这些块对割线 C_j 的最佳分划，可采用对交换法对 β 中各块顺序处理，并可采取各块间迭代处理方法。

STEP 4：用分划得到的一些新的块代替原来的块。

STEP 5：如果所有的块仅含一个可移动元素，迭代布局过程结束，否则转 STEP 2。

（4）块和割线的选择

上述二种算法在原则上也可用于准任意元（semicustom）和多元胞模式。不同的是在利用对交换进行迭代改善时，不仅需要考虑单元间的联结度关系，而且必须考虑单元的大小和区域的容量（或区域内的单元面积总和）。

由上述算法也可看到，最小割算法的处理效率及处理精度与选择的块与割线的处理顺序也有很大的关系。

在块的处理顺序上存在着二种基本的方式：

① 广度优先方式（breadth-first method） 在这种处理方式中，只有所有 j 级的块都已分划、处理完毕后，才开始处理 $j+1$ 级的块。

② 深度优先方式（depth-first method） 在这种处理方式

中，每次处理的是允许继续分划的块中具有当前最高级别的块。

割线一般选择在安置位置的行和列间，这里存在着二种基本的处理顺序：

① 条状切割(slice cut)　如图 5.5(a)所示，在条状切割中，选择割线的处理顺序为 $C_1, C_2, C_3, \cdots$。使每次分划时，割线一侧(下侧)具有大致相同的元素数。

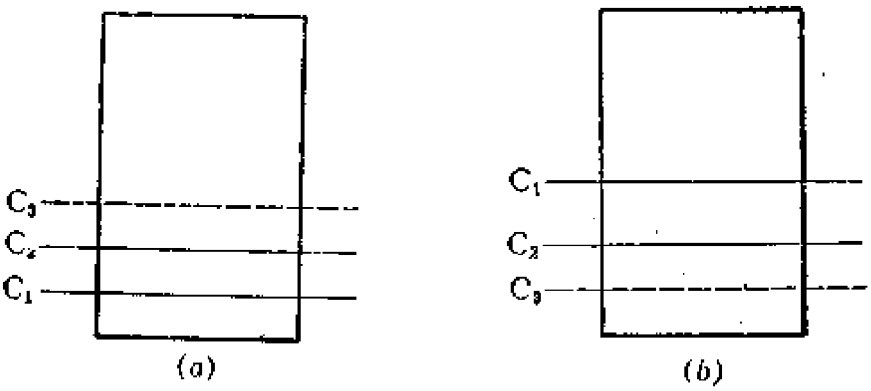


图 5.5　条状切割示意图

② 对分切割 (bisection cut)　如图 5.5(b) 所示，这时割线 $C_1, C_2, C_3, \cdots$使每次被切割的块都被分划为二个元素数大致相同的下一级块。

此外，也可由程序动态地决定割线的选择，设存在一个初始布局并用最小割算法进行迭代改善，可在一维方向（如 x 方向）首先以单位距离作平行于 y 轴的割线集$\{C_i\}$，并计算各处割值 N_{Ci}，然后令在 $x=x_i$ 处的可用通道数下限值为 V_i，则可计算出各处相应的溢出值 $S_i=N_{Ci}-V_i$。一种割线选择的方法就是每次首先选当前具有 $\mathrm{Max}\{S_i\}$值处的割线进行处理，通过迭代，降低该处的割值。

(5) 三种固定顺序的最小割布局算法

这里介绍的三种固定顺序的最小割算法，都具有一定的实际应用背景，它们都是面向块的最小割算法。

① 正交方式 (quadrature) 的最小割布局算法　这是一种适用门阵列模式的最小割算法。考虑到一般完成布局后，芯片中心

部分布线密度将高于芯片四周的布线密度(参看 4.9 节)以及序列割线割值最小化目标函数的特点，为使芯片上布线密度均匀化，可按排割线处理顺序为“水平——垂直——水平……”，交替地对分切割芯片。这是一种广度优先的切割方式，并首先保证芯片中心区布线密度尽可能低，再此在基础上求得四周的合理分划，从而使结果各部分布线密度比较均匀，如图 5.6 所示。

② 对分方式(bisection)的最小割布局算法　这种方法首先把芯片在水平(或垂直)方向连续地对分，直至各块不能继续对分为止，然后再在垂直(或水平)方向连续地进行对分，直至各块不能继续对分为止。在每个方向上，其处理顺序可以是广度优先，也可以是深度优先。这种方式很适用于多元胞模式，其实际意义可理解为首先把单元分配到行，然而通过垂直方向的分划将行内单元逐个分配到列，而最终完成了布局，如图 5.7 所示。

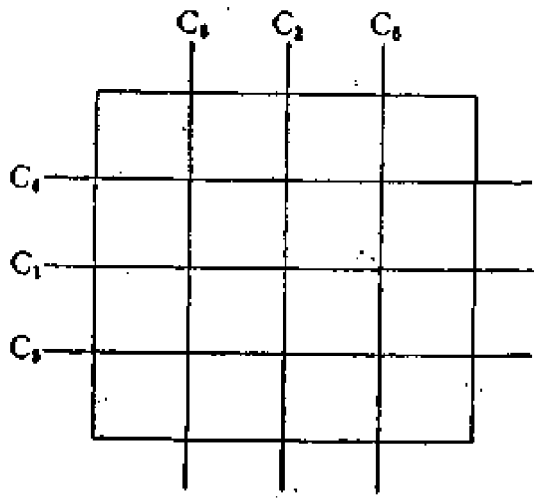


图 5.6　正交方式最小割示意图

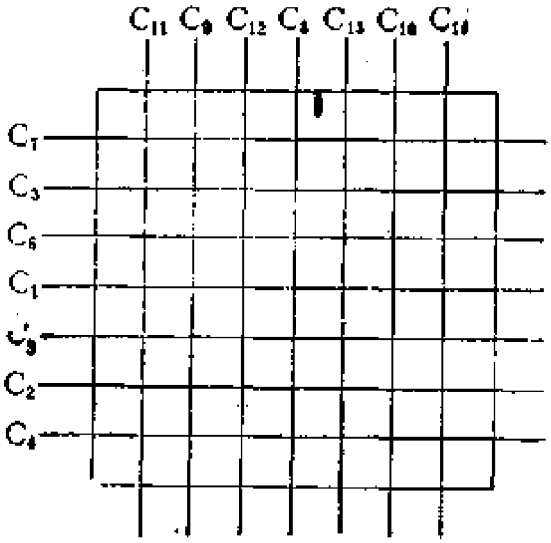


图 5.7　对分方式最小割示意图

③ 切条/对分方式(slice/bisection)最小割布局算法　这种方式首先从区域的一边开始，通过割线将单元集分为 k 和 $n-k$ 二个子集(其中 n 为单元总数，k 为大于 0 的常数)并使 N_C 最小化。重复上述过程，直至全部单元分配到行(图 5.8)；然后通过垂直对

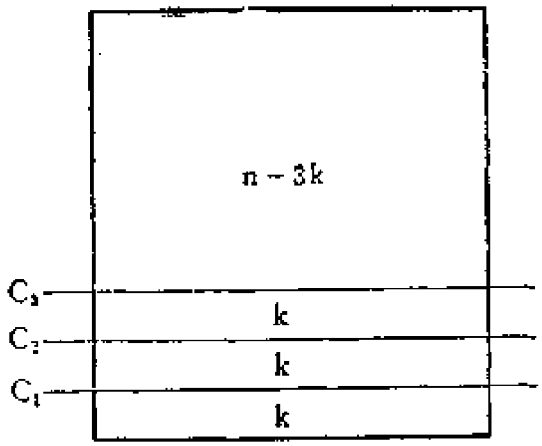


图 5.8　切条/对分方式最小割示意图

分将每行的单元分配到列。

这种方式的另一种形式是先进行对分，然后再切条，它比较适合近引出接点区域布线密度高的情况。

最小割算法由于其目标函数与可布性有较好的对应关系以及算法本身的多适应性。它是一种重要的迭代改善布局算法。

五、准任意元模式的改善布局方法

在准任意元模式中，单元为大小不同的矩形，引出结点排列在矩形的四边上。

1．评价当前布局的图模型

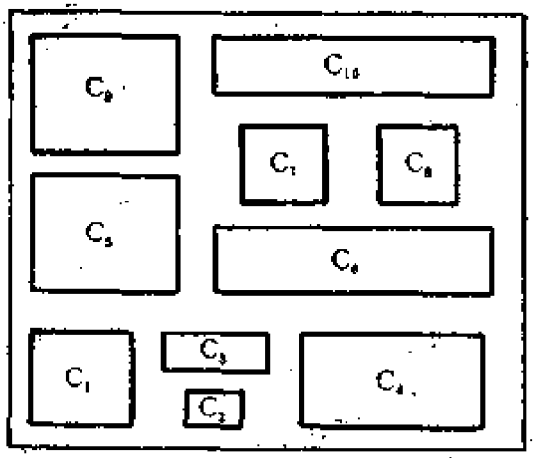


图 5.9　一个准任意元初始布局结果

为了在满足布线要求的前提下，使设计得到的芯片面积最小化，在得到了一个初始构造布局后，必须进行迭代改善。此时首先必须解决的问题是如何评价当前的布局质量。

图 5.9 是一个准任意元的初始布局结果，文献 [58][59] 介绍了用三种图来描述布局的结果，即通道区交叉图 G_w，水平的通道区位置图 G_x 和垂直的通道区位置图 G_y。

(1) 通道区交叉图

图 5.10 就是图 5.9 布局结果的通道区交叉图，图中每一个顶点对应着二个通道区的交叉点。每条边描述了布线通道区或子通道区。

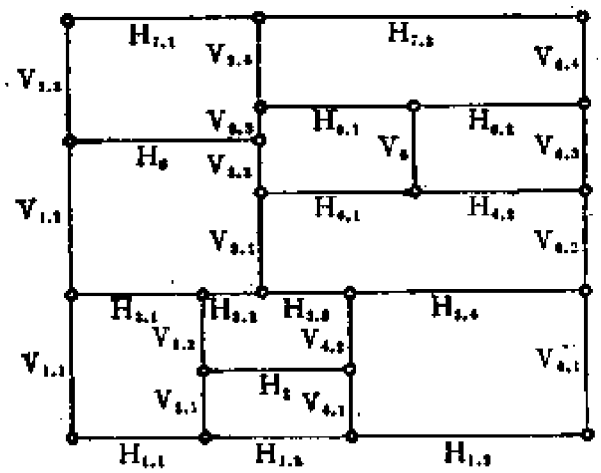
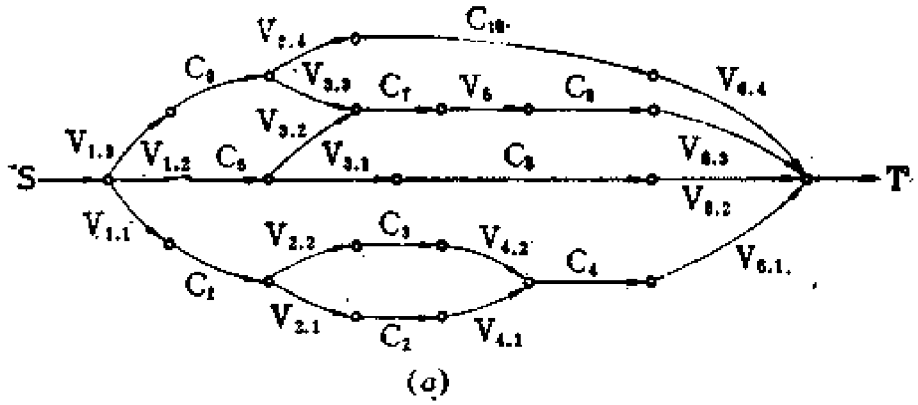


图 5.10 通道区交叉图

通道区交叉图描述了芯片上各布线通道区的相互关系，并提供了一个进行总体布线(参看第七章)的不等大小的矩形网格图。它的各边的权重可由总体布线的结果决定。边的权重描述了该通道区中满足布线要求所需通道数的下限值。

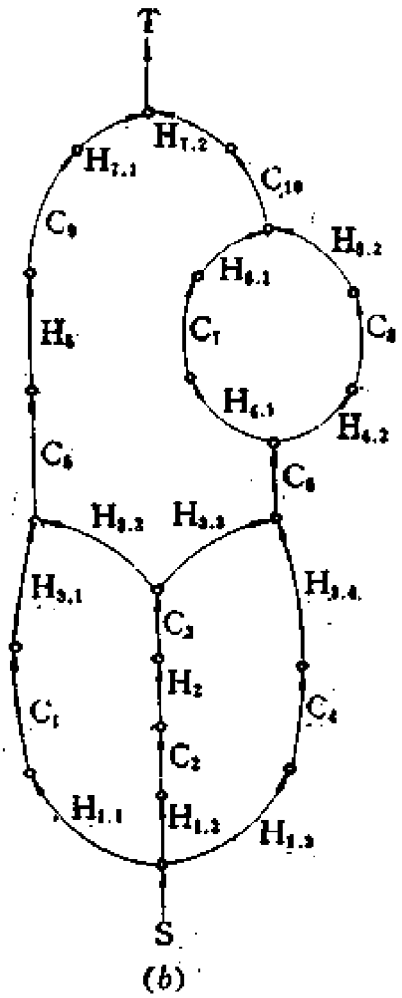
(2) 通道区位置图

图 5.11(a)、(b) 分别为图 5.9 所示布局结果对应的水平的通道区位置图(horizontal channel position graph)和垂直的通道区位置图 (vertical channel position graph)。图中每个顶点对



(a)

(a) 水平通道区位置图 G_a



(b)

(b) 垂直通道区位置图 G_v

图 5.11　为图 5.8 布局结果的水平和垂直通道区位置

应于单元和通道区邻接的界面，每条边分别对应着相关的单元或通道区。边的权重描述了单元在该方向的尺寸或通道区在该方向的宽度 [58]。在有些文献中 [59] 则令图中每个顶点对应着单元(或通道区)而与顶点相连的边对应着单元间的通道区（或通道区之间的单元)。

(3) 关键路径、关键单元和关键通道区

基于上述描述，我们可以方便地得到一些布局结果的重要信息。我们称由 S 点到 T 点的最长路径(路径上边权和最大)为关键路径 (critical path)，显然，关键路径的长度对应着布局设计后芯片该方向长度的下限值，因此 G_x 和 G_y 的关键路径长度乘积就可用来作为评价布局结果所需芯片面积的一个很好的参数。处于关键路径上的单元和通道区称之为关键单元和关键通道区，显然，为了改善布局，必须尽可能缩短关键路径的长度。而优先调整迭代的对象应是关键单元。

2. 迭代改善过程

迭代改善的过程将是一个循环过程。不断地试探一个新的布局，能否使 G_x 和 G_y 的关键路径长度乘积值 $S_{x,y}$ 缩小，直至满足某种过程结束条件为止(如连续一定次数的试探，布局无改善或过程处理时间已超过预定时间等)。

新的布局可以通过下述三种方式获得，即关键单元的方位调整(oriestation improvement)，关键单元的移位处理，以及成对交换。

(1) 方位调整

方位调整是指把选定的关键单元绕其中心点进行 $\pi/2$、$3\pi/2$ 弧度的旋转，或以其中心轴进行 x 或 y 向镜象变换。当单元的长度与宽度不等时，旋转 $\pi/2$ 或 $3\pi/2$ 将使通道区位置图中与该单元对应的边的权重发生变化。在这些变换中，一般讲并将使有关的通道区对应的边的权重也发生变化，从而可能获得一个 $S_{x,y}$ 改善了的新布局。

(2) 移位处理

移位处理是指把选定的关键单元相对于其它单元在水平或垂直方向移动一段距离。其目的是试图通过移位改变原关键路径中边的组成情况，从而可能获得一个 $S_{x,y}$ 改善了的新布局。如在图5.9的布局中，设当前 G_x 中的关键路径为 $V_{1,3} \rightarrow C_9 \rightarrow V_{3,3} \rightarrow C_7 \rightarrow V_5 \rightarrow C_8 \rightarrow V_{6,3}$。$G_y$ 中的关键路径为 $H_{1,2} \rightarrow C_2 \rightarrow H_2 \rightarrow C_3 \rightarrow H_{3,3} \rightarrow C_6 \rightarrow H_{4,2} \rightarrow C_6 \rightarrow H_{6,2} \rightarrow C_{10} \rightarrow H_{7,2}$。此时 C_9 是关键单元之一，设当将 C_9 从原位置上移一段距离，使 $V_{3,3}$ 通道区 y 方向长度为 0。这就使 G_x 中关键路径发生了变化。就有可能获得一个改善了的布局。

(3) 成对交换

成对交换是指对选定的关键单元选择一个交换对象，并将它们互换位置。这种交换一般应满足下述条件：即这种交换将使通道区位置图中对应这二个单元的边的权重发生变化，但不改变通道区位置图的拓扑结构。

上述这些变换，只有当变换处理后得到的新的布局 $S_{x,y}$ 值缩小(即芯片面积缩小)时，才予以承认，否则重新进行其它变换或选择其他关键单元进行处理。

(4) 迭代改善过程描述

下面用计算机语言方式描述了上述算法（包括初始布局部分）：

```
PLACE
  INITIAL PLACE
    ESTIMATE SIZE OF CHIP
    PLACE FIXED I/O PINS
    FOR EACH CELL DO
      FIND MAXIMALLY CONNECTED CELL
      PLACE THIS CELL
      END DO
```

```
    END INITIAL PLACE
  PLACE IMPROVEMENT
    FIND CHANNELS, DETERMINE SIZE, IDENTIFY
        CRITICAL CELLS AND CHANNELS
    ESTIMATE TRACK DENSITY FOR
        INTERCONNECTIONS
    REPEAT UNTIL STOPPING CONDITION DO
      FOR EACH CRITICAL CELL DO
        SEARCH FOR ALTERNATE ORIENTATION
        SEARCH FOR ALTERNATE LOCATION
          (MOVE)
        SEARCH FOR EXCHANGE PARTNER
        IF CHIP SIZE IS SMALLER DO
          MAKE THIS PLACEMENT CORRECT
          BREAK
          END DO
        ELSE DO
          RESTORE PREVIOUS PLACEMENT
          END DO
        END REPEAT
    END PLACE IMPROVEMENT
  END PLACE
```

从上面的讨论可以看到，单元大小不同的准任意元模式布局的迭代改善实际上是很困难的。这个领域也是一个有待于进一步研究开发的领域，尤其是在 LSI/VLSI 布图设计中引入分级设计思想后，尽管基本单元可以是规则的，但由基本单元组成的各级模块(module)即使都设计成矩形，要求它们大小相同或符合某种限制是很困难的。加上宏单元的应用(RAM、ROM PLA 等)使模块的布局问题成为一个典型的准任意元的布局问题。因此，准任

意元布局方法的研究对于分级设计来讲也具有重要的意义。

本节介绍的通道区位置图，通道区交叉图是一种很有用的描述手段。它不仅描述了单元的大小，相互位置关系，而且描述了单元间通道区的大小、通道区之间、通道区与单元间的相互位置关系。有了这种描述方法，布线和布局的相互关系，以及它们间的反馈迭代(feed back)过程的描述就变得简单了。

六、邻接交换法

1. 邻接交换法概述

邻接交换法(Neighbor hood interchange)[72]在概念上与成对交换法很类似，其区别在于互相换位的单元必须是邻接的。广义地讲，互相换位的单元之间的距离必须小于预先定义的邻接距离 S_P。

(1) 邻接交换法的目标函数

一般的邻接交换法可以连线总长最小化为目标函数，迭代对象 x 的选择可依下述函数确定：

$$f(x)=\mathrm{Max}\{\sum_{\forall y}P(x,y)\cdot d|P(x)-P(y)|\}$$

也可随机地确定。

(2) 迭代对象的选择

迭代对象 x 的交换对象 y 的选择范围为：

$$y\in\{Y|d(P(x)-P(y)|\leqslant S_P\}\quad y\neq x$$

在该范围内逐一进行试交换，并计算连线总长是否下降，选取下降最多的 y 与迭代对象 x 换位形成新的布局，此过程一直进行到满足过程结束条件为止。

2. 力指向交换法

当采用力矢量松弛法的概念和邻接交换法结合时，可形成一种新的迭代改善方法——力指向交换法(Force-directed interchange[28]，该方法的主要步骤可如下述：

STEP 1：计算每一个单元 x 的总联结度 $\sum_{\forall y} P(x, y)$ 并按照它们的总联结度值由大至小地将它们排列起来。

STEP 2：将单元序列中下一个单元取出，并计算出它的合力向量 F_x，及其相应的目标位置。以单元当前位置和目标位置为顶点作一矩形。

STEP 3：首先沿着矩形的长边进行水平的或垂直的交换。如果布局没有改善，就根据矩形的短边给的方向试交换，如果仍没有改善，则试验与对角线方向上的单元进行交换。

STEP 4：在每次成功的交换后，一个新的矩形被确定，可对原给定的单元，按新的矩形继续进行交换。当找不到一个改善了的新布局，或已到达目标位置时，对该单元的处理过程结束。为了使试交换有更高的成功率，一个稍微改进一点的办法是：每次成功的交换后，可重新计算目标位置（自然，每次交换后目标点位置至多改变一个单元）。

STEP 5：如果单元序列中还有未处理过的单元，转 STEP 2；如果已处理完，且这个循环中布局的改善小于预先给定的界限，整个迭代改善过程结束，否则转 STEP 1（从序列的第一个单元重新开始处理）。

邻接交换法一般来讲，可得到较为满意的结果，但处理速度比较慢。

七、多元胞模式中块内单元布局的迭代改善

1. 块内单元布局迭代改善的特点

在多元胞膜式中，在完成了块内单元的初始构造布局后，将对块内单元布局进行迭代改善，此时，如果用一般的对交换方法，则由于单元的长度不同（例如图 5.12 中，单元 x 与长度不同的单元

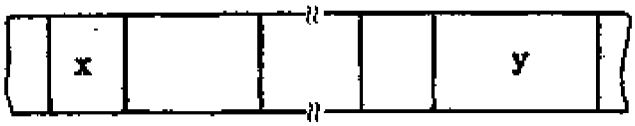


图 5.12　长度不同的对交换示意图

y 进行交换）将使单元 x 与单元 y 之间的所有单元的位置都随之发生改变。为了计算连线总长的变化情况就不仅需要计算与单元 x，y 相关的线网连线长的变化，而且还必须计算单元 x，y 之间各单元相关线网连线长的变化，因此使计算复杂性上升，效率下降，同时也使迭代对象的选择难以控制。而若采取邻接交换法（本块内直接邻接）则可避免上述问题。

2. 与方位选择结合的邻接单元交换法

引文[47]中介绍了双边单元块内迭代改善的具体处理过程。我们采用了与方位选择结合的邻接单元交换法。块内单元具有二种可能的方位，即 $+x$ 方位，$-x$ 方位（参见 4.6 节）。因此块内任一单元 i（除块内右端单元外）与其右侧紧邻单元 j 共存在八种可能的位置组合。即：(i, j)（表示单元 i，j 的原安置位置）、$(-i, j)$、(j, i)、$(j, -i)$、$(i, -j)$、$(-i, -j)$、$(-j, i)$、$(-j, -i)$。其中 $(-j, -i)$ 表示单元 i 和 j 换位并都取 $-x$ 方位，其中 $(-i, j)$、$(i, -j)$ 实际上即为单元单独进行方位调整的情况。我们对块内各单元对，各种组合情况求其连线总长下降最多的修改方式，并进行相应的交换、方位调整，直至不能继续改善为止。

3. 目标函数的改进

为了获得更合理的布局结果，改进的办法是以块间通道区的有关参数作为目标函数。这些参数主要有：通道区中在布线时需作专门处理的线网数量 U，通道区的最大布线密度，通道区的最大布线密度分布的广度等[62]。

(1) 通道区中在布线时需作专门处理的线网数量 U

通道区布线一般采用双层布线，而且将所有的垂直线段安排在一层上，将所有的水平线段安排在另一层上，二层间必须的连接通过通孔来实现。在上述布线条件下，图 5.13(a)、(b) 所示的情况中，其中至少有一条线网需作专门处理（必须作适当的转向处理 (dogleg) 或按其它分层原则实行布线）才能使线网 100% 地完成布线。这类线网（可参见第七章）的总数记作 U。

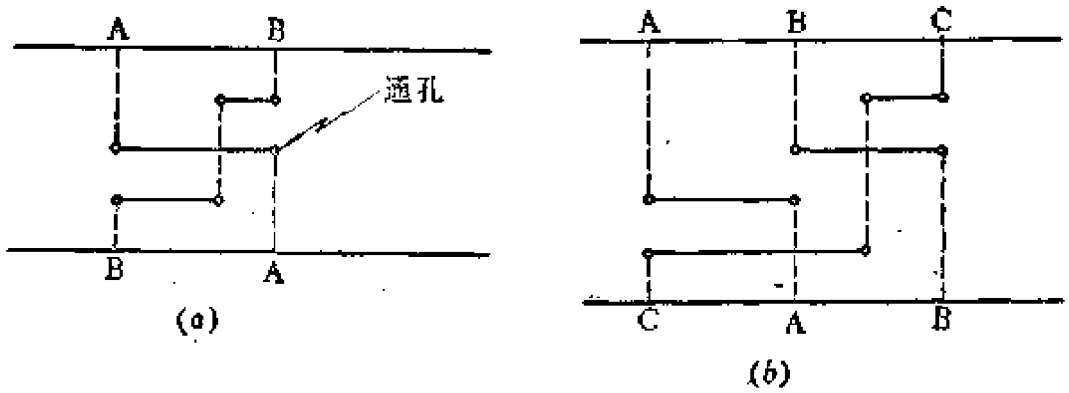


图 5.13　双层布线示意图

(2) 通道区的最大布线密度 D_{max}

通道区的最大布线密度D_{max}是指块相邻的布线通道区满足布线要求所需的通道数下限值。在这里通常考虑块相邻的二个通道区的最大布线密度之和 D_s。

(3) 通道区最大布线密度的广度 S

我们知道通道区最大布线密度 D_{max} 可利用割值的计算求得。即与通道区中轴垂直，以单位距离为间隔作平行的割线，分别求它们的割值，割值的最大值即为 D_{max}。而割值等于 D_{max} 的割线数量即称为最大布线密度的广度 S。显然，在其它条件相同的情况下，S 值大的通道区最终完成布线时实际需要通道数大于 D_{max} 的可能性也将较大。

在实际处理过程中，可以将上述诸参数构成一个分级控制的目标函数 Q，Q 值为块相邻的二个布线通道区的目标函数值：

① Q 值得到改善的条件为：

U_S 值下降　或

U_S 值不变而 D_s 值下降　或

U_S、D_S 值不变而 S_S 值下降；

② Q 值不改善的条件为：

U_S、D_S 和 S_S 值都不改变；

③ Q 值恶化的情况是：

U_S 值上升　或

U_S 值不变而 D_S 值上升　或

U_S、D_S 值不变而 S_S 值上升。

4. PRO 算法

贝尔实验室 LTX 系统 PRO(1D)算法(Placement for Routing Optimization)就采用上述分级控制的目标函数 [62]，其具体处理过程如下述：

① 从块的左端开始，将每个单元依次以 y 中心轴作镜象变换(即水平方位选择)如 Q 值改善，则认可该变换，否则进行下一个单元的处理，由左至右将整行单元处理完。

② 从块的左端开始，将每一个单元依次与其邻接的右侧单元互换位置进行试探，如 Q 值改善，则认可该变换，否则不承认而进行下一个邻接单元对的处理，直至处理到块内最右端为止。

③ 把块邻接的通道区的密度、广度和"不能"实行布线的线网数(即 U 值)通告用户。如果结果已令人满意，块内迭代过程结束，否则由①重新进行新的一轮处理。

此外，用户还可以选择决定当 Q 值不变时的"中性变换"是否予以成立。用户也可对分级控制的目标函数进行修改，如使处理过程仅考虑 D_s 的下降，而不考虑 U_S 和 S_S 的变化等。

大量的布图实践和布线算法的发展表明：U 的值即使开始时较大，要保证可布性并不十分困难。只要对少数等价接点进行处理或单元的方位选择处理就可使 U 值下降甚至等于 0。而对于 U 值不为 0 的布线问题，现有的一些布线方法 [149][151] 也可在只增加少量的或不增加通道的情况下 100% 地完成布线，同时，还因为 U 值的计算时间(一般采用近似方法，该问题本身是一个 NP 完备问题)往往比通道区最大布线密度的计算时间多几十倍，因此，实际的分级目标函数可以减少 D_S 为首要目标。关于 U_S、D_S、S_S 的具体计算方法我们将在第七章有关部分里进一步讨论。

5.3 链式交换方式

在成对交换方式中，以力矢量法为例，其理想情况是迭代对象单元A的目标位置若在B处，则希望B位置恰好是一个“空”位，或B位置上的单元的目标位置恰好在单元A的当前位置上。显然，这种情况在实际问题中出现的概率是很小的，即使把上述目标位置扩展为一个预先定义的邻接区域，成对交换方式仍可能丢失不少可能的迭代改善的机会。M．哈南和J.M．库尔茨伯格的力矢量松弛法[63]实际上采用了一种链式交换方式来克服上述成对交换方式的缺点。后来S.GOTO又将其发展为一种广义的力矢量松弛法[36]而取得了更好的结果。此外，贝尔实验室采用了一种“单元插入法”直接将单元插入到目标位置，但同时将引起当前位置与目标位置之间一系列单元的位移，实际上也构成一种链式交换的迭代方式。这种方式在一定程度上可弥补对交换方式的不足，提高布局的精度。

一、力矢量松弛法和广义力矢量松弛法

1．力矢量松弛法及与其它方法的联系

力矢量松弛法(force directed relaxation)的目标函数与力矢量成对交换法相同，但与迭代的交换方式不同。它允许同时进行λ^*个单元的交换以获得一个新的布局。如图5.14(a)所示，设单元A为当前布局中离目标点位置最远的单元，B为其目标位置，C、D为邻接B并满足一定条件的准目标位置。定义参量ε^*为选择的目标邻接位置总数。对于B位置的单元，设其目标位置在E。E位置的单元目标位置在Q……。定义另一个参量λ^*，为交换链的长度。这样，就可构造出一棵搜索树(有向树见图5.14(b))，单元A为该有向树的根，树上每一个顶点都为其上一辈顶点所对应的单元的目标位置(或准目标位置)，且限定每个顶点的下一辈顶

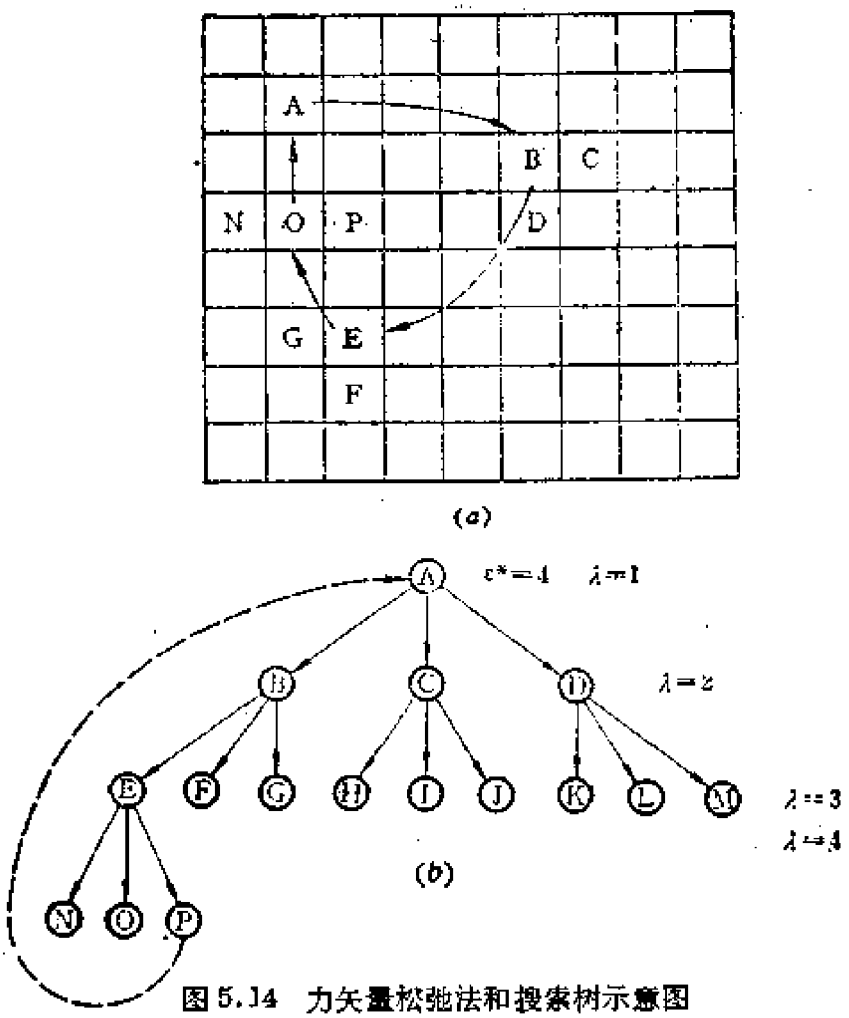


图 5.14　力矢量松弛法和搜索树示意图

点数为 ε^* 个。该树上最长的有向路径长度为 λ^*-1，即树上每个顶点的广度为 ε^*，搜索树的深度为 λ^*-1。显然，当 $\varepsilon^*=1$、$\lambda^*=2$ 时，该搜索树描述的就是力矢量成对交换法。当 $\varepsilon^*=1$ $\lambda^*=\infty$ 时，该搜索树描述的是力矢量松弛法（FDR）。当 $\varepsilon^*=N$（N 为单元总数）$\lambda^*=2$ 时，即为一般的对交换法。

2．广义力矢量松弛法

广义的力矢量松弛法的主要处理过程可如下述（以等分接点法确定目标位置，以连线总长最小化为目标函数）：

STEP 1：选择当前离目标位置最远的未标记单元 A 作为迭代对象，并确定其目标位置 B。根据 ε^*，选择其邻接的准目标位置。如 $\varepsilon^*=3$ 时，选择 C，D 为准目标位置。

STEP 2：试交换 $A \to B$，$B \to A$；$A \to C$，$C \to A$；$A \to D$，$D \to A$；此时 $\lambda = 2$。若只有一对交换使目标函数收敛，则予以认可并以新的布局代替原布局，转 STEP 1 继续处理；若多于一对交换使目标函数收敛，则选择使目标函数值下降最多的一对交换实行实际交换，构成新的布局，转 STEP 1；若没有一对交换可使目标函数收敛，则 $\lambda = \lambda + 1$，转 STEP 3。

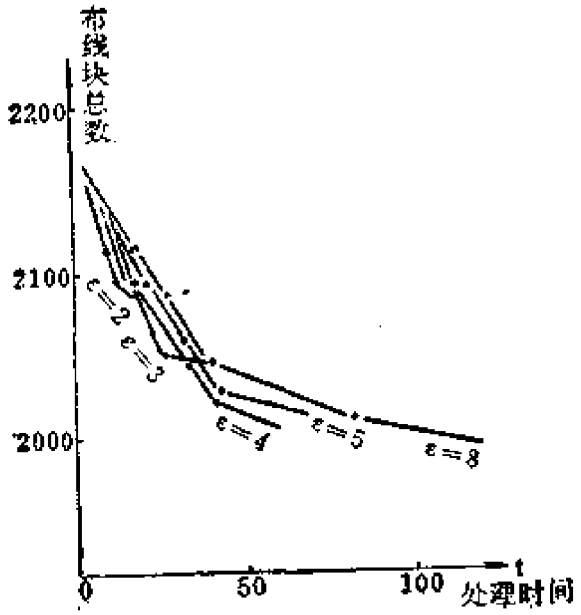
STEP 3：$\lambda = 3$ 时，令 $\varepsilon = 1$，确定 B 位置单元的目标位置 E，F，G，并进行下述三种链式试交换：$\overrightarrow{A \to B \to E}$，$\overrightarrow{A \to B \to F}$，$\overrightarrow{A \to B \to G}$；若其中存在使目标函数收敛的链式交换，则选择其中使目标函数下降最多的一种链式交换方式实行交换，构成新的布局，转 STEP 1。若不存在可行的链式交换，考虑到处理效率，可从中选择使目标函数上升最少的一种链式交换方式进行 $\lambda = 4$ 的试交换。

STEP 4：$\lambda = 4$ 时，设 $\lambda = 3$，$\varepsilon = 1$ 时 $\overrightarrow{A \to B \to E}$ 为最有利的一支，则现在开始进行 $\overrightarrow{A \to B \to E \to N}$，$\overrightarrow{A \to B \to E \to O}$ $\overrightarrow{A \to B \to E \to P}$ 三种链式试交换，若存在可使目标函数收敛的链式交换，可按 STEP 3 中原则处理，若不存在，则如同 STEP3 中处理，选择最有利一支向更深一层搜索，但当 $\lambda \geq \lambda^*$ 时，则令 $\lambda = 3$，$\varepsilon = \varepsilon + 1$ 开始搜索一个新的分支。在本例中即开始试交换 $\overrightarrow{A \to C \to H}$，$\overrightarrow{A \to C \to I}$，$\overrightarrow{A \to C \to J}$，并继续按上述原则处理之。当 $\varepsilon > \varepsilon^*$ 时，该次迭代过程找不到成立条件转 STEP 5。

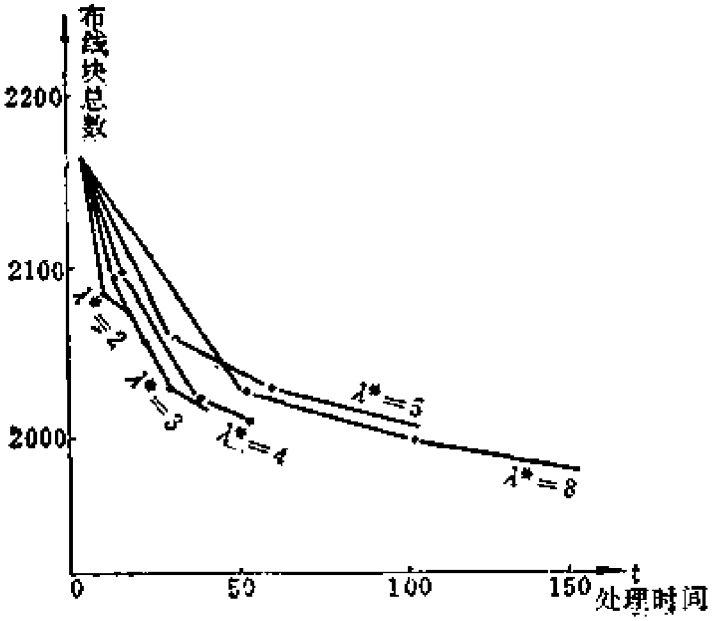
STEP 5：若满足一定的过程结束条件，则迭代改善过程结束，否则对单元 A 作标记，转 STEP 1。

在上述链式交换过程中，若交换链中存在一个“空”位，显然，该交换链实行交换时，一定使目标函数收敛。因此邻接目标位置的空位应优先选作准目标位置。

算法中的过程结束条件可以是当前被标记单元数大于某个预先规定的常数则过程结束。当该常数等于单元总数时，实际上即为

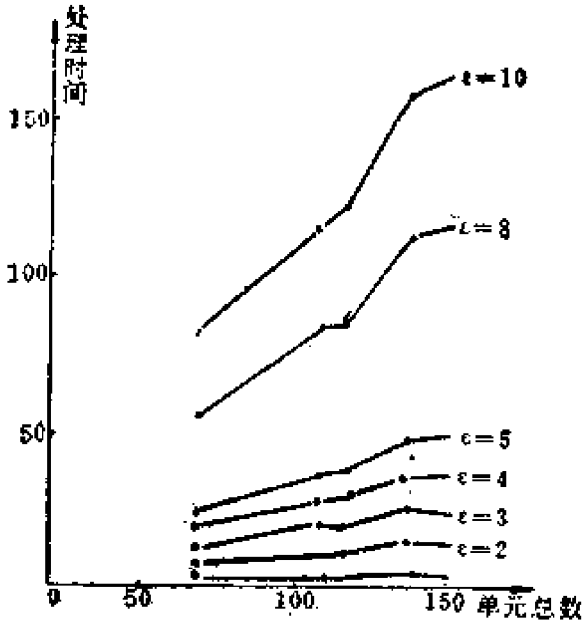


(a)

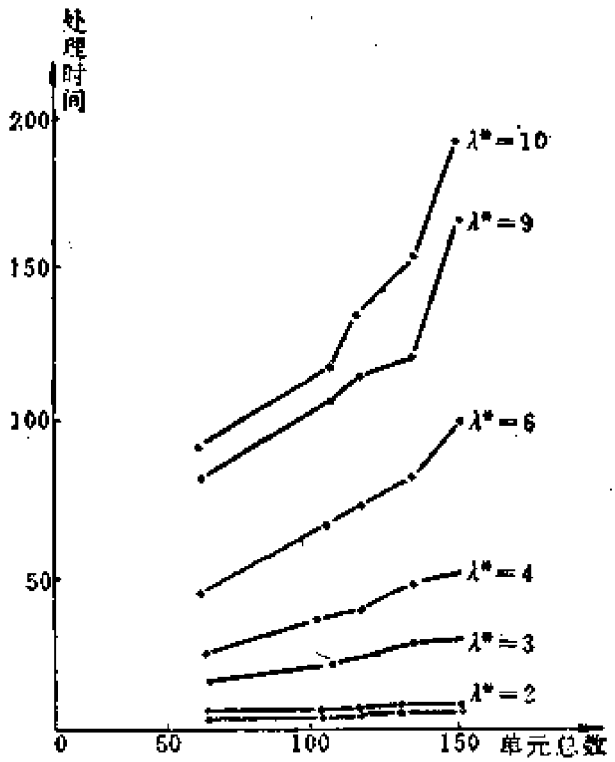


(b)

图 5.15　迭代布局解的精度与 λ^*、ε^* 的关系图



(a)



(b)

图 5.16　处理时间与 λ^*、ε^* 的关系图

穷尽迭代，需要注意的是，每次成功的迭代交换后，应删去所有单元上的标记。

3．实验结果及其评价

(1) 参量 λ^*、ε^* 对布局结果的影响

实验结果表明，迭代改善布局解的精度随 λ^*、ε^* 增大而提高，如图 5.15 所示(注：问题规模为在 11×15 网格(门阵列上)、电路单元总数为 136 (另含 I/O 单元 15 个)、线网总数为 419、每个单元平均与 5.98 个线网相关、每个线网平均与 2.35 个单元相关、线网含单元数为 2～9 个)。

同时，处理时间 T 和 λ^*、ε^* 也成线性递增关系，如图 5.16 所示。图 5.17 并表明问题的规模(单元总数)与处理时间也基本上为线性增长关系，随 λ^*、ε^* 增大，变化的斜率也将增加。$\varepsilon^*>6$，$\lambda^*>5$ 时，对规模较大的问题，处理时间将迅速地增加。

(2) 参量 λ^*、ε^* 的选择

由上述实验结果可以看到，为获得一个满意的解可在一定的处理时间内改变参量 λ^*、ε^* 值而设法获得尽可能多的局部优化解，然而从中选择一个最好的解作为实际解。令 P 为这些局部优化解中出现目标函数值(连线总长等)小于定值 V 的概率。令 T_0 为求得这些局部优化解的总时间，则在时间 T 内求得最佳局部优解目标函数值小于 V 的概率为：

$$\hat{P}=1-(1-P)^{T/T_0}$$

以图 5.15 所示实例在 $T=30$ 分钟内，随机地产生多个初始布局并将 λ^* 固定为 4，选择不同的 ε^*，设 $\hat{p}$ 为不同的 ε^* 参量下求出优于联线总长为 2000 单位长的解的概率。实验结果为：$\hat{p}'_2=0.68$，$\hat{p}'_3=0.71$，$\hat{P}'_4=0.95$，$\hat{p}_5=0.92$，$\hat{p}'_6=0.70$，当 $\varepsilon^*=4$ 时结果最好。可由同样地方法对 λ^* 参量的影响进行分析，将 ε^* 固定为 4，令 $\hat{p}^*$ 为不同 λ^* 下求得优于联线总长为 2000 单位长的解的概率。实验结果为：$\hat{p}^*_2=0.74$，$\hat{p}^*_3=0.93$，$\hat{p}^*_4=0.98$，$\hat{p}^*_5=0.89$，$\hat{p}^*_8=0.75$。当 $\lambda^*=4$ 结果最好，对其他例子进行结果分析，

结论也是类似的。因此以解的质量和处理效率综合来看，在广义力矢量法中 $\varepsilon^*=4, \lambda^*=4$ 为最好的迭代改善参量。从实验结果来看，在 $\varepsilon^*=4$，$\lambda^*=4$ 条件下的力矢量法的改善布局结果优于 $\varepsilon^*=1$，$\lambda^*=2$ 的力矢量成对交换法，$\varepsilon^*=n$，$\lambda^*=2$ 的成对交换法和 $\varepsilon^*=1$，$\lambda^*=\infty$ 的力矢量松弛法，是一种比较好的改善布局方法。

二、多元胞模式改善布局中的单元插入法

在多元胞模式的布局改善中，一般分二步进行。在块的构造完成后，首先通过迭代交换改善各块中单元的构成，使连线在垂直方向的分量之和趋于最小化，然后进行各块内的单元位置构造，而后通过迭代交换确定块内单元的排列，使与该块单元相关的连线在水平方向的分量之和最小化。单元插入法就是块内布局改善的一种方式。设如图 5.17 为一含 N 个单元的多元胞模式的块(行)。该方法从左至右依次将每个单元选作迭代对象，然后将迭代对象分别插入到行内每一对邻接单元之间去（除了迭代对象原来的位置外）。此时在迭代对象原位置与插入位置之间的单元都将朝着迭代对象原位置方向移动一段距离 d，显然 d 等于迭代对象的长度。从图 5.17 可以看到这实际上是一种链式的交换位置，每次试

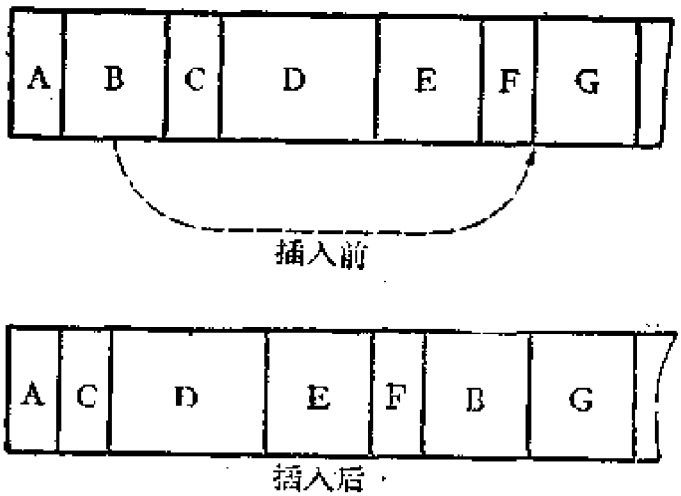


图 5.17　单元插入法

探性的插入后，都必须计算与该块单元相关的连线水平总长的变化情况，选择插入后能使连线水平总长下降最多的位置作为迭代对象的实际插入位置，如果在所有的插入试探中，都不能使连线的水平总长下降，则该单元不改变原位置，而继续进行下一个单元的处理。

单元插入法的计算复杂性为 $O(n^2)$，对于块中含单元多的情况，计算时间将比邻接交换法长很多，贝尔实验室 LTX 系统在块内布局改善中，采用了单元插入法，并和其它块内改善布局算法结合在一起形成一个系统，在应用时可根据具体情况选择相应的算法。

5.4 组合交换方式

布局问题从本质上来讲是一个比二次分配问题还要"复杂"的一个问题，因此，采取单元间一对一换位或链式换位求解，在不少情况下将受到局限，而且使得布局的迭代改善受初始布局影响较大。在图 5.18 中对上述情况进行了说明，从图上可看到，较合理的布局应使 (C, D) 单元与单元 (E, F) 交换位置，然而非常不幸，在几乎所有的对交换或链式交换方法中都难以通过各种试探而达到上述目标(在实际问题中出现上述情况的机会是较少的，大部分情况在初始构造中会得到某种程度的解决)，这也是我们曾在前文中

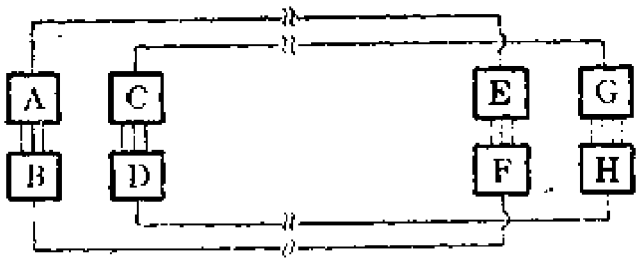


图 5.18 布局中换位求解局限性示意图

提到过的所谓初始布局“构形”对最终结果的影响的一种情况。其次，在对交换和链式交换中每次考虑的几乎都是一对单元之间相互位置关系或一定条件下（链式）的多个单元依次的相互位置关系，缺乏较好的总体考虑。

组合迭代方法在迭代改善布局时，对当前布局中的一个单元子集(满足一定条件)中的元素位置进行组合式的交换，或将二个单元子集的位置进行交换，试图获得较好的最终布局。

一、斯坦伯利(Steinberg)算法[65]

斯坦伯利(steinberg)算法[65]有时又称作最大独立子集算法。

1. 独立子集和最大独立子集

定义：独立子集。在图G中，若存在一个图G的顶点子集g满足：g中的各顶点都不邻接，则称g为图G顶点集的独立子集。

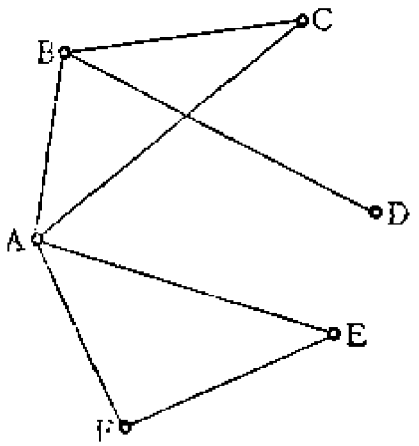


图5.19 独立子集示意图

图5.19中，顶点子集$\{A, D, \}$，$\{C, D, E\}$都是图G顶点集的独立子集。当用线图描述法描述布局问题时，其实际意义为与独立子集对应的单元子集中任何二个单元之间都不存在公共线网；或者也可以讲是，没有一条线网中含有同一独立子集中的二个或二个以上的元素。

定义：最大独立子集。若图G的一个独立子集g满足：如果给g增加任何一个新的元素都将破坏其独立性，则称该独立子集为最大独立子集。

图5.19中，顶点子集$\{C、D\}$为独立子集，但不是最大独立子集，而$\{C、D、E\}$和$\{A、D\}$都是最大独立子集，因为无论给这二个子集增加任何一个新的元素，都将破坏这二个子集的独立性。

2. 斯坦伯利算法主要思想

由于独立子集中任二个元素对应的单元都不属于同一线网，也即独立子集中各元素间不存在互连问题。设独立子集 $C=\{C_1, C_2, \cdots C_m\}$ 对应的位置集为 $P=\{P_1, P_2, \cdots P_m\}$，如果将 C 集中任一元素 C_k，分别安置在 P 中的每一位置上，则与 C_k 相关的连线总长仅与 C_k 当前所在位置有关，而与独立子集中其他元素的安置位置无关。因此，独立子集中各单元位置的最佳位置分配问题就成为一个标准的线性分配问题（可参见 4.10），对于独立子集的 m 个单元对应的 m 个位置，可构造一个 $m\times m$ 的价格矩阵 M，$M(i,j)$ 为单元 i 安置在 j 位置时与单元 i 相关的连线总长。独立子集中单元重新安配问题即可描述为求一个 m 个单元的位置的最佳分配，使与 m 个单元相关的连线总长最短。

实际上位置数往往大于单元数，即位置数 n 大于单元数 m 的情况，此时可设想存在 $n-m$ 个虚拟单元，构成 $n\times n$ 的价格矩阵 N，其中虚拟单元安置在任何位置上的价格为无穷大。

为了提高处理效率，减少所需处理的独立子集的个数，一个自然的想法就是使每次处理的独立子集为最大独立子集。在有些情况下并要求被处理的最大独立子集具有足够多的元素数和在顺序处理的二个集中应当有某些部分重叠即含有一定数量的公共元素等[66]。

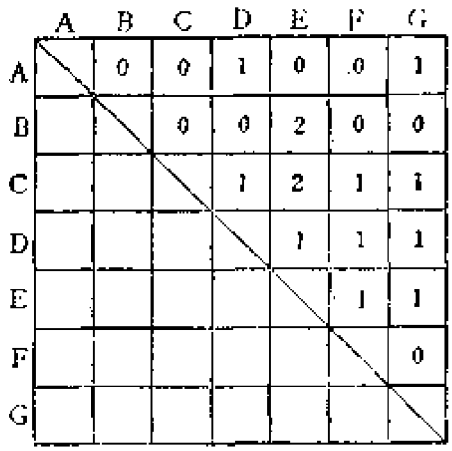
3．求最大独立子集的两种方法

(1) 递推法

① 首先产生所有的两元素的独立子集($k=2$)，k 为独立子集元素数。

在线图描述法中求出这些子集是相当方便的，在对应的单元邻接矩阵中，上三角部分若 $M(i,j)=\phi$，则表明 i 单元和 j 单元是独立的。检索出上三角部分中所有 $M(i,j)=0$ 的单元对就生成了全部二元素的独立子集。

如图 5.20 中{A、B}，{A、C}，{A、E}，{A、F}，{B、C}，{B、D}，{B、F}，{B、G}，{F、G}都是二单元独立子集。



	A	B	C	D	E	F	G
A		0	0	1	0	0	1
B			0	0	2	0	0
C				1	2	1	1
D					1	1	1
E						1	1
F							0
G							

图 5.20　单元邻接矩阵和偶图描述

在偶图描述法中，因 $S(A)=\{S_2\}$，$S(B)=\{S_1,S_3\}$，$S(A)\cap S(B)=\phi$，因此 $\{A,\ B\}$ 为二单元独立子集。同样也可依次求出全部二单元独立子集。

② 对于 $k>2$ 情况，$k+1$ 个元素的独立子集都可从 k 个元素的独立子集中产生。

在 k 个元素的独立子集中选择二个子集 Q_k，和 P_k，使 $Q_k\cap P_k$ 的元素数 $|Q_k\cap P_k|=k-1$，且 $(P_k-Q_k)\cup(Q_k-P_k)$ 为独立子集，则由 Q_k 和 P_k 可构成一个 $k+1$ 个元素的独立子集 Q_{k+1}。Q_{k+1} 中的元素为 Q_k 和 P_k 的交集元素加上 Q_k 和 P_k 中剩下的一对非公共元素即：

$$Q_{k+1}=(Q_k\cap P_k)\cup(Q_k-P_k)\cup(P_k-Q_k)$$

按此原则，可从图 5.21 对应的所有二元素独立子集中产生如下三元素独立子集：

由 $\{A,\ B\}$，$\{A,\ C\}\rightarrow\{A,\ B,\ C\}$

由 $\{A,\ B\}$，$\{A,\ F\}\rightarrow\{A,\ B,\ F\}$

由 $\{B,\ F\}$，$\{B,\ G\}\rightarrow\{B,\ G,\ F\}$

同时，所有元素数为 k 的集合 $(k>2)$，当它们是一个元素数为 $k+1$ 的独立集的子集时都应将其删去。

重复上述过程，直至不可能产生更多元素数的独立子集时为止，即求得了全部最大独立子集。

在上例中这些最大独立子集为：

$\{A,\ B,\ C\}$；$\{A,\ B,\ F\}$；$\{B,\ G,\ F\}$；$\{A,\ E\}$；$\{B,\ D\}$。

(2) 随机抽样法

这种方法可方便地产生顺序处理时所要求的具有一定重叠度的最大独立子集。

① 设所有单元的集合为 M，在上述图例中，$M=\{A,\ B,\ C,$

$D, E, F, G\}$。对于第一个最大独立子集 U_1 的求法为首先从 M 中任取一单元，放入 U_1 集，设取出的单元为 A，则 $U_1=\{A\}$，然后从 M 中删去所有与 U_1 中单元属于同一线网的单元，形成 M'。在讨论的实例中 $M'=\{B, C, E, F\}$，再随机地由 M' 中取出一元素，设为 B，放入 U_1 集，则 $U_1=\{A, B\}$。依上述原则可得到新的 M' 集，$M'=\{C, F\}$。重复上述过程，直至 M' 为空集，得到的 U_1 集即为所要求的第一个最大独立子集 $U_1=\{A, B, C\}$。

② 在生成 U_{i+1} 独立子集时，可以从 $U_i(i\geqslant 1)$ 中随机地选取 n 个单元加入 U_{i+1}，其中 n 值可由所要求的重叠度确定，然后从 M 集产生 M' 集，U_{i+1} 中的其它元素按上述原则从 M' 中逐次随机地选取。值得注意的是，在求得 U_{i+1} 集后，应检查，使其不是以前的集合 $U_j(j<i+1)$ 或 U_j 的子集。一般要求选出的独立子集应有足够多的元素数，当 U_i 元素数太少时则重新选取。

上述随机抽样法过程继续到生成系统要求的 P 个最大独立子集为止。

4. 斯坦伯利算法的主要处理过程

斯坦伯利算法的主要处理过程如下述：

STEP 1：用递推法产生全部最大独立子集。为提高处理效率，在规模较大的问题中可删去元素数较少的一些独立子集，并将这些集合依其元素数大小从大至小排列，令 $N=0$，独立子集数为 M。

STEP 2：令 $N=N+1$，若 $N>M$，stop；否则对第 N 个独立子集的各单元位置进行线性分配。

STEP 3：若目标函数(连线总长)不收敛，转 STEP 2；若目标函数收敛，则对第 N 个独立子集内各单元位置实行再分配。转 STEP 2。

过程的结束条件也可以为通过连续 P 个独立子集的试分配，目标函数都不收敛，则过程终止。

用随机抽样法求最大独立子集的斯坦伯利算法过程是类似

的，本书从略。

二、群法中子群位置的迭代改善

群法中经过结群处理生成一个结群二元树后，可以认为，一个结群二元树的具体描述对应着一种行式的单元布局。因此，群法中子群位置的迭代改善又称行式布局。

在结群二元树上，与每一个顶点相连的二个子群表示了它们彼此间的联结关系将比它们与其他元素的联结关系紧密，对应于布局，则表示它们的安置位置应尽量相邻，而二个子群绕其顶点转动(即互换位置)将在不破坏结群关系的情况下对应着一个新的行式布局。对于 n 个元素的结群二元树，树的顶点数为 $n-1$，不破坏结群关系的行式布局的可能个数为 $2^{(n-2)}$。当 n 足够大时，我们将不可能穷尽所有的可能以寻求一个最佳的布局。在实际处理时，往往采用一些近似的解法。

1. 自底向上的行式布局

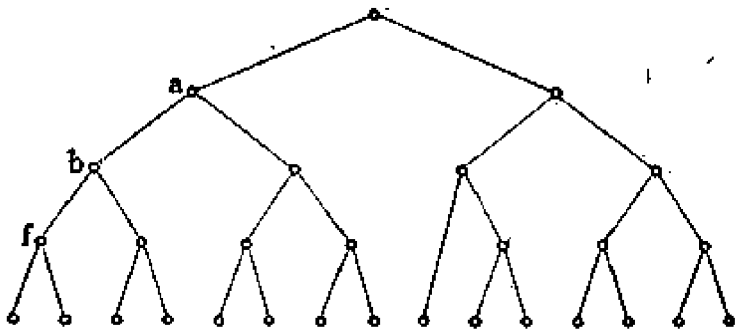


图 5.21 二元树

定义：令二元树各边权重为 1，二元树各顶点到达端点(单元)的最远距离称作顶点的级。图 5.21 中，顶点 a、b、f 的级分别为 3、2、1。

自底向上的行式布局从顶点级为 1 的顶点出发，确定与该顶点相连的两个子树的行式布局，当所有顶点级为 i 的顶点处理完

后，再处理 $i+1$ 级的顶点，直至全部顶点处理完毕，完成了所有单元的行式布局。

在每个顶点的处理中，设与该顶点相连的两子树分别有 m 个和 n 个单元，且子树内单元的邻接关系已经确定。由于二个子树中的单元不能互相混合，因此它们在不改变子树内单元的邻接关系条件下只能有四种方式排列：

$[(1, 2, 3, \cdots, m), (1, 2, 3, \cdots, n)]$；$[(1, 2, 3, \cdots, m), (n, \cdots, 3, 2, 1)]$；$[(m, \cdots, 3, 2, 1), (1, 2, 3, \cdots, n)]$；$[(m, \cdots, 3, 2, 1), (n, \cdots, 3, 2, 1)]$。

算法从中以一定的目标函数(如连线总长最短)选择其中一种排列为两子树的行式布局结果。

上述算法是比较简捷的。它的问题主要是每次从四种排列中选优时，考虑的仅仅是两个子树之间的关系，而无法考虑它们与其他单元的关系，也即缺乏总体的优化考虑。这是影响其布局精度的主要原因。

2. 自顶向下的行式布局

设结群二元树顶点的最高级数为 R，自顶向下的行式布局从顶点级为 R 的顶点出发，确定与该顶点相连的两个子树的行式布局排列。当所有 i 级顶点处理完后再处理 $i-1$ 级的顶点，直至处理完树上所有的顶点为止。

在每个顶点的处理中(从理论上来讲，与 R 级顶点相连的两个子树的行式布局排列是任意的)，当忽略子树内的单元位置排列时只存在二种子树的可能排列。即一个子树排列在右边，另一个排列在左边或一个子树排列左边而另一个排列在右边。此时，若以连线总长最短为目标函数，则设全部的互连都连到子树的中点。这虽然不考虑也无法考虑每个子树内的排列，但必须考虑每个子树的尺寸。子树的尺寸是指子树所含单元尺寸的和，并以此为基础计算在每种可能的排列中两个子树和其它子树的互连线总长。图 5.22 为两个子树 (m 和 n) 在两种可能排列中互连总长的计算

$L'_1=C_m\times 0.5S_m+C_n\times(S_m+0.5S_n)$

$L'_2=C_n\times 0.5S_n+C_m\times(S_n+0.5S_m)$

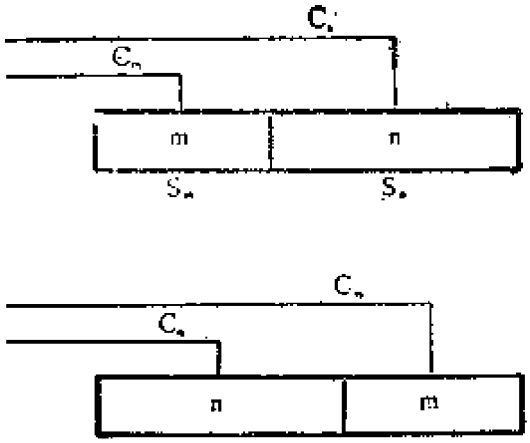


图 5.22　两子树互连总长计算示意图

注：C_m、C_n 为子树(m, n)与其它子树的互连线数

情况。

选择具有较小 L' 值的排列作为 m 和 n 子树行式布局的结果然后再分别处理下一级顶点，决定 m 和 n 子树内排列问题，直至确定每个单元的排列为止。

自顶向下行式布局在每个顶点处理时较好地考虑了子树间的相互关系，虽然在连线长计算时作了一些近似的处理，其结果一般都比自底向上的行式布局的结果为好。

3．ε-算法

1976 年 GOTO 运用分支限界法求行式布局的局部优化解[67]，其目标为寻找一个单元的行式排列，使该排列下布线的最大密度最小化。在实际应用背景下，这与使芯片面积最小化或使布局具有较好的可布性存在着较好的对应关系。该算法可使解的精度控制在 $1+\varepsilon$ 倍最佳解的范围之内，随着 ε 的减小将使处理效率急剧降低。这个算法也称为 ε-算法，对于“单元”数较少的情况，如门阵列或多元胞“块”的安置(此时，每个“块”看作是一个“单元”)具有较好的实用意义。

5.5 改善布局方法评价

一、改善布局方法评价的复杂性

由于布局问题的复杂性，对各种不同的布局方法，尤其是改善布局方法的严格评价也带来相当的困难。常常出现这样的情况，一种方法对于有些问题取得了相当满意的结果，而对另一些问题结果却不十分理想，甚至结果相当差。同时，另一种算法虽然一般结果不十分好，但对这一类问题却可获得较好的结果。对于同一个问题，改善布局的方法的结果处理往往又与其初始布局构形有关，这样也使改善布局方法的评价带来更多的困难。因此在评价时，必须使用一定数量的与实际问题相同的（或非常类似的）问题作为试验对象，而使用随便选取的假想例子常常会使人得到错误的印象。例如对于一类不真实的随便选取的问题，成对交换法看起来是相当好的，初始的布局构形似乎对其影响较小。此外对算法采用的模型合理性，算法目标与实际目标的一致性，算法处理过程逻辑合理性以及算法处理效率给出理论上的评价也是十分重要的。

二、改善布局目标函数的评价

从改善布局的算法目标来看，整个设计要求在这个阶段中能在满足布线要求的前提下使布图设计得到的芯片面积最小化（多元胞设计模式，任意元胞设计模式等）或在限定的芯片面积内满足可布性的要求(门阵列设计模式)。然而由于这些目标实现时的复杂性，不少算法以连线总长最短化或其局部优化解(合矢量和最小化，中值位置偏差和最小化等)作为算法的实际目标。在这一类算法中，广义力矢量松弛法是值得注意的一种有效算法，但必须同时指出，尽管该实际目标和设计目标有较好的对应性，但它们在本质上并不是完全一致的。目前的实验表明，当连线总长偏离其最

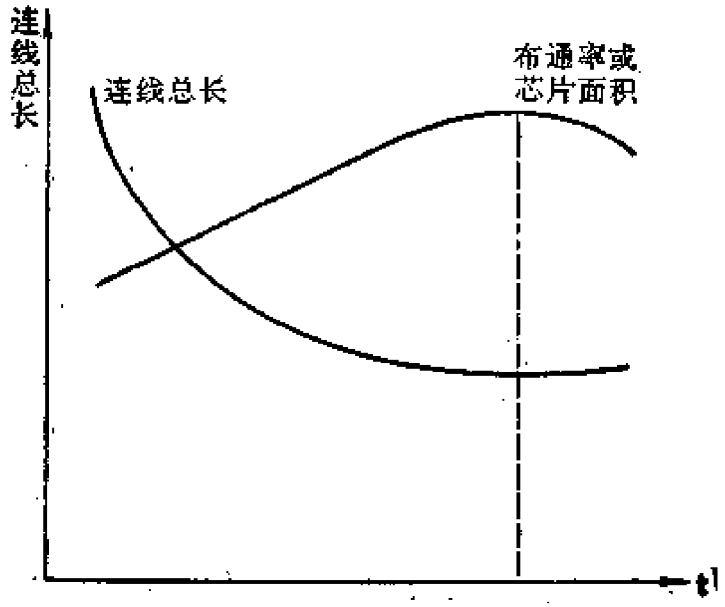


图 5.23　连线总长与布通率或芯片面积对应关系

佳值较大时，连线总长的下降与布通率的提高或芯片面积的减少有着较好的对应关系，但当连线总长下降到接近最佳值的一个临界值附近时，它们之间不仅对应关系很差，而且甚至往往向相反的方向变化（图 5.23）。在这方面以可布性为目标的最小割算法就具有前一类算法所不具备的优点，并得到人们的重视 [76]。此外一个自然的想法是，在改善布局的前一阶段（即连线总长达到临界值前）采用连线总长最小化作为目标函数，由于该目标具有易于度量，总体性较好的优点，可使算法的结果精度改进和处理效率比较令人满意，而当连线总长接近临界值或到达临界值时（取决于收敛的速度）采用其它目标函数（如可布性目标，或芯片面积最小化目标）对布局进一步进行改善以获得符合设计要求的结果。

三、改善布局处理方式评价

从改善布局的处理方式来看，由于布局问题本质上是一个比二次分配问题更“复杂”的问题，因此一些最佳化算法（如分支限界法等）[68]、[69]，在实际应用时由于处理效率低等原因而对问题的规模将有所限制，实际算法中大量采用的是迭代改善的处理方

式。在迭代改善处理时，目前常用的是对式交换或链式交换。这些交换方式处理原则比较简单，对很多问题可取得较好的结果。需要指出的是，即使对于二次分配问题来讲，对交换或链式交换方式将使进一步提高解的精度受到限制。从原则上来讲，链式交换方式将优于对式交换，并已得到实验证明。文献[36]证明了当允许交换链长度为4($\lambda^*=4$)时，从解的精度和处理效率的综合指标来看是令人最为满意的。同样，组合交换方式从原则上来讲将优于链式交换，尤其是引进结群概念(群体大小若为1个单元时即为对式交换或链式交换）的组合迭代将可能获得更高精度的解。因此，在布局改善处理中引入结群的概念，并适当地用群体组合迭代的方式作为对式或链式交换的补充，将对改善布局的处理结果质量是很有益的。

四、初始布局和改善布局的关系

在目前的布局算法中，不少算法都把布局过程划分为二个子过程，即初始布局与改善布局。在第四章中我们已经初步讨论了初始布局的评价问题，特别提到了初始布局构形对最终布局结果的影响。为解决这个问题，原则上还有一个解决途径是使改善布局的算法对初始构形的灵敏度降低。这方面的深入研究自然也是令人感兴趣的，有些文献建议把初始布局和改善布局联系起来构成一个迭代过程，即用处理效率作为代价来换取解的合理性。

五、改善布局算法和布图设计模式的关系

最后需要指出的是各种改善布局算法和布图设计模式之间的关系。我们知道，目前比较成熟的对式交换或链式交换方式对比较规则的布图模式(门阵列、多元胞)适应性较好，而随着布图模式的不规则性的增加，这些算法的适应性将下降，甚至带来很大的困难。如在任意元胞模式中如何应用对式交换将存在许多困难，应用链式交换的困难将更大。而连线总长最短的目标函数也将随布

图模式不规则性的增加而实际效果下降。因此，目前在任意元胞的布局中更倾向于使用可布性目标函数(如最小割算法)。另一方面，虽然在描述不规则布图模式的布局图模型方面作了不少有益的研究。但在如何迭代改善及其评价方面仍存在不少问题。从原则上来讲，影响单元布局的二个主要因素是：

① 单元间的联结关系。

② 各个单元的几何形状。

在极其规则的布图设计模式(如门阵列模式等)中，由于各单元大小，形状(都为矩形)相同，因此在布局中基本上只需考虑单元间的联结关系就可以了。而随着布图设计模式的不规则性的增加，单元形状，大小对布局的影响也随之增加。有些文献认为，在任意元胞设计模式中，单元几何形状的影响将是主要的，退一步讲至少是非常重要的。历史上，大部份布局算法都是基于单元联结关系的分析而考虑的，对不规则的布图设计模式或者是根本不适用，或者是适用性较差。由于 VLSI 发展的需要和分级设计技术的应用，对于不规则的布图设计模式的布局算法的研究正在并将进一步引起人们愈来愈大的关注，发展一种综合考虑单元联结关系和单元几何形状的布局算法必将会引起人们很大的兴趣。

综上所述，布局算法尤其是改善布局算法的研究面临着许多待解决的问题，需要更多的创造性劳动，这方面的工作也必将随着 LSI/VLSI 的发展得到不断的发展。

第六章　面向线网的布线方法

6.1 引　　言

通过电路的描述和单元的布局安置，电路中单元的联接关系和线网的性质已得到确定，线网中的接点位置也已得到确定或初步的确定（布线过程中还可能进行等价接点的选择和单元位置的微调整）。下一步设计的主要任务是如何在给定的布局条件下，实现电路所需的互连。

一、布线设计的目标

在布图设计中，布线设计的目标可描述为：根据电路的连接关系描述（连接表），在满足设计、工艺规则的要求和满足电学性能的要求的条件下，在限定区域（面积、形状、层次等）内100% 地完成所需的互连。或者是在100% 完成所需互连时，使所需的芯片面积最小化。同时要求尽可能优化其设计结果（如连线长度最小化，通孔数最小化等）和具有足够高的设计效率。

显然，这是一个极其困难的目标，尤其是对于布图密度高、互连关系复杂的电路设计更是如此。

二、布线设计方法分类

电路的布线设计方法大致可分为两大类：

① 面向线网的布线设计方法。主要以线网的布线问题作为考虑对象。

② 面向布线区域的布线设计方法。主要以布线区域内的布线问题作为考虑对象。

二者虽有一定的联系，但在设计思想和设计方法上也有明显的不同，有着各自的研究课题。本章将着重讨论面向线网的布线设计方法。在下一章我们将讨论面向布线区域的布线设计方法。

三、面向线网的布线设计方法面临的问题

在第一章中，我们知道在不存在障碍的情况下，在平面上实现一条线网的布线一般并不困难，但对多点线网（即一条线网联结着多于2个接点的情况），当接点数足够多时，如需求其连线总长最短的布线方案则是一个复杂的难题。而在实际问题中，线网数一般相当多，情况将更加复杂。除了必须考虑线网需满足的电学、工艺要求外，当第i条线网布线时，为了不与同平面的已布线交叉，不仅需要考虑线网自身各接点间的关系，而且还必须考虑已布线的影响及对今后其它线网布线的影响。此时，是否存在一条实现互连的路径，如何找到互连的路经，如何找到一条最短的路经，或找到一条对今后布线最有利的路经，都成为相当困难的问题。很有兴趣的是，这个问题非常类似于迷宫问题。古代的建筑大师们曾创造出一些至今令人叹为观止的迷宫。今天人们如何借助电子计算机在芯片上这些比历史上曾经有过的最复杂的迷宫还要复杂千百倍的“迷宫”中找到一条通向成功的道路，是我们布图设计自动化工作者面临的又一个困难而又引人入胜的问题。

6.2 李氏算法

在任何布线问题中，首先必须考虑的约束条件是实际布线的几何尺寸。这里包含着二种条件：即芯片制造工艺所允许的布线最小宽度W和线间必须保持的最小间隔S。比较方便的办法是把这二项要求合并为一个单一的从线中心到相邻另一条线中心的距离要求$d=W+S$。如果设想布线区域上存在着一个间距为d的矩形格网，且接点都在格网中，则只要使所有布线都沿着格网走，每

个格网在一个方向上只允许布一条连线，我们自然就保证了布线结果一定将满足上述约束条件。目前芯片上线宽和线间距一般在10μ左右。

使用矩形格网后，布线实际上是沿着曼哈顿路径实现的(参看4.1节)，其优点是易于描述和计算效率高（曼哈顿距离的计算比欧几里得距离的计算可快一个数量级），缺点是不再允许或需特殊处理才能走斜线或曲线，这样使连线长度最短化受到一定的限制。目前的布线设计方法中广泛采用的是上述矩形网格概念。

一、李氏算法的一般描述

1. 李氏算法的基本思想

李氏算法（Lee—More Algorithm)[80] 实际上是在运筹学和图论中广泛使用的最小路径算法在布图设计中的一种应用算法。它的算法思想也可描述为波传播过程的模拟。在一个存在障碍的湖面上，若需寻找连接点A和B之间的最短路径，可以在点A处投下一个小石子，然而观察所引起的水波的传播情况。假定水波的传播过程中能量没有损失，则当遇到障碍时，波发生绕射，最先到达目标点的波前所经过的路径必是一条最短路径，而且只要二点间存在通路，则从点A开始扩展传播的水波一定将波传播到点B，也即只要通路存在就一定能找到这条通路。这个过程可以形象地在计算机中进行模拟。

2. 李氏算法的布线过程

(1) 数据准备

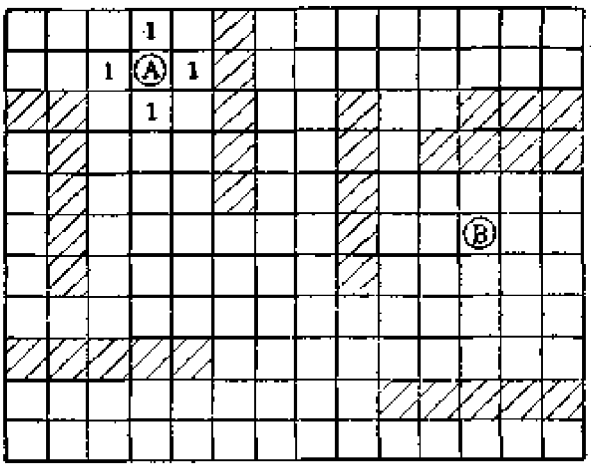
首先在存贮器中建立一个二维数组$M(I, J)$，使芯片上每个矩形网格映射为该数组的一个元素。数组的维界分别表示了芯片的长和宽，其下标值和网格的坐标（以$d=W+S$为单位）相对应。在一条线网实现布线时，将不允许布线的区域(如待布线网接点等）和已布线段占用的网格作上标志。算法中并可定义二个波前场，一个为当前波前场WF_1，记录当前波传播所到达的位置。

另一个为上一个波前场 WF_0，记录波本次扩展前的波前位置。初始时，WF_0，WF_1 都为空集。$M(I, J)$ 数组中未占用的元素值（下面也称格值）都为 0。

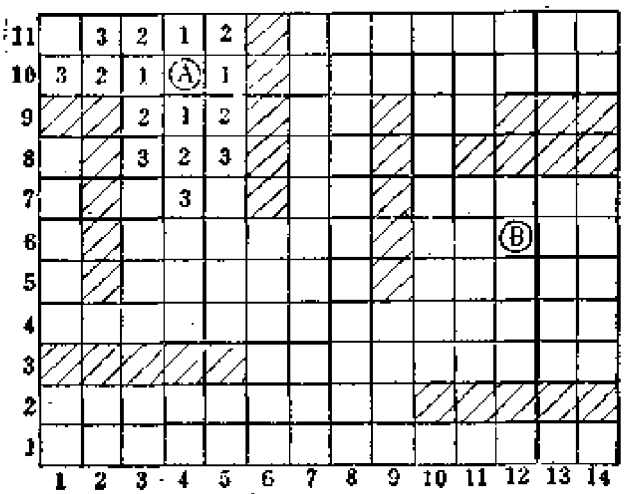
(2) 扩展过程

过程开始时，任选 A、B 中一点为波源点，并将其送入 WF_0，同时在 $M(I, J)$ 数组相应位置作起点标志。另一点定义为目标点，并在 $M(I, J)$ 相应位置作目标点标志。

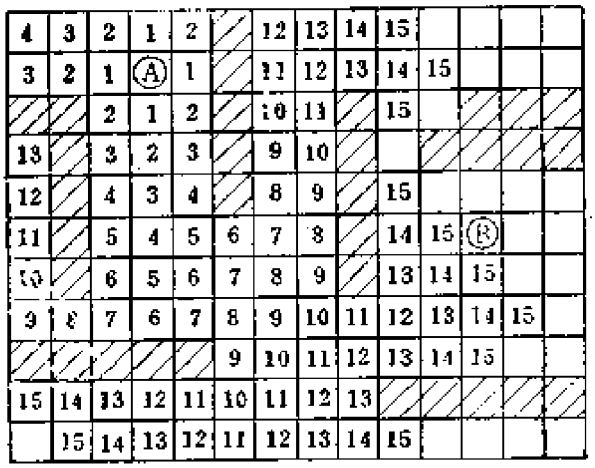
当 WF_0 非空时，顺次将其中的元素取出，并向邻格扩展。扩展后的格值为 WF_0 元素格值加 1（波源点格值为 0），也即表示了将扩展到的网格离波源的最小曼哈顿距离。记该格值为 R，当其邻



(a) 第一步

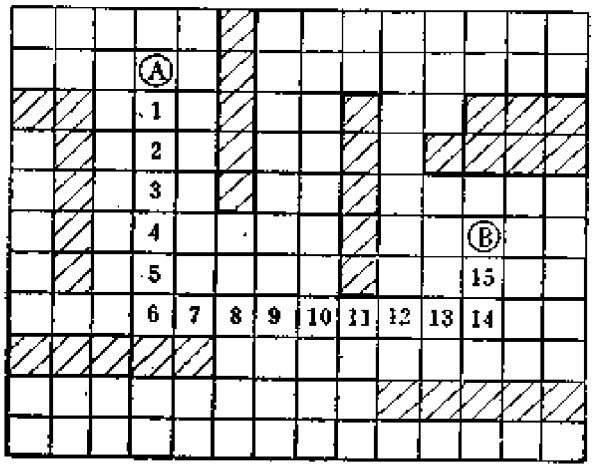


(b) 第三步



(c)

第15步



(d)

回找过程

图 6.1　李氏算法扩展过程示意图

格不是障碍或原格值大于R时，将R填入该邻格，并将此邻格送入WF_1。当WF_0中元素处理完毕后，将WF_1元素全部送入WF_0并清WF_1场。扩展时若其中一邻格为目标点，则扩展过程结束，说明已找到一条最短的连通路径。

扩展到第三步时，WF_0为 {$M(9, 3)$, $M(10, 2)$, $M(11, 3)$, $M(8, 4)$, $M(9, 5)$, $M(11, 5)$}。WF_1为 {$M(8, 3)$, $M(8, 5)$, $M(7, 4)$, $M(10, 1)$, $M(11, 2)$}。（参见图 6.1(a)、6.1(b)、6.1(c)）。

若尚未到达目标点，而WF_1场为空，则该线网布线失败，即所

有的路径都已被封死，不存在任何连接 A、B 的可行路径。

(3) 回找过程

如果扩展过程中到达了目标点，则开始回找处理，即确定布线路径。图 6.1(d) 中到达 B 时的格值为 $k(k=16)$ 则 B 的邻格中至少有一格的格值为 $k-1$。同理 $k-1$ 格值的网格，其邻格中也至少有一格值为 $k-2$ 的网格。如此递推，则很容易由 B 返找回 A 并确定了其布线路径(参见图 6.1(d))。

在实际问题中，尤其是在已布线数量尚少的情况中，连接 A、B 的最短路径往往并不难找。从连接 A、B 的需要来讲，这些路径在理论上是等价的。但考虑到布线质量及对后布线的影响，在实际处理时一般需作专门的考虑。一种考虑是使一条路径的拐弯数尽可能少。在图 6.1(d) 中，格值为 10 处，存在着多种选择，但从使拐弯数尽可能少的考虑出发，应记录先前的回找方向，尽可能不改变回找的方向。在图例中应选择 $M(4, 7)$ 为下一个格网。自然，在回找时，也会出现路径必须拐弯的情况。如果有二种选取的可能，为使布线比较均匀，一个有用的办法是存上一张"罗盘"优先数表，例如北—东—西—南。如有可能应先选北；如果不行，再选东；等等。在回找开始时，就可用此法选取回找方向。主要的是，如有可能，应选取相同的方向。另一个有用的办法是，当存在二种候选方向时，选取离芯片中心远的网格为实际回找路径。这时，将利于把一般芯片中心区布线密度高的矛盾适当予以缓和。一些更细致的方法将在下几节中进一步予以介绍。

3. 李氏算法的特点

从上面的介绍可以看到，李氏算法在解决二接点的互连问题时，只要接点间存在着通路，则无论其间障碍何等复杂也一定能找到其中最短的的一条路径，换言之，李氏算法具有极强的绕障能力且保证当前连线的最短化。由于布线区域的格网化，使算法可较容易地描述布线区域各处的特性，并易于模拟各种具体的工艺要求。因此，李氏算法的另一重要特点是具有很好的适应性。在 P.

C.B.(印刷电路板)和IC(集成电路)的布线设计中，李氏法得到了广泛的应用。

需要指出的是，李算法的算法目标是使当前连线长度最短化。这一点与使整个布线连线总长最短化并不一定存在着对应关系。如图 6.2 所示，线网 1 和线网 2 以不同的顺序用李算法实现布线时，分别实现了当前连线长度最短的目标，但连线总长却可相差较大。自然，它与可布性目标的对应情况在复杂的问题中将更差。因此，常常出现当前布线把待布线接点"围死"的情况而影响布通率(或布线成功率，定义为：已布成布线的线数/总线数)。在复杂的布线问题中，李算法布线成功率较低。其次，由于布线区域的格网化和网格扩展方式的布线过程使李氏法所需的存贮量较大，在布线区域相当大时，其所需存贮量甚至是令人难以忍受的，并使布线效率较低。因此在规模相当大的布线问题中，用李氏法直接布线将受到很多限制。

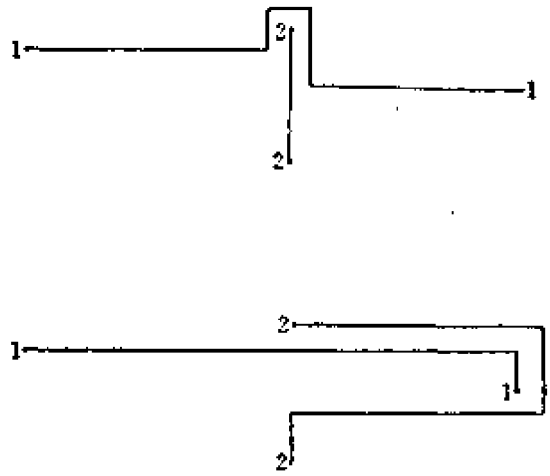


图 6·2　两线网用不同顺序布线后的情况

二、李氏算法中提高布线成功率的方法

从图 6.2 我们可以看到，布线顺序将对连线总长产生很大的影响，显然也将严重地影响布线成功率，但这一点可用在布线前作

适当的顺序处理来解决，我们将在下面专门讨论这个问题。下面将就布线过程中如何提高布线成功率的其他方法作一点讨论。

1. “逼近障碍”的布线方法

(1) “逼近障碍”布线原则

当二点间存在着多条互连路径时，哪条路径将使将来的布线较易于进行这实在是一个难题。一般讲，如果当前连线尽量短，即占用布线区域尽量少将有利于将来的布线，这一点由李氏算法中已可得到保证。但仅仅这样一个启发性原则仍是不够的。另一个合理的想法是使当前连线耗费的格网线段尽可能少[28]，图6.3中说明了这一点。图中每个线段交点表示了一个网格，线段表示了网格间存在的通路。设在图中6.3(a)中已布了二条线a和b，现在要求把C和D连接起来。一般布线时，仅考虑已占的网格数。现在我们不光考虑网格的使用情况，而且也考虑可利用的格网线段(即网格间的通路)的情况。由图6.3(b)可以看到，当a、b布线后，实际上除a、b已占用的格网线段外，还有23条线段也已不能使用了，因为它们直接与a，b线相连。

此外，CD连线后，自然也会使一部分线段不能再用作其它线

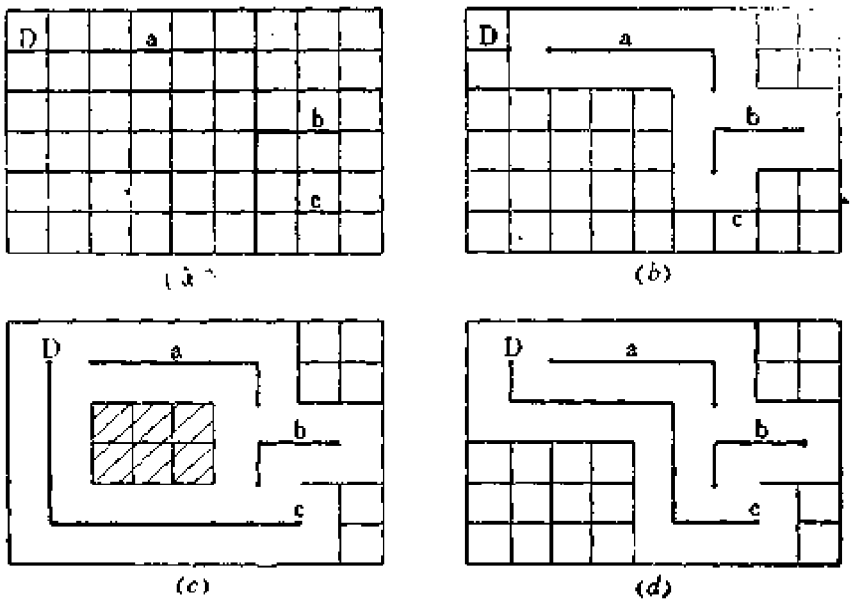


图6.3 减少连线耗费格网线段的布线示意图

网的布线。在图 6.3(c) 中，选取了一条连接 CD 最短的路径，且该路径仅有一个拐弯点，路径本身占用 10 条线段，同时还使 23 条与该路径相连的线段不能用于其他线网的布线。但是，若选取图 6.3(d)所示 CD 的连接路径，则路径本身仍占 10 条线段，而且只使 15 条线段变成无用线段。显然，一般讲，图 6.3(d) 所选取的布线路径将较有利于今后其他线网的布线。值得注意的是，在图 6.3(c) 中阴影区内 17 条线段实际上已无法被该区域内的线网用作布线，而图 6.3(d) 中避免了这一点，上述布线原则也可称为“逼近障碍布线原则”。

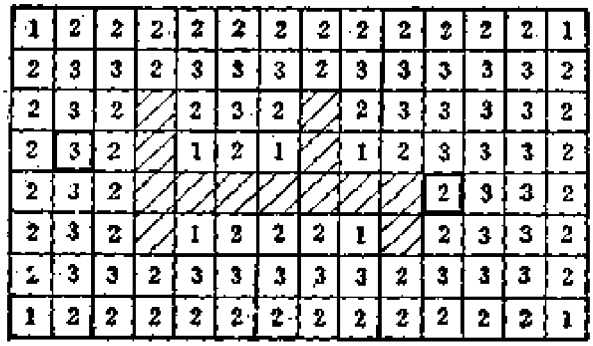
(2) “逼近障碍”布线原则在李氏算法中的实现

在李氏算法中要实现上述原则是比较容易的。可对每个网格定义一个权重 W。对于每一个尚未利用的网格，邻接于这个网格(图 6.3 中的线段交点)的可用网格线段数 $P(0\sim4)$。在布线时使用了这个网格就将使这些线段都不能再用于其他线网的布线，除了布线路径所占线段外，实际消去的可用线段数为 $p-1$。因此可定义每个网格的权重 $W=p-1$。其实际意义是，W 愈小，则利用该格布线的损失愈小（消去的可用线段数愈少）。在实际算法中，不必同时在各网格中存贮权信息，在扩展过程中到达各个网格时，可由其邻接网络的性质计算出其当前权重。为了叙述方便，在图 6.4(a) 中我们给出了一个当前布线时的各网格权重的阵列图。

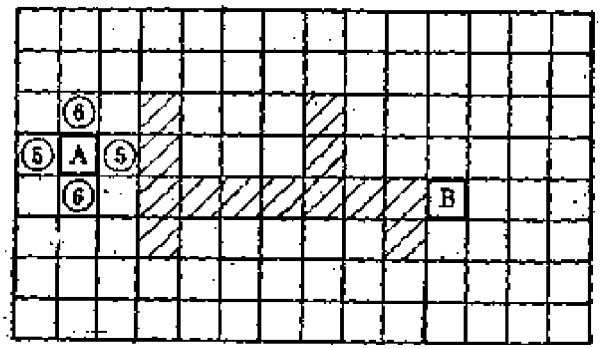
图 6.4(b)～图 6.4(e) 表示了在加权网格上李氏算法的扩展过程。首先以连线的一个接点(图中为 A) 作起始点，按李氏算法的扩展规则向邻格扩展，当该邻格非障碍且格值为 0 或大于其将赋入的格值时，将一个加权的格值赋入该网格。此时，该格值等于 A 点网格权重加该格的网格权重，令为 S。图 6.4(b)中打圈的网格为第一步扩展后的波前。然而由波前的每一个网格继续向邻格扩展，并赋波扩展的网格的格值为 $S+W$(W 为该格权重)。这个过程可以一直继续下去，直至到达目标点为止。

在图 6.4(d) 中，第 11 步扩展后，形成图中打圈的网格组成的

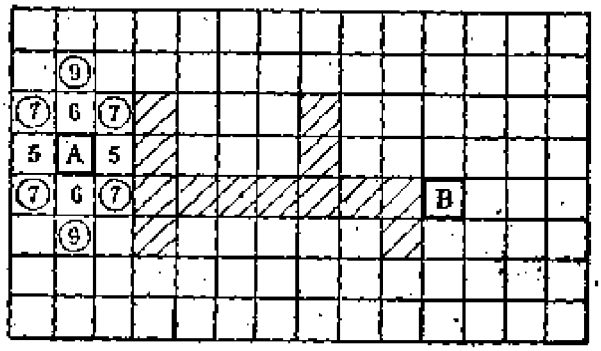
波前。当(3.9)网格(格值为 25) 扩展时，邻格 (2.9) 的原格值为 29，大于 25+3。则(3, 9)扩展后，(2, 9) 网格格值变为 28，并作为新的波前元素继续扩展。这在实际上是意味着：存在着一条形式上更短的路径，并将其代替原来的路径。在图 6.4(*d*) 中即为用 (2, 5)→(3, 5)→(3, 9)→(2, 9) 路径代替(2, 5)→(2, 9) 的原路



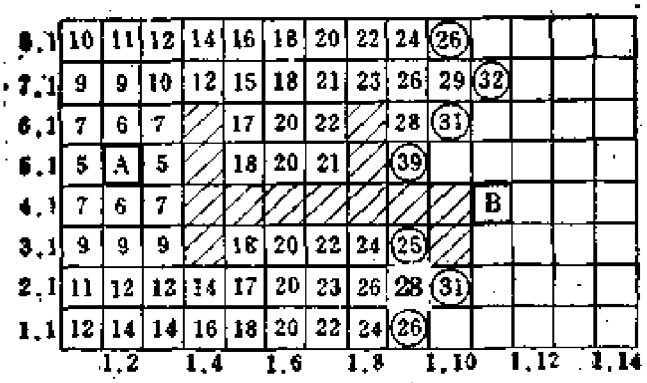
(*a*) 阵列图



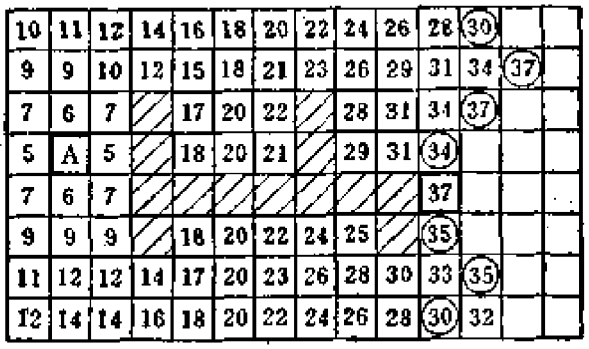
(*b*) 第一步



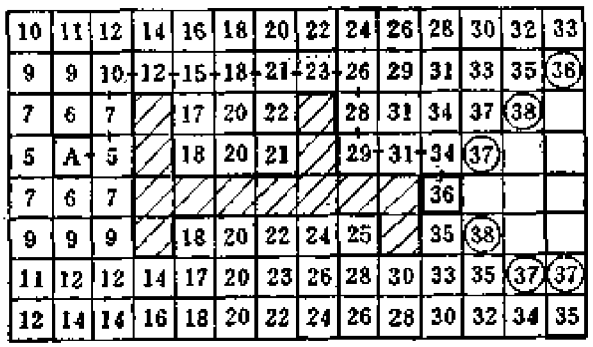
(*c*) 第二步



(d) 第11步



(e) 第14步 〈图为13步时结果〉



(f) 最终路径

图 6.4 在加权网格上李氏算法扩展过程图

径(注意:前者实际路径长度大于后者)。

在图 6.4(e) 中,第 14 步的扩展中,设波前中(1, 11)先扩展,然后(3, 11)再扩展,而其它波前元素还未扩展。此时由于(3, 11)网格的扩展,首先到达了目的点,格值为 35 + 2 = 37。但是,在加权的李氏算法扩展过程中,首先到达的路径不一定是形式上最短的路径,而必须检查一下是否还可能存在更佳的路径。因此扩展

过程必须继续进行下去，直至所有的波前元素格值都大于可行解中目的点的格值为止。在图 6.4 的实例中，最终的结果目的点 B 的格值应为 36。

回找过程和一般李氏法基本相同。图 6.4(f) 中，从 B 点开始，寻找能得到扩展格值为 36 的邻格，即其本身的格值加上网格 B 的权重等于 36 的那个邻格。于是找到了格值为 34 的那个网格；等等。最后的结果如图 6.4(f) 所示。

从图 6.4(f) 可以看到：

① 采取加权李氏算法后，得到的路径将是形式上最短的，而并不一定是实际曼哈顿最短路径。设想目的点在(2.9)处，即可出现这种情况。

② 上述加权方法并不保证找到一条“最逼近障碍”的布线路径，因此这种方法也可看作是一种实际路径长度最短化原则和“逼近障碍”布线原则的折衷考虑(请注意：“逼近障碍”布线原则与其他一些布线原则一样，并不是一种“完美”的原则，从可布性来讲，也存在着反例)。

2. Q 函数法

文献[81]中采用了另一种改善可布性的启发性原则。在扩展时，采取一般李氏算法，仅在回找过程中并出现二个候选回找网格

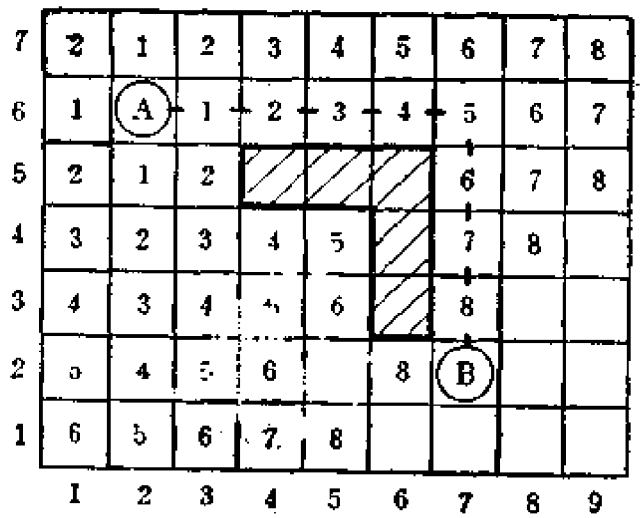


图 6.5　改善可布性启发式布线示意图

时，对候选网格定义一个Q函数。如图6.5中，目的点B回找有二个候选网格(2, 6)和(3, 7)。Q函数定义为该格8个邻格(包括对角的4个邻格)中障碍（已布线等）网格的数量。(2, 6)网格的Q值为1，(3, 7)网格的Q值为2时，选择Q值大的候选网格作为继续回找的网格。该方法处理方便，且很多情况下效果较好。

3. “迭代布线”法

1980年J.Vintr提出一种“迭代布线”[82]算法以提高李氏算法的布线成功率。对于实际上存在布线路径，而因其他布线的影响布线失败者，其原因往往是：该线的接点被已布线“围死”或布线通路被已布线封死，如图6.6所示。

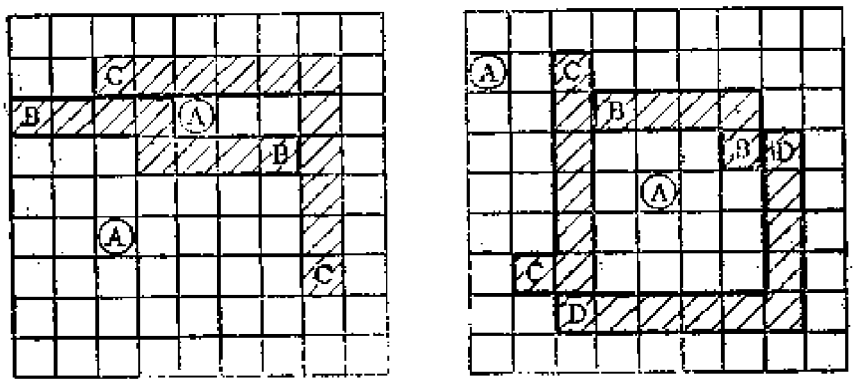


图6.6 布线通路或线的接点被封死的情况

我们知道，李氏算法在布线时实际是串行的，即一条线布完后再布另一条。这样，先布的线不可避免地将对后布线产生影响。一种自然的想法是，为了避免已布线封死待布线接点或封死待布线布线路径，应力图使布线过程变成是一种“并行”的布线过程，而减少布线顺序对布线完成率的影响。在Vintr's算法中，每条连线的互连不是一次完成，而分几次完成，如分5次完成。每次仅完成各连线的1/5，并且使后续的布线根据当前的布线结果对先前的结果进行迭代改善。

Vintr's算法的主要思想可说明如下：

第一次应用李氏算法依一定顺序对每条连线进行布线。但每条连线布完后仅承认其由接点引出的长度分别为1/10总长的部

分，其余部分删去。因此，第一次布线完成后，出现的结果是每个接点都长出了一条“小尾巴”。当出现无法实现布线的连线时，及时地对其他线进行必要的调整。

第二次仍用李氏算法依一定顺序(顺序也可适当地变化)对每条连线进行布线。每条连线布线前，先将其原布线结果全部删去，然后重布。布完后，承认其接点引出的2/10总长的部分，其余部分删去。注意，此时的布线结果中，由接点引出的1/10总长部分可能与原结果相同，也完全可能与原结果不同，即已根据当前布线结果进行了修改。这样，各接点处引出的“小尾巴”又分别增长了1/10。若出现布线失败的连线，及时地对其他线进行调整。

重复上述过程共5次，就完成了整个布线过程。

上述算法显然将使布线效率降低，但由于李氏算法布线效率随着布线区域内已占用网格数的增加而提高，因此效率的降低远不如想象的那样严重。该算法可实现大致上的“并行”布线，因而受布线顺序影响较小。并由于具有根据当前布线结果修正原布线的能力从而可望得到较高的布线完成率。文献[3]认为这是迄今为止最好的面向线网的布线算法，在实用上是非常有用的算法。

三、李氏算法中提高布线效率的方法

在李氏算法中，如何提高布线效率是一个严重的问题。一个实际的布图系统中，往往百分之七十到百分之八十的时间要化在李氏算法上。因此，在李氏算法的实际应用中，提高布线效率的方法得到了人们相当的关注。

1. 提高布线效率的一般方法

图6.7所示为三种一般的加速方法。

(1) 源点优选原则

图6.7(a)为布线时源点的选择对布线效率的影响。从前述李氏算法的思想可以看到，李氏算法中影响处理效率的主要过程是它的扩展过程。而扩展过程的处理效率与其到达目的点前（非

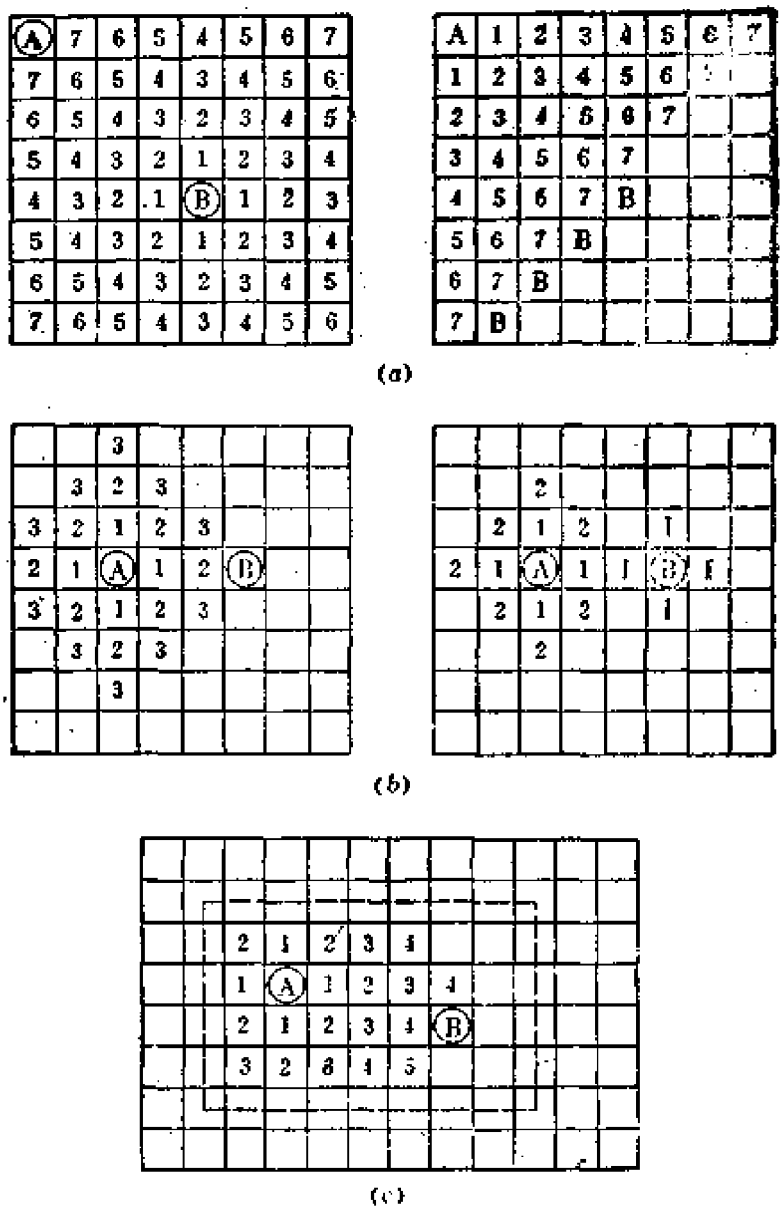


图 6.7　提高布线效率的加速方法示意图

加权扩展情况)需扩展的网格数有很好的对应关系。在图 6.7(a)中，线网的二个接点为 A，B。如果选择 B 点为源点，则需在所有的网格上都填上格值后，才能到达 A。反之，若选择 A 作为源点，只要对略大于一半的网格中填入格值便可到达 B。由此可得到一条启发式的源点选择规则，即应选择线网离布线域中心远的接点作为源点。这条原则在开始布线时有一定的加速效果。

(2) 双向搜索技术

图 6.7(b) 所示为双向搜索技术(有时也称双向迷宫)。在一个无穷大的布线场中，设二接点间曼哈顿距离为 n。则以一个源点向外扩展时，且不考虑障碍的存在，则与源点距离小于等于 n 的所有网格都扩展到，才能到达目的点。此时需扩展的网格数为 $2n^2-2n+1$。当 n 很大时，可近似为 $2n^2$。当把二个接点都定义为源点，同时向外扩展，直至不同源点的波前相遇为止，则每个源点仅需向外扩展 $n/2$ 远。扩展的总网格数为 $2\times 2(n/2)^2=n^2$，为单向扩展的 1/2。很明显，双向迷宫的程序将稍复杂一些，但效率的提高是显著的。

(3) 区域"迷宫"法

图 6.7(c) 是实用上较有效的区域迷宫方法。在一条线网布线前，在覆盖线网接点的最小矩形外加一个矩形的框，这个矩形框一般比上述最小矩形大 10～20%。在扩展时，限定该矩形框外的网格(包括框)不允许填数扩展。由于大部分连线在框内很容易完成布线，而大大减少了需扩展的网格数，而使布线效率得到提高。这一效果尤其在开始布线时更为显著。当然，如果在框内找不到布线路径，可把框放大，重新处理。如果失败次数较多，则可把框去掉再布。一般在布线的后期，区域布线失败机会较大，因而不采用此法。

2. 提高布线效率的 Rubin 策略

在上述这些方法中，存在着一个共同的问题，即在扩展时，离源点相同的波前中，各元素都继续向外扩展，而不管它们离目标是远了还是近了。若适当地对波前中的元素依其是否朝向目标加以分类，并赋以不同的扩展优先权，显然在许多场合下将大大提高扩展和到达目标的速度。1974 年 F.Rubin 提出的算法首先将这种思想付之于实践 [83]。Rubin 的定向搜索算法的思想可概述如下：设以源点和目标点为顶点的矩形为非退化矩形(即矩形的长和宽都大于 1)，则源点向邻点扩展时，当邻点都非障碍时，将产生四

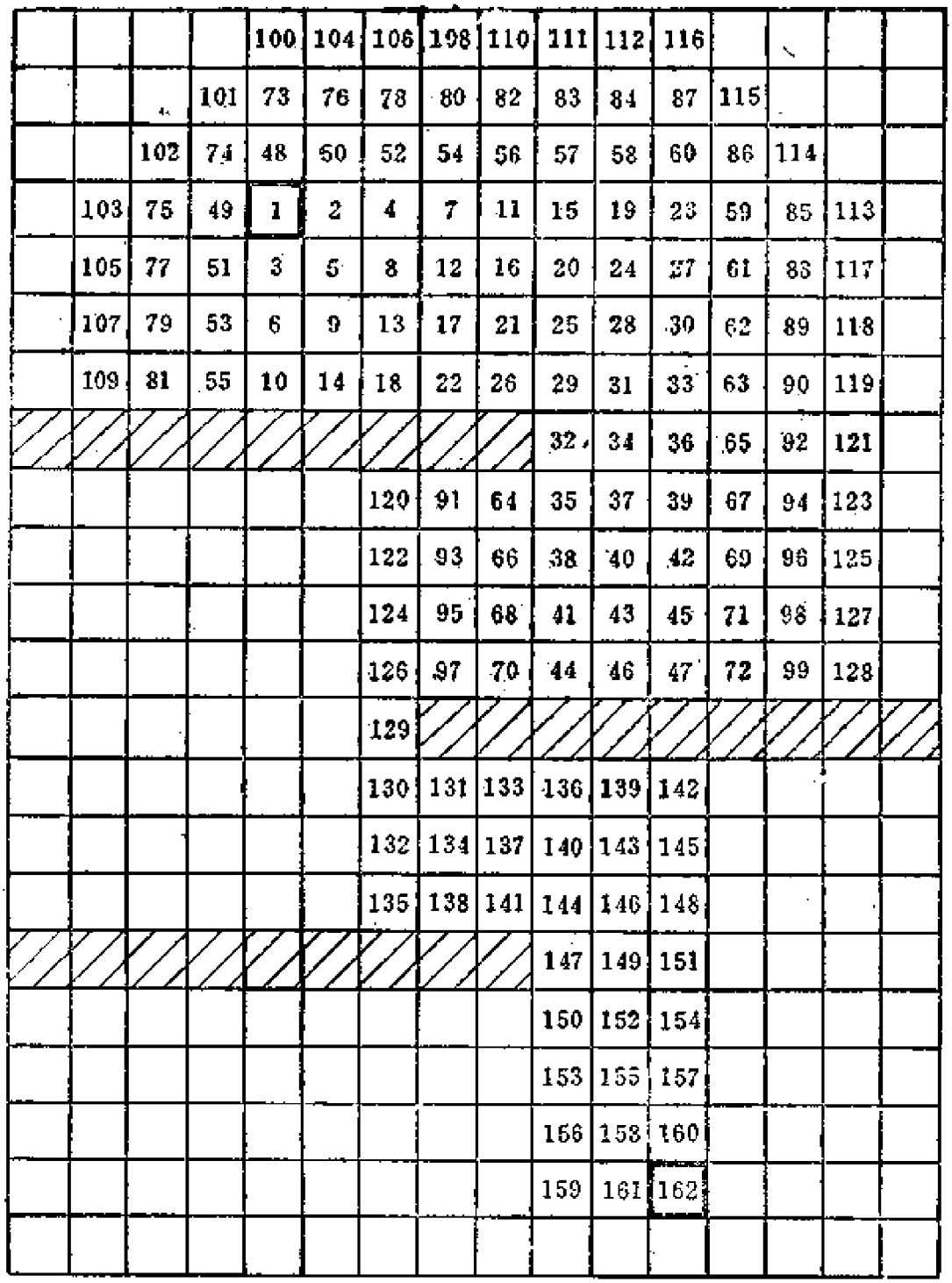


图 6.8　定向搜索法示意图

个波前网格点。其中二点离目标点的距离将小于源点离目标点的距离，这种点可称为朝向目标的波前网格点。相反，另二个波前网格点可称为背离目标的网格点。在扩展过程中，将产生的波前元素分为二个子集：朝向目标的网格子集及背离目标的网格子集。每次扩展时，优先扩展那些朝向目标的网格点。只有当波前表中没有朝向目标的网格点时，才扩展那些背离目标的网格点。图 6.8 即为该算法实现布线的一个实例。注意，在同一个波前元素的子集中，扩展顺序按先进先出为序。其目标点到达的判断及回找过程与一般李算法相同。图 6.8 中，应用定向搜索法，布线时需扩展的网格总数为 162，而一般李氏算法需 411。效率的提高是显著的。容易证明，用该法求得的路径为最短路径。

Rubin 的算法在引入"深度优先"搜索概念后，处理效率可进一步得到提高。

所谓"深度优先"搜索是指：

① 在朝向目标的波前元素子集中，最新的元素优先扩展。

② 设当前待扩展点为 j，一般 j 有二个朝向目标的邻格。若 j 点是由北面的一个邻格扩展而来的，则 j 点南面的邻格(朝向目标时)被选为下一个扩展到达的点，即扩展方向与 j 原方向相同的朝向目标的邻格优先扩展。

③ 对背离目标的波前网格点进行分级。第一次产生的背离目标的波前元素称为第一级，由第一级背离目标的波前元素产生的背离目标的新的波前元素称为第二级元素，依次类推。级别低的背离目标的波前元素先扩展。

④ 在同级背离目标的波前元素中，最新的元素优先扩展。图 6.9 为一个深度优先的定向搜索布线实例。例中布线中需扩展的网格总数仅为 114。

Rubin 的方法较好地描述了波前元素扩展方向的不同性质，但没有考虑这些波前元素的实际位置对其扩展优先度的影响。

3. 改进的定向扩展方法

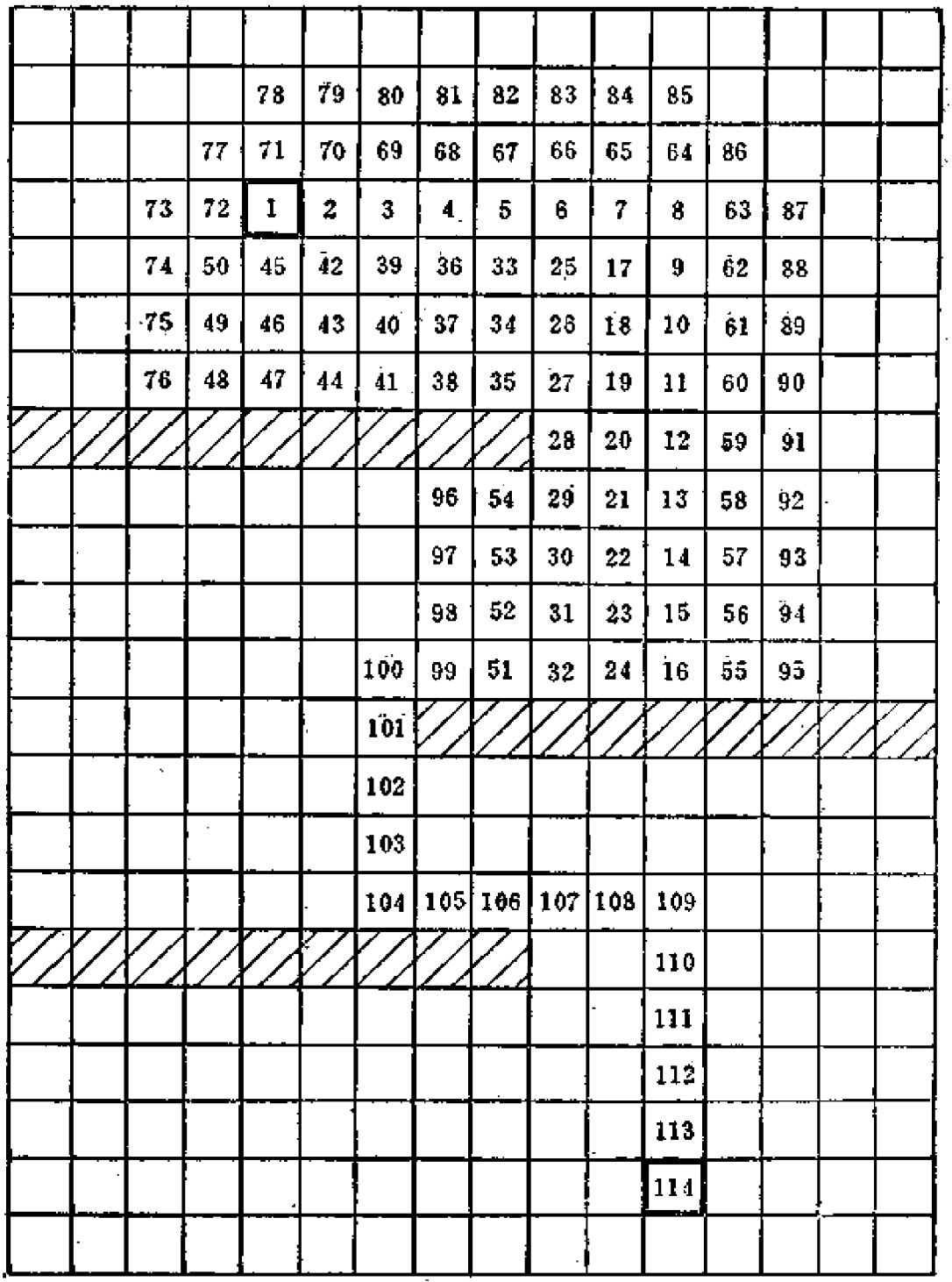


图 6.9　深度优先的定向搜索

文献[85]提出了一种新的波前元素扩展优先度的描述方法。该方法对波前中各元素定义了如下的扩展价格 $f(c)$：

$$f(c) = \cos t(s, c) + \alpha \times LB(c, t)$$

其中：$\cos t(s, c)$ 为源点 S 到网格 C 点的实际价格。当网格权重为1时，即为源点 S 到 C 点的最小曼哈坦距离。$LB(c, t)$ 为网格点 C 到目标点 T 的最低价格。可为 C，T 间的欧几里得距离。α 为方向因子。背离目标时 $\alpha = 2$，朝向目标时 $\alpha = 1$。

在每次扩展时，选取当前具有 $\min f(c)$ 值的波前元素先扩展。当相同代价点不唯一时，采取后进先出原则。

由于考虑了波前元素的实际位置，在不少情况下可进一步减少扩展的总网格数，从而进一步提高布线效率。

当以 $1.5 \times LB(c, t)$ 代替前式中的 $LB(c, t)$ 项时，即可增加在 $f(c)$ 计算中，由 C 点到目标点距离的权重，而使接近目标的波前元素优先扩展。虽然有可能丧失最短路径，但却可以收到更快速的布线效果。

四、李氏算法的多适应性及其应用

李氏算法之所以被较广泛地用于布线过程，除了它具有极强的绕障碍功能外，还由于此算法具有很好的适应性，易于修改和适应各种特殊情况和约束条件。前几节中所述的提高布线效率及布线完成率的方法从某种意义上也可看作是李氏算法多适应性的一种体现。本节将对其他应用条件下的情况进一步作一点介绍。

1. 适应不同电路的电性能要求采用各种不同的网格场

(1) 三角形网格场及其应用

在有些电路的布线中，如一些线性电路或超高速电路的布线时，对连线长有更严格的要求。在一般矩形网格的李氏算法中，求得的最短路径实际上是最小曼哈顿长度，并不是实际的最短路径（欧几里得长度）。这时，一种修改方法是采用三角形网格[86]，即定义每个网格的邻格为8个（如图6.10），这样与该点对角相邻的

网络也成为其直接的邻格，并定义这些邻格的扩展价格是相同的。其扩展过程和回找过程与一般李氏算法相同。从图 6.10 上可以看到，其扩展的波前呈矩形，而不再是一般李氏算法中的菱形。图 6.10(b) 为一条连线的布线实例。显然，从连线长度来看，其精度将高于一般李氏算法。它的问题是：由于对角邻格的扩展价格和实际距离的不对应性，而可能造成二条扩展总价格相等的路径在实际长度上并不相同。如图 6.11(a)。另一个问题是连线间距的不均匀性，如图 6.11(b) 所示。图中水平连线段间距为 D，而 45° 斜线段间距为 $d=\frac{\sqrt{2}}{2}D$。

(2) 六角形网格场及其应用

在平面上以单位间距的水平平行线集(0°)，60° 及 120° 斜线集构成六角形网格场，如图 6.12(a) 所示。此时，每一网格共有 6

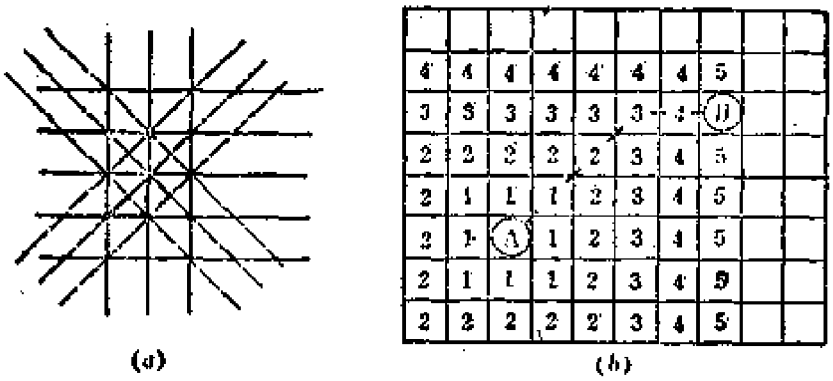


图 6.10 三角形网格布线示意图

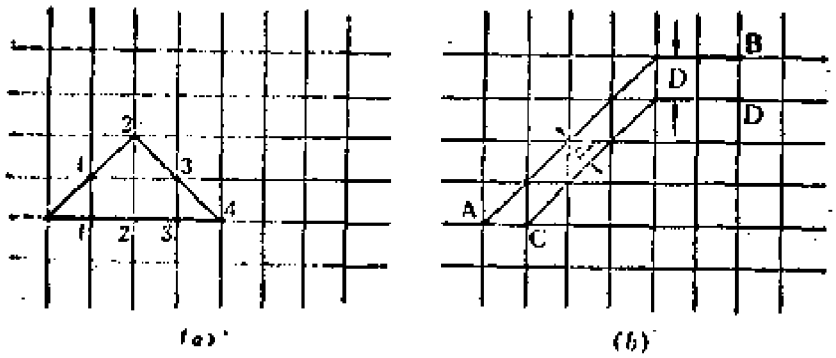


图 6.11 三角形网格布线所存在的问题示意图

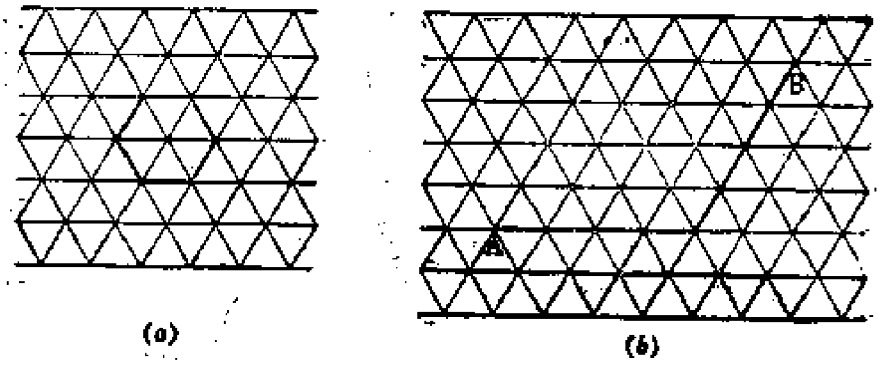


图 6.12　六角形网格布线示意图

个邻格，且这些邻格与该点是等距的。图 6.12(b) 为一条连线的布线实例，其连线长度的精度将优于一般李氏算法（矩形网格）且连线间距均匀。

2．李氏法用于多层布线

在实际布线问题中，往往允许布线可在几层平面上进行，不同层（平面）上的连线可通过通孔实现必要的互连。为说明方便，本节以二层布线为例。

(1) 一般的二层布线

在二层布线中，人工布线的实践表明：为获得较高的布线完成率，一个有效的方法是把连线的水平线放在一层，而把垂直线放在另一层。考虑到尽可能减少通孔。上述原则可修改为把大部分水平线或近似为水平线的连线放在一层，而把大部分垂直线或近似为垂直线的连线放在另一层。而李氏算法应用于双层布线时，可很方便地模拟人们的这种布线原则。

从布线层的特性来看，上述原则可等价地描述为在一层上，水平方向的布线价格较小而垂直方向的布线价格较大，因而在该层上宜于布水平线，另一层上恰相反。因此可规定在一层上水平方向扩展的网格权重为 1，而垂直方向的扩展网格权重为 2（或更大）。而且另一层水平扩展网格权重为 2，垂直扩展网格权重为 1，令设计者为尽量减少通孔数而规定在上层上连线垂直线段长度小于和等于 L 时（在下层连线水平线段长度小于、等于 L 时）不穿孔，

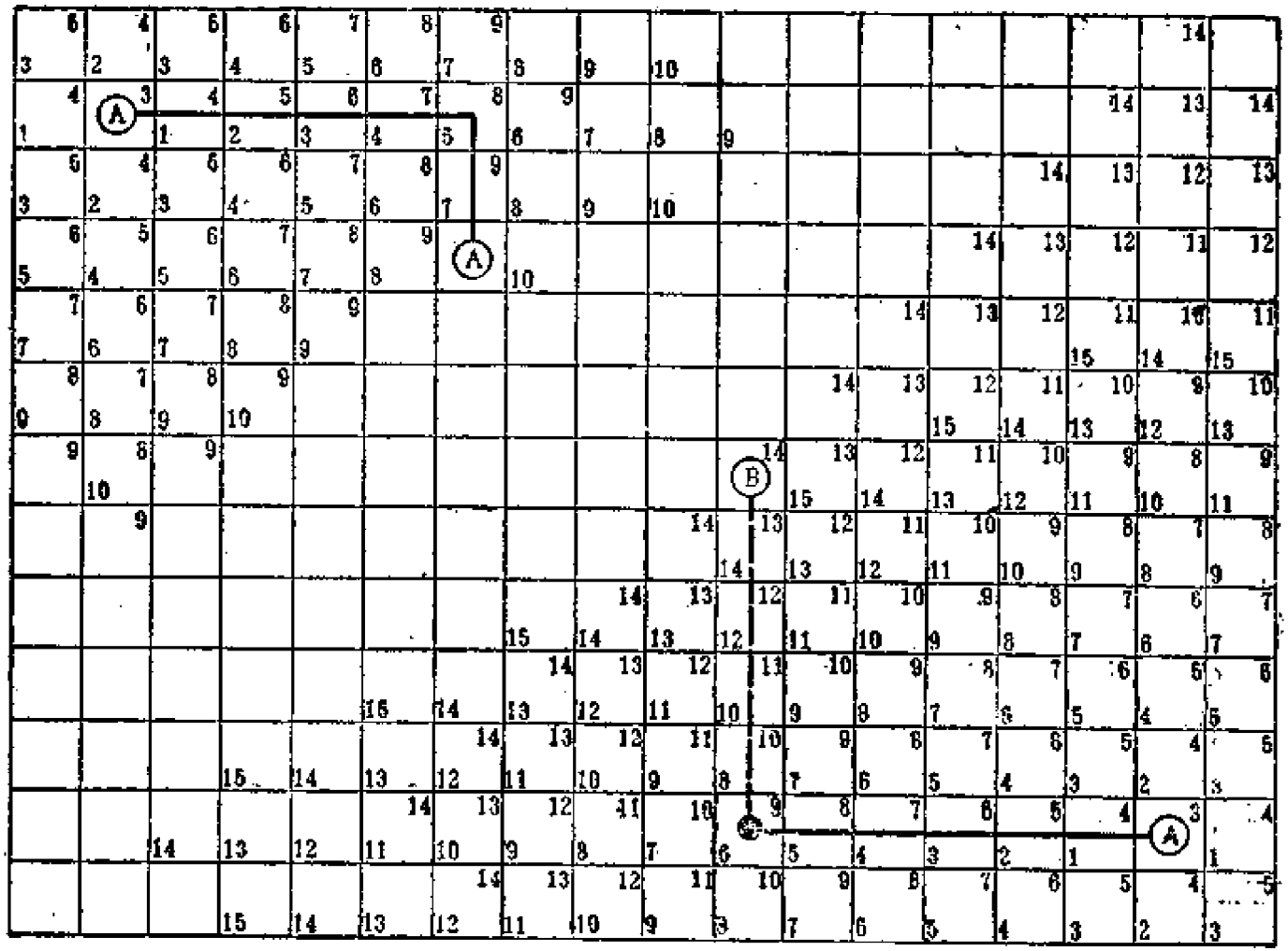


图 6.13　在二层布线中二条连线的实际布线图

而全部布在上层(或下层),可计算得到穿孔的价格

$$\cos t_3 = \frac{L}{(\cos t_2 - \cos t_1)},$$

其中 $\cos t_1$ 为上层水平方向(或下层垂直方向)扩展价格,$\cos t_2$ 为上层垂直方向(或下层水平方向)扩展价格。当 $L=3$, $\cos t_1=1$, $\cos t_2=2$, 则 $\cos t_3=3$。通过调节 $\cos t_1$, $\cos t_2$, $\cos t_3$ 即可很好地模拟人工布线时的考虑。图 6.13 为二条连线的实际布线例,图中左下角为上层格值,右上角为下层格值,实线连线表示上层布线,虚线连线表示下层连线,圆点表示通孔,显然当 $\cos t_3=0$ 时,即为严格地横竖线分层布线。

(2) "一层半"布线

在实际的集成电路工艺中,如 MOS 工艺中,布线往往除可以通过一个金属(Al)层来实现外,有些连线或连线的一部分还可通过多晶硅或扩散区的互连来实现。虽然多晶硅和扩散区不在一层上,但由于被利用来作为连线的多晶硅线条和扩散条不会发生交叉(否则形成晶体管了),因此从布线的角度可把它们都看作是在同一个布线层上。但该布线层与金属布线层在性质上有较大的差别:

① 电阻率的差别。多晶硅条或扩散条电阻率高,因此,对多晶硅条或扩散条的长度有较严格的限制。

② 在多晶硅,扩散区组成的"辅助布线层"上,除布线接点外,还存在许多用以形成晶体管等电路元件的区域是不允许用于布线的。

实际可用于布线的区域远小于金属层,因此我们称金属层为整层,称该辅助布线层为半层。在人工布线时,考虑到半层上电阻率高,可利用区域少等因素一般可确定一个半层上走线长度的上限值为 d_1。另一方面规定只有当连线在整层布线时需绕行相当长的距离为 d_2(或无法完成时),才利用通孔将一段连线布在半层上(渡桥)。而李氏算法可相当好地模拟人的这些规范化的设计考

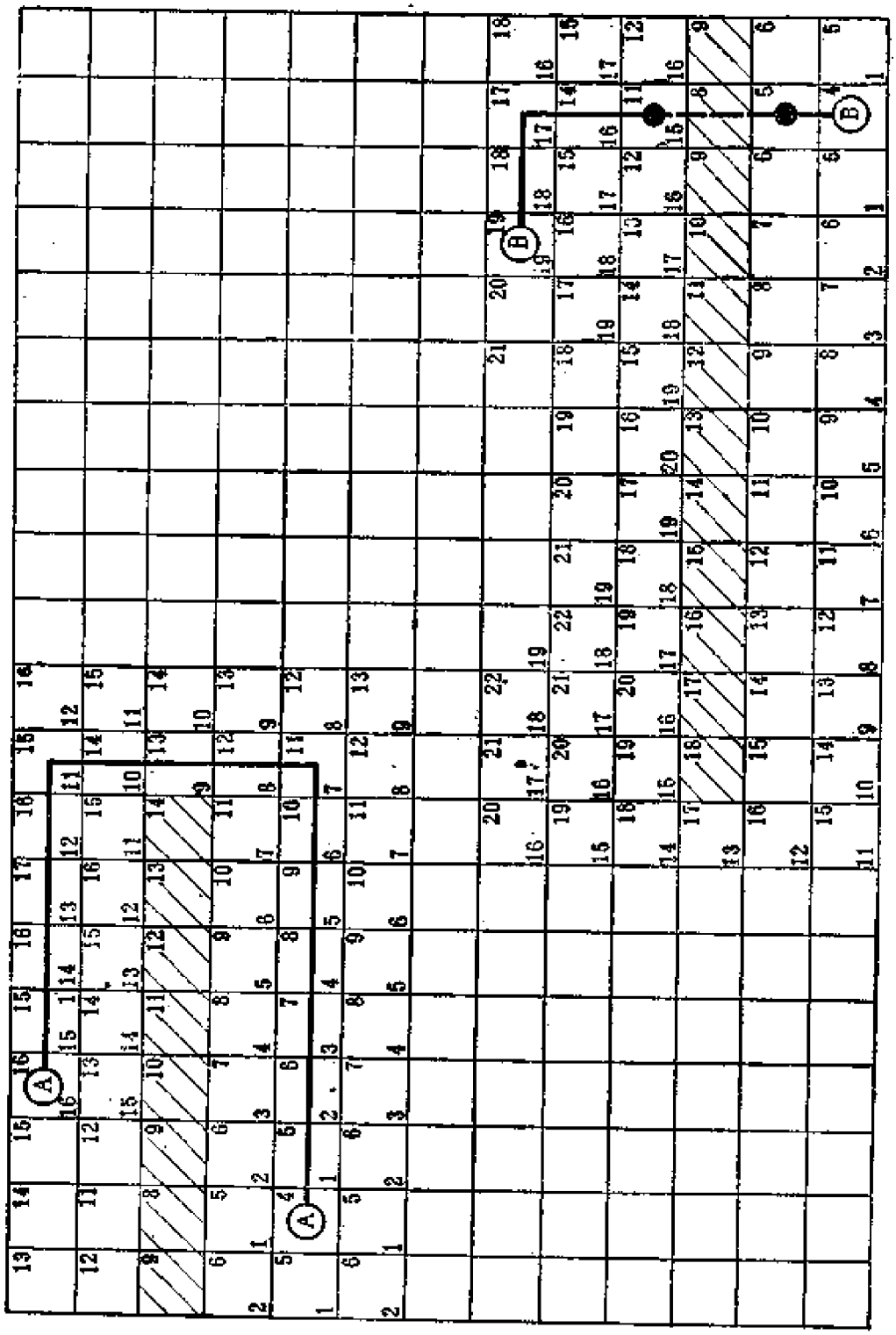


图 6.14　一层半布线示意图

虑。可分别定义整层布线价格 $\cos t_1$，一般 $\cos t_1 = 1$，半层布线价格 $\cos t_2$，通孔价格 $\cos t_3$。根据给定的 d_1，d_2 值可求出对应的 $\cos t_2$ 和 $\cos t_3$。

$$\begin{cases} d_1 = 2 \times \cos t_3 / (\cos t_* - \cos t_1) \\ d_2 = [2 \times \cos t_3 + (W+1)\cos t_2] / \cos t_1 \end{cases}$$

其中：W 为障碍宽度；d_* 为绕行 W 宽度障碍的连线长度上限。一般为处理方便，令 $W = 1$，并取与之相应的 d_2 为算法参量。

例如当 $d_1 = 4$，$d_2 = 14$ 时，求得 $\cos t_2 = 3$，$\cos t_3 = 4$。这些价格的物理意义是：

① 当接点在半层上时，若接点同距离小于，等于 4 时，连线可全部布在半层上，否则，利用通孔布到整层上。

② 当接点在整层上时，如果绕行一个单位宽度的障碍，将使连线长度的增加部分大于 14 时，则利用通孔，用半层作“渡桥”实现布线，如图 6.14[81]。

这个算法几乎有无数种可能的方案。也可把此算法推广到三维网格场并用于有通孔的多层布线 [88]、[89]。实际应用充分证明了李氏算法有较好和较广的适应性。

五、李氏算法用于多接点线网的互连

一般的李氏算法可实现二点间的最短路径的连接，但在实际问题中，不少线网的接点多于 2 个，这些线网可称作多接点线网。为使整个线网的连线总长度最短，必须解决接点间的实际连接关系，即决定任二个接点间究竟应该直接相连，还是通过其它接点或附加接点实现电学上的连通。

对多接点线网一般存在着三种基本的互连方式，即最小链接树方式(图 6.15(*a*))，最小树方式(图 6.15(*b*))，最小斯坦纳树方式(图 6.15(*c*))。

1. 最小链接树方式

树上每个顶点对应着线网的接点，且树上每个顶点的度数小

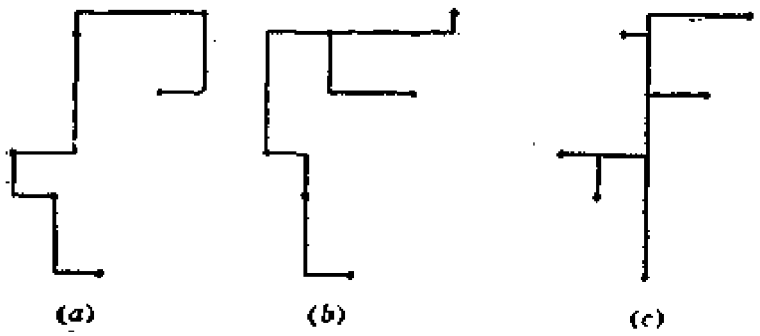


图 6.15　在矩形网格场中的三种基本连接方式

于、等于 2。求一个线网的最小链接树是与经典的巡回售货员 问题 [90]、[91] 等价的。从理论上来讲，当线网接点数足够大时，求其最小链接树将耗费相当多的机时。在实际设计时，由于 70～80% 的线网的接点数小于等于 4，且线网数相当多，同时实际互连时还必须考虑其他影响连线长度的因素（如已布线造成的障碍等），因此常常采用一些近似的方法来求准最小链接树。

求准最小链接树的一种方法是，首先选取一接点 A 作开始的连接，被连接的接点是离其最近的那个接点 B。此时形成一条当前链 A—B。下一步在未连接点中求与当前链二端接点（即度数为 1 的接点）距离最近的接点 C，并实现 C 与相应链端点的连接，形成新的当前链。重复上述过程，直至所有接点都已实现连接为止。这个方法在接点数较多时，虽不能保证求得最小链接树，但具有快速与结果接近最小链接树的优点。当用该方法决定了接点对之间的实际连接关系后，多点线网即可转换成 $(n-1)$ 对二点线网，并可方便地实用一般李氏算法实现实际的连接。

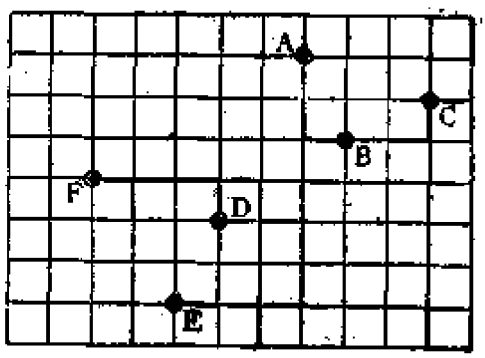
2. 最小树方式

树上每个顶点对应着线网的接点，对顶点的度数没有限制，此时，可采用最小树算法 [92]、[93] 来求解。

首先计算各接点对之间的距离，并将其从小到大排列（共为 $\frac{n(n-1)}{2}$ 对）。算法从连接距离最近的一对结点开始，然后联接下

一个距离最近的接点对。此时需检测，新联结的接点对是否是已有联结路径的接点对（即是否可能产生一个回路而破坏其树的特性），若是，则跳过而不必联结，去试联结下一个距离最近的接点对。重复上述过程，直至所有的接点都实现了联接为止。

在判定待联结的接点对间是否已存在联结路径时，可利用联结矩阵[C]。这是一个 $n\times n$的 对称 0—1 矩阵，n 为接点数。若接点 i 和 j 之间存在一个直接的连接（即 i 和 j 间存在一连线），则 $C_{ij}=1$，否则 $C_{ij}=0$（C_{ii} 总等于 1）。使用这个矩阵记录每次选用的接点对的情况，如图 6.16(b) 中记录了目前已选用的接点对 A—B、B—C、D—E、D—F。若用通常的方法对图 6.16(b) 所示的



(a)

	A	B	C	D	E	F
A	1	1	0	0	0	0
B	1	1	1	0	0	0
C	0	1	1	0	0	0
D	0	0	0	1	1	0
E	0	0	0	1	1	1
F	0	0	0	0	1	1

(b)

	A	B	C	D	E	F
A	1	1	1	0	0	0
B	1	1	1	0	0	0
C	1	1	1	0	0	0
D	0	0	0	1	1	1
E	0	0	0	1	1	1
F	0	0	0	1	1	1

(c)

	A	B	C	D	E	F
A	1	1	1	0	0	0
B	1	1	1	0	0	0
C	1	1	1	0	0	0
D	0	0	0	1	1	1
E	0	0	0	1	1	1
F	0	0	0	1	1	1

(d)

图 6.16　利用联结矩阵求最小树

矩阵求平方，但用逻辑加来代替所需的加法运算，就得到图 6.16(c)所示的矩阵，在所得的矩阵中，元素值为 1 所对应的接点对间必存在联结的路径，且该路径的组成边小于、等于 2。而 C^k 能指示出由 k 条边组成的接点对间的路径，因此，为判定接点对间是否已存在联结的路径，只需对图 6.16(b) 所示的矩阵求 k 次方 ($k=2$, 3, …)直至矩阵中的 1 元素不再增加为止 (图 6.16(d)是 $k=2$ 时的情形)。所以，新联结的接点对应满足二个条件：在 C^k 中对应的元素值应为 0；且接点对距离为当前最小值。据此。应选择 B—D 为新的联接点对 (A—C、E—F 在 C^3 中对应元素值为 1)。当矩阵的 k 次方后，全部元素都为 1 时，该线网的所有接点实现了联结。求得最小树后，可方便地把多点线网转换成二点线网并用一般李氏法实现布线。

在利用最小树方式实现联结时，可方便地考虑其他布线约束条件。这只需在选取每个接点时，检查一下，加进这条边是否会破坏附加的布线约束条件。若破坏，则可选取下一个最好的接点对。

最小树算法还有另一种有用的方案。它的优点是不需要联结矩阵。但此方案不易进行修改以适应附加的布线约束条件。我们可从任意一接点开始检查。然后每次只需在被检查过的接点和未检查过的接点间找出距离最近的接点对。并选用该接点对，把相应的接点列为已检查接点。重复上述过程，直至所有的接点都已被检查过为止。

3. 最小斯坦纳树方式

在不考虑其他约束条件时，线网连线总长最短的布线方式是斯坦纳树方式[94]、[95]、[96]。线网的斯坦纳树即除线网接点外，加入 k 个附加接点 (k 可为 0) 后，$n+k$ 个接点的最小树。此问题无论在欧几里得或曼哈顿距离情况下，都没有取得一般的解决。当利用李氏法实现布线时，较好的方法是结合布线过程来解决。

我们可用一个实例来加以形象地说明。如图 6.17(a)，一条

线网含8个接点(A—H)。开始时可选取一个接点(如为E),把E作为源点而把其他7点作为目标点作一般李算法的扩展过程，过程继续到第一个目标接点(D)到达为止。这条路径于是被实际布线(图6.17(b)中线1)。然而将线1的各点作为源点，把其余6个接点作为目标点，又把扩展过程继续到第一个目标接点到达为止(此时为C接点)。这里需指出的是，在实际布线时，C并不直接连到D或E上，而连到DE连线的某一点，使连线线长尽量短，*号处即为斯坦纳树上的附加上的顶点。现在把C—D—E线网上各点作为源点，过程一直继续下去，直至实现全部联结为止。图6.17(b)为最后布出的结果，数字表示布线的顺序。

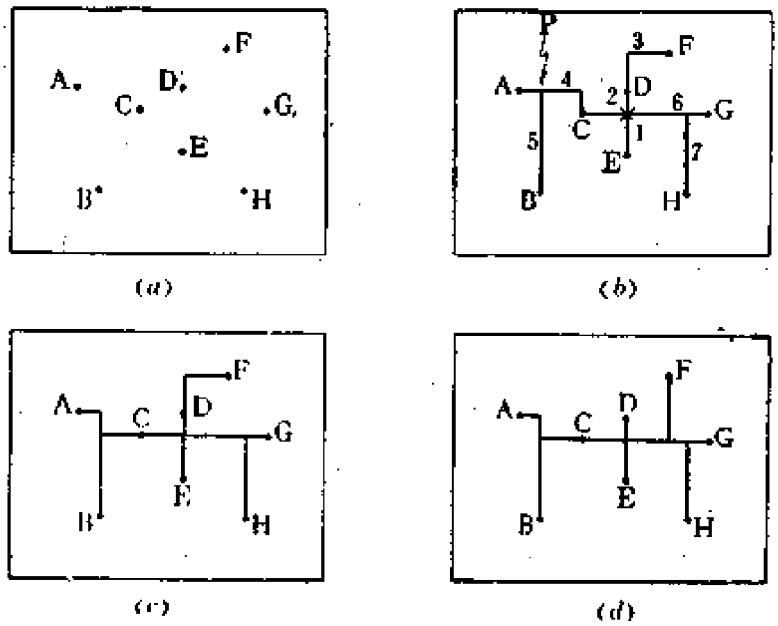


图6.17　最小斯坦纳树布线方式示意图

可以看到，在实现斯坦纳树方式联结时，李氏算法需作的修改是很小的，但一般来讲，结果并不是最小斯坦纳树。为了获得更好的结果，可采用部分迭代的方法。

当移去树上的一条边后，可得到二个子树(其中一个子树可能只有一个顶点)。我们以其中一个子树上所有点（包括连线上的点)作为源，以另一子树上所有点为目标点，用李氏算法求其联结的最短路径。如所求的最短路径小于被移去的原边的长度，则用

该最短路径代替原边而获得一个更好的解。在图 6.17(*b*) 中，当移去 PC 边后，可求得一更短的联结路径，得到图 6.17(c) 的结果。同样，在图 6.17(c) 中，当移去 *DF* 边后，可获得一更好的结果，如图 6.17(*d*) 所示。

考虑到布线时存在着障碍的情况，上述方法是求多点线网布线解的较好方法。

六、李氏算法应用中节省存贮量的方法

由李氏算法的原理可知，它必须把一个实际的布线问题映射为一个网格场上的格网的联结问题。对于每个被处理的网格，在计算机中必须有相对应的存贮单元。对于一片面积为 8×8 mm 的芯片，如线宽和线间距分别为 4μ，并采用双层布线，则每个网格的信息需占用 2 个字节，总共约需 8000000 字节的存贮量，而且当线宽和线间距进一步缩小时，所需存贮量将以平方的关系增加，这在很多情况下是令人难以忍受的，这也是李氏算法的主要问题之一。

为了减少李氏算法在应用中所需的存贮量，让我们来考察一下最后一步填数后得到的网格场中数据的规律(见图 6.1(c))。可以发现，任何网格值为 k 的邻格的网格值不是 $k+1$ 便是 $k-1$，分别表示了 k 值网格的“先行”者($k-1$)和“后继”者($k+1$)。这一点提示我们，如果格值的按排能够将它和它的邻格之间的“先行”——“后继”关系描述清楚就可以了。因此可设计一个简单的填数序列，如 1—1—2—2—1—1—2—2……按此数序进行填数，就可方便地确定每一个网格与其邻格的“先行”——“后继”关系，很方便的是在回找时只要按照上述给定的数序回找就可以了。使用这种填数序列，每个网格中的信息只是下述四种之一：空、障碍、数 1 或数 2，因此记录每个网格的信息只需 2 位二进位就够了，存贮量可减少几乎一个量级（对于大的布线问题而言）。图 6.18 所示为这种方法对图 6.1 所示布线问题实行李氏算法布线的情况（填数已到

达目标点 B)。首先在 A 的邻格中填上 1，接着由已填数的邻格向外扩展，再填 1，在下一次扩展中填 2，然后再扩展，还填 2，在二次填数 2 后，继续扩展时填 1，等等，直至到达目标点 B。图 6.18 上并标出了回找的路径。

需要指出的是，当李氏算法为适应其他需要(如提高布线成功率等)而进行修改后，将较难直接运用上述 1—1—2—2 那样的简单填数方案，但通常可找到类似的填数方式以减少存贮量的要求，只要这种填数方案能够满足在回找时对每一个网格来讲，都可清楚地确定它和邻格的“先行”——“后继”的关系就可以了，文献[28]介绍了用于多层布线的填数方法。

尽管如此，需要存贮量大仍是李氏算法的一个令人遗憾的问题之一，为此，提出了不少其它的布线方法，试图在布线效率和所需存贮量方面有较大的改进。

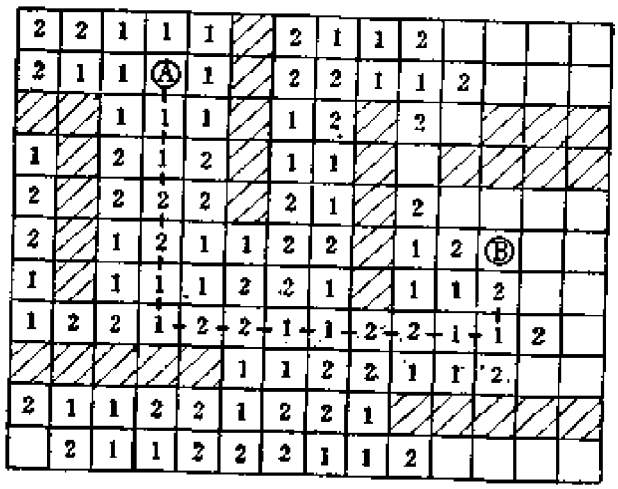


图 6.18　填数方法修改后的李氏算法

6.3　其它面向线网的布线方法

由 6.2 节我们可以看到，由于李氏算法中搜索可布路径是基于网格场的扩展(有时我们把这种逐个网格扩展的方法称作为“点扩展”方式)，使李氏算法具有极强的绕障能力，并能保证获得当前

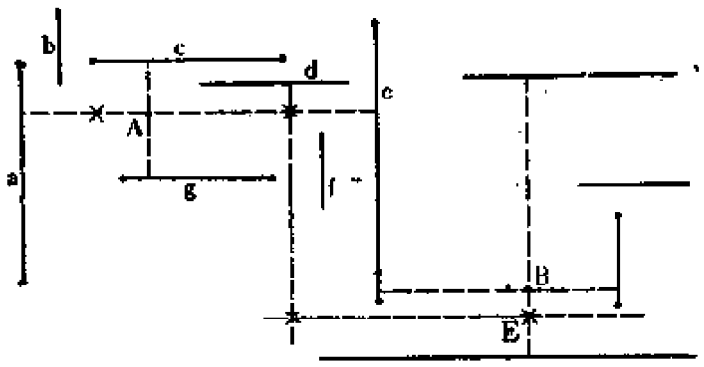
最短的联结路径。与此同时，付出的代价是所需的存贮量大和扩展、搜索的效率低。由于李氏算法中在布线时是逐个线网顺序处理的，即串行方式，因此并不能保证总的连线长度最短化，也不能保证获得100% 的布线完成率。为提高布线效率和减少对存贮量的要求、为进一步提高布线完成率，提出了一些其它的面向线网的布线方法。

一、线探索法

1969 年 D.W.Hightower 提出了一种基于线扩展的布线方法，也就是线探索法(linear expansion)[99]。

1．问题的描述

设在图 6.19 中，A、B 二点为一线网的二个接点，实现 A、B 间布线的问题可描述为：给定一个不可穿越的垂直线段集 C_V 和一个不可穿越的水平线段集 C_H（即已布线）。求一条从 A 到 B 的且不与任何 C_V、C_H 元素相交的尽可能短的路径。



•——接点，*——逸出点，×——相交点

图 6.19　线探索法布线示意图

2．算法的描述

(1) 定义

为了算法描述的方便，我们首先给出一些简单的定义。

*D*1：如果由点 P 作一条垂直线，与之相交的 C_H 的元素称为 P 的水平障碍(cover)。由 P 作一条水平线，与之相交的 C_V 的元素

称之为 P 的垂直障碍。

图 6.19 中，c、g 为 A 的水平障碍，a、e 为 A 的垂直障碍。而 b、d、f 线段不是 A 的障碍。

$D2$：通过 P 点的垂直线段（终止于 P 的水平障碍）称为 P 的垂直逸出线(escape line)。同理可定义 P 的水平逸出线。

$D3$：P 的垂直(水平)逸出线上的一点 E，若满足下述条件：

① P 点的一条垂直障碍已不是 E 点的垂直障碍，且

② P 点与该垂直障碍之间的任何垂直线段都不是 E 点的垂直障碍。

则 E 点称为是该逸出线上的一个逸出点(escape point)。

$D4$：由接点 A 衍生的逸出线及其逸出线上逸出点进一步衍生的逸出线集定义为 $\{L_H^A, L_V^A\}$，其中 L_H^A 为水平逸出线，L_V^A 为垂直逸出线。同理可定义 $\{L_H^B, L_V^B\}$。

(2) 线探索法的主要思想

线探索法的主要思想是非常直捷的；首先可由线网的待联结接点引出一条水平逸出线和一条垂直逸出线，构成最初的 $\{L_H^A, L_V^A\}$ 和 $\{L_H^B, L_V^B\}$。若它们中有一对元素相交，交点为 (x, y)，则自然立即找到了一条连接路径 $(A, (x, y), B)$。否则在 $\{L_H^A, L_V^A\}$ 和 $\{L_H^B, L_V^B\}$ 上分别求逸出点 $\{E^A\}$，$\{E^B\}$，然后，作过这些逸出点的新的逸出线，并加入 $\{L_H^A, L_V^A\}$ 和 $\{L_H^B, L_V^B\}$。当出现一对元素相交时，即找到了连接路径。如果没有任何一对元素相交，则可重复上述过程直至得到一对相交的逸出线或 $\{L_H^A, L_V^A\}$ 和 $\{L_H^B, L_V^B\}$ 都没有新的元素产生(布线失败)为止。

连结路径的确定：可由相交点出发返找，相交点所在逸出线的原逸出点即是路径上的一个拐弯点，设为 E_i^A 和 E_j^B，然后可由 E_i^A 找到下一个相关的逸出点 E_{i-1}^A，直至分别返找到 A 和 B，而确定了连接路径为 $(A, E_1^A, \cdots E_i^A, (x, y), E_j^B, E_{j-1}^B, \cdots B)$。

为了提高搜索效率，可对逸出点的搜索顺序给定一些简单的原则，如离源点近的逸出点先搜索，离目标点近的逸出点先搜索，

朝向目标点的逸出点先搜索等。

(3) 逸出点的求法

算法中逸出点的求法有二种方法：

① 设 t_1, t_2, t_3, t_4 为 P 点水平障碍的端点，并设以离源点近的逸出点先搜索为逸出点选优处理原则，如图 6.20 所示，可首先计算出 P 到 t_1, t_2, t_3, t_4 的欧几里得距离，并按其距离大小，由小至大排列，即该排列为 (t_4, t_1, t_3, t_2)。然后依次根据各端点坐标求逸出点位置，令 U 为单位长度(线宽 + 线间距)，对 t_4 来讲，可在 P 的水平逸出线上求一点，使该点坐标为 (x_4+U, y_P)。并检查，该点是不是一个未用过的逸出点，继续上述处理，对 t_1，求得的点为 (x_1-U, y_P)……。同理，可按上述方法可求得垂直逸出线上的逸出点。

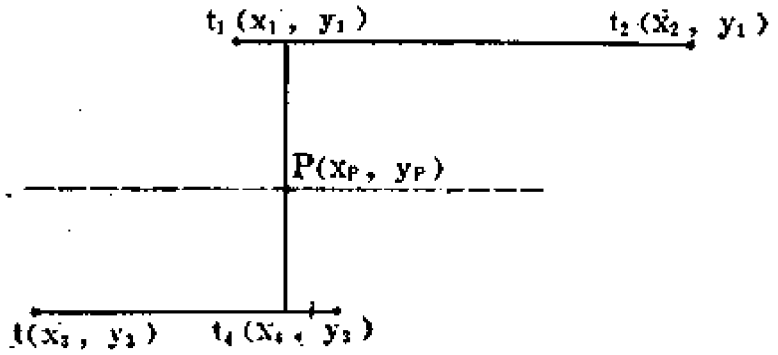


图 6.20 逸出点求法示意图

② 仍如图 6.20 为例，(t_1, t_2) 为 P 的水平障碍，可在 P 的垂直逸出线上取一点 $F(x_P, y_1-U)$，通过 F 点，作一条水平逸出线，若其与目标点的逸出线集的一个元素相交，则已找到联结路径；否则按第一种方法去该水平逸出线上寻找新的逸出点。同理，可对 P 点的其他障碍作类似的处理，以寻找新的可能的逸出点。

(4) 结果的优化

由于在上述算法中，缺乏从众多的逸出点中选择最有效的逸出点的功能，一个可能的结果就是使求得的路径并不是最短的，因

此，需要一些算法对结果进行优化。

① 删除路径中多余的拐弯点的方法。如图 6.21(a) 所示，若 E_k, E_{k+1}, …, E_{k+m} 是路径的一部分，且 E_k 和 E_{k+m} 处在同一条逸出线，E_{k-1} 不在这条逸出线上。$\{E_{k+m}, P\}$ 中的每一点都描述了一个路径的拐弯点，即没有二个点是处于同一条线上的，则 E_{k+1}, …, E_{k+m-1} 中诸点可以删去，并得到了一条更短的联结路径。

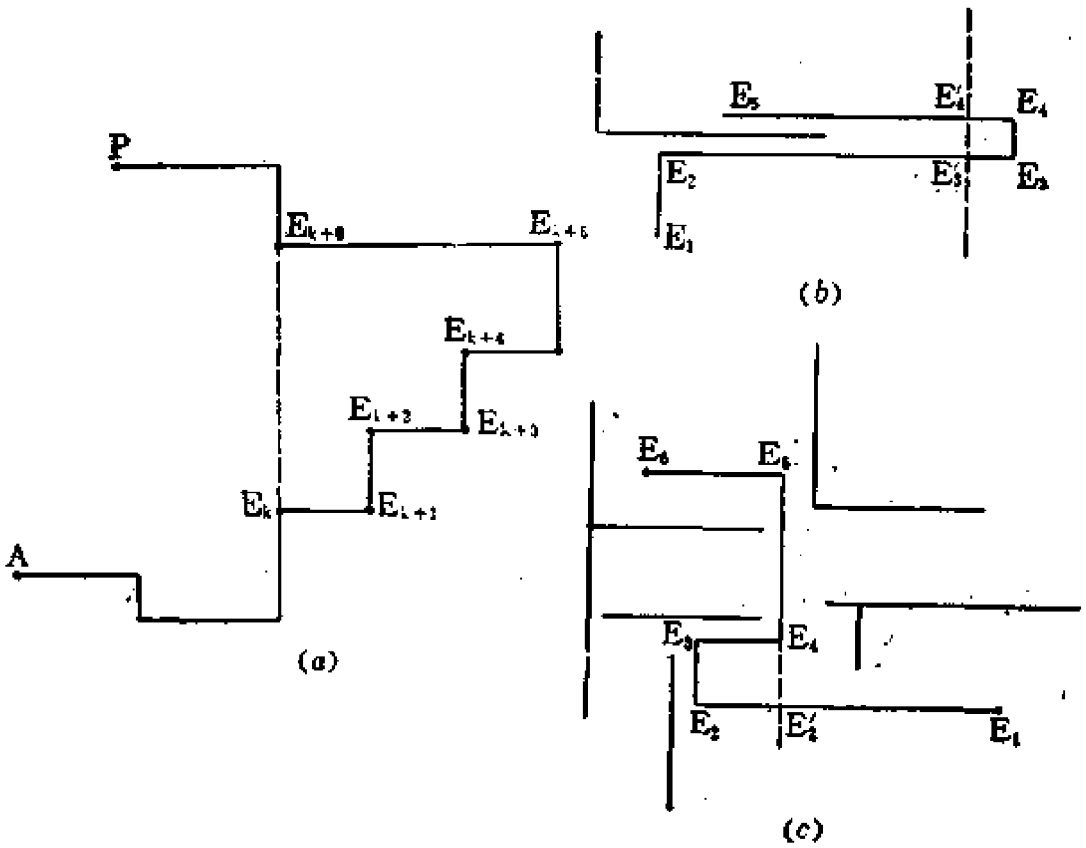


图 6.21　结果优化示意图

② 删除路径中多余的段的方法。如图 6.21(b)，E_2E_3、E_4E_5 段可再缩短一些。其优化方法是作垂直线段平行的直线，检查它是否仅仅与原线段中的二条水平（或垂直）线段相交，若是，则用 $E_3'E_4'$ 代替 E_3E_4，并删去多余的线段 $E_3'E_3$、E_3, E_4、E_4E_4'。如图 6.21(c)所示，E_1E_2 线段可再缩短一些，其路径优化的方法是将原线段延伸，检查它是否在遇到障碍前与路径中的其它线段相交。图

例中当将 E_5E_4 延伸时，它将和 E_1E_2 相交。然后将 E_2' 到 E_4 段重新处理，即将 E_2' 代替 E_4，并删去原 E_2、E_3、E_4 各点。

(5) 线搜索法的特点

从上面对线探索法的介绍可以看到，由于采用线扩展，使算法可望具有较高的布线效率，尤其在开始布线时，由于已布线数量较少，障碍较少，该算法有极高的布线效率。这一点和李氏算法的情况恰好相反，李氏算法一般地来讲，随着已布线数量的增加，由于所需扩展的网格数减少，而使布线效率得到一定的提高。

由于线探索法采用线扩展的方法，因而对存贮量的要求也明显降低了，而且当布线问题规模增大时，所需内存容量也不象李氏算法那样成平方关系地增加，使其较为适合用于规模大的布线问题。

采用线扩展的方式另一个优点是使求得的路径的拐弯数得到最小化。这对不少实际布线问题是有利的。

线探索法的问题是：它并不总是能获得最短路径。在实际应用时，其布线结果常常是非常接近李氏算法的布线结果的，该算法的布线效率随着已布线数量的增加而下降。当采用较简单的求逸出点的方法时，有时在客观上存在着连通路径时，会出现找不到路径的情况。换言之，其绕障能力低于李氏算法。

(6) 算法的具体应用

在实际应用时，人们将李氏算法和线探索法结合起来应用。在开始布线时，采用线探索法，使布线效率大大提高。此时，线探索法的缺点也是不明显的。而当已布线数量足够多时，采用李氏算法布线，利用李算法极强的绕障能力以提高布线完成率。同时，在这种情况下，李氏算法可具有相对较高的布线效率。

线探索法可方便地推广到二层布线中去，此时，可采用所谓"先布线，后分层"的处理原则。按这种原则实行布线时，对未布线而言，只有其它各线网的接点和已布线的交叉点是当前的布线障碍，可用上述算法同样地实行布线。当布线全部完成后，再应用分

层及通孔最小化算法(参见6.5节)将布线适当地分到二层上，并使所需的通孔数最小化。

(7) 算法的发展

近年来，有些研究人员把线扩展和网络扩展结合起来提出了一些新的算法。

在Heyns等人的方法[100]中，象Hightower方法一样，他们的方法也是从线扩展开始，但是然后又象李氏算法那样去填充一个区域，与李氏算法不同的是，在这种方法中并不需要把整个区域保存在内存贮器中而只需记录区域的边界线，然后从这些边界线出发，继续扩展。最终得到的互连通路具有Hightower算法类似的特点，可使路径的拐弯点最少化。而且在这种方法中只要连接的路径存在就一定能找到实现布线的通路。由于这种方法本质上是非网格化的，因此可望减少所需的存贮量并提高布线效率。

文献[101]描述了另一种方法，它利用波传播中的绕射规律，在简单的线探索的基础上，把波扩展到整个区域。当遇到布线障碍时，波发生绕射，并记录这些绕射边界，然后绕过障碍继续扩展，直至找到存在的互连路径为止。这种方法布线时，可使布线路径是"逼近障碍"的，将有利于提高布线完成率，同时由于在本质上是非网格化的，因此所需的存贮量将比李氏算法少而布线效率较高。这种方法布线时可具有很强的绕障能力，当存在着互连路径时，一定能找到实现布线的通路。

二、整体布线

1. 面向线网的布线方法存在的问题

在面向线网的布线方法中，如何估计当前布线对未布线的影响，使已布线不影响未布接点互连的实现，这一问题是一个人所共知的难题，也是影响布线成功率的一个主要原因。

图6.22和图6.23为先布的线网将使未布线网无法实现互连的二种基本情况。

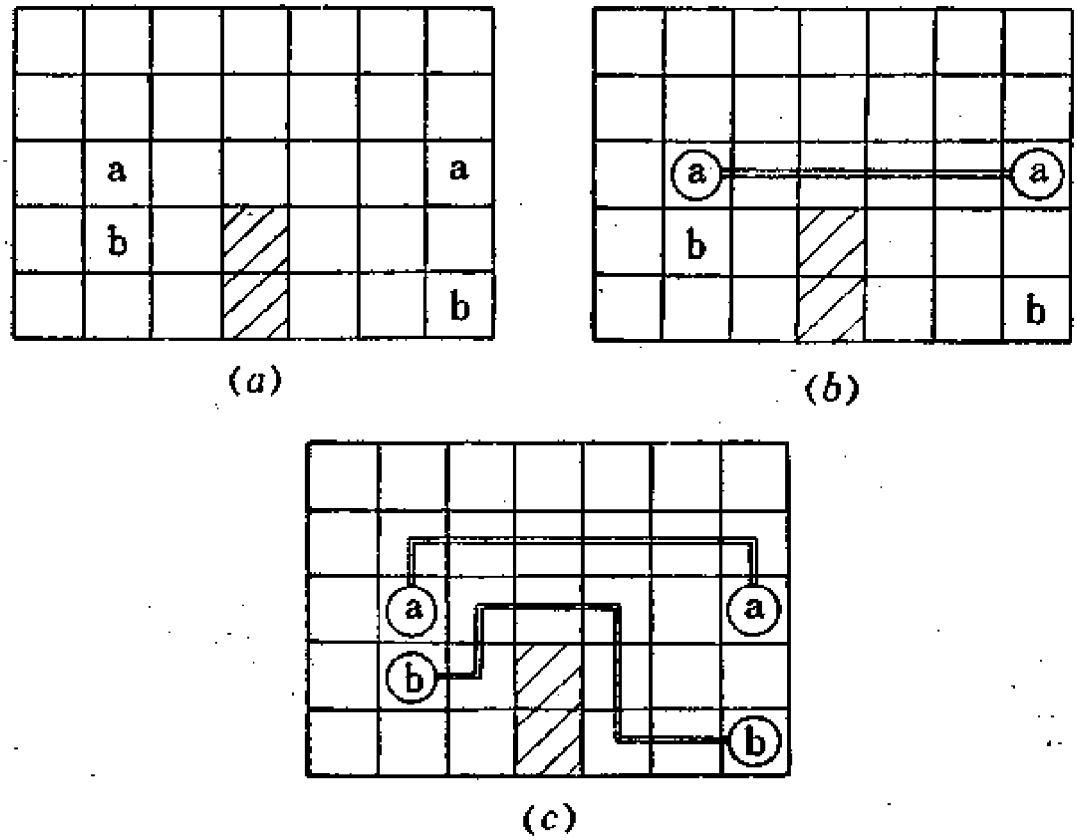


图 6.22　当前布线对未来布线影响的例一

在图 6.22 中，大多数布线器将先布线网 (*a*)，而这样就将使线网(*b*)无法实现连接。这就要求布线器应具有更高的智能，或是使线网(*b*)先布或是在线网(*a*)布线时预留下线网 (*b*) 的布线通道(如图 6.22(*c*) 所示)。

在图 6.23 中也有类似的情况，此时，当线网(*a*)以最短路径先实行布线后，线网(*b*)的接点将无法实现互连。然而这二种情况也

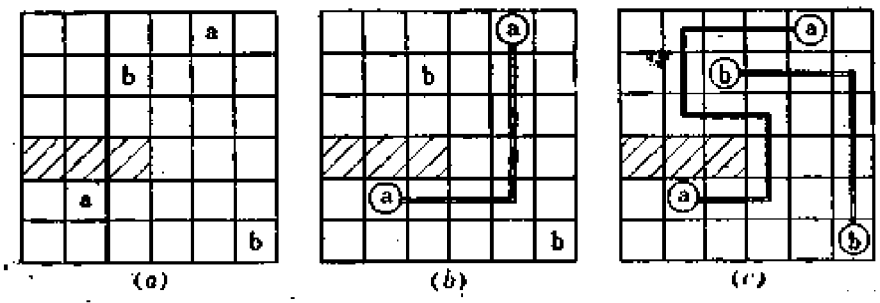


图 6.23　当前布线对未来布线影响的例二

有不同的地方。可以看到，在图 6.22(*b*) 中，要实现线网 (*b*) 的连接，要求线网(*a*)的原布线应适当地弯曲、向外推。换句话说，此时未布线网(*b*)的接点都在线网(*a*)的同一侧，它要求把线网 (*a*) 的已布线推向另一侧。而在图 6.23(*b*) 中，为了解决线网(*b*)的布线，就必须使线网(*a*)绕到线网(*b*)的接点的另一边去(此时，未布线网(*b*)的接点分布在线网(*a*)原布线的二侧)。

2. 整体布线的基本思想

针对上述情况，J.Soukup 提出了一种有趣的思想：即"同时"布线的想法，也就是整体布线的思想[102]。其设想为每条连线在最后确定前并不以线条的方式出现，而却象一条变形虫一样可以伸展、收缩，似乎是"活的"，而相互竞争实现接点互连所需的空间。开始时从每一个接点扩展为一个小的区域，并定义为相应的子线网，然后这些区域进行同时的扩展、移动并相互作用，直至同一线网的不同的子线网的区域相互接触时，形成一个连接。其它子线网在不破坏已有连接性的条件下，可以通过推动已有的布线去实现它们之间所需的连接。直至完成所有线网的布线为止。

整体布线的扩展过程非常象李氏算法，不同点是在这种布线思想中，不存在已布线所形成的那种"僵硬的"障碍，它们是连续地变化而且不断地相互作用，直至最后才以线条的形式实现所有线网的实际布线。

3. 整体布线的基本要求

为了使过程是可行的，在这种布线方法中必须满足下述要求：

① 待连接的线网应比已连接线网扩展得快。

② 其它子线网的扩展必须不破坏任何已连线网的连接性。

③ 为了避免振荡，不允许任何二个子线网具有相同的优先度。

④ 不允许一个子线网扩展到另一线网的接点所在区域中去。

4．“瓶颈”的概念

为了避免在子线网的扩展中把已布线网弄断（即上述要求②），关于“瓶颈”(bottleneck)的概念是很重要的。瓶颈是指网格场中的这样一些单元，如果把这个单元分配给其他线网，则该线网被分成二个或更多的含至少一个接点的子连通区。图 6.24 中阴影线画出的单元就是一些瓶颈单元。从图中还可以看到，若把阴影线画出的任一单元赋于其他线网，都将使线网 a 的连接性受到破坏，它将被分为二个各含一个接点的子连通区。

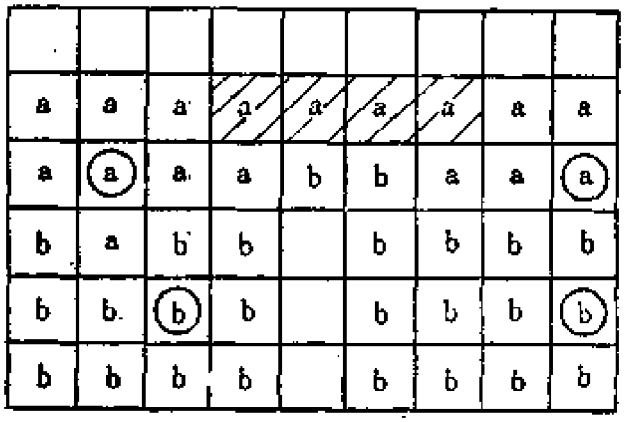


图 6.24　瓶颈单元示意图

瓶颈单元的严格判定是比较困难的，但判定一些单元肯定不是瓶颈单元却是十分简单的。如图 6.25(a) 中，对打*单元的判定，可考虑该单元相邻的 8 个单元，沿一定方向(如顺时针方向)搜索。若此 8 个单元中的线网号由中心单元的线网号变化为其他线网号或未用单元、障碍单元的次数小于 2 次，则该单元一定不是瓶颈单元。图 6.25(b) 中用此法判定后可知不存在瓶颈单元。值得注意的是，当不满足上述判定法则时，并不能确定该单元即是瓶颈单元(如图 6.25(a) 中打*单元)。此时尚需进一步判定：删除该单元后，原连通区是否被分为至少二个子连通区？图 6.26 打*单元不满足上述法则，但并没有将原连通区分为二个子连通区，因此，该单元并不是瓶颈单元。此外，还必需进一步判定断开的子连通区中是否至少有二个子连通区中是含线网接点的？图 6.26 打○的单

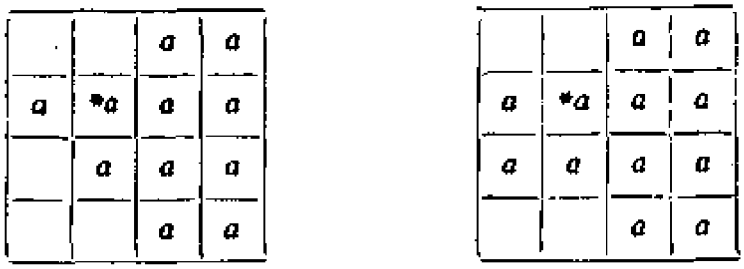


图 6.25　判定不是瓶颈单元方法示意图 1

元实际上也不是真正的瓶颈单元。

运用瓶颈单元的概念，一个子线网要扩展到另一个线网已占的单元中去必须满足下述条件：

① 该单元不是另一个线网的瓶颈单元。

② 扩展的线网应具有比已占该单元的线网更高的扩展优先度。

③ 该单元不是另一个线网的接点。

在这种布线方法中采用了三种优先度。

第一种：（最高优先级）未布通线网的子线网具有比已布线网

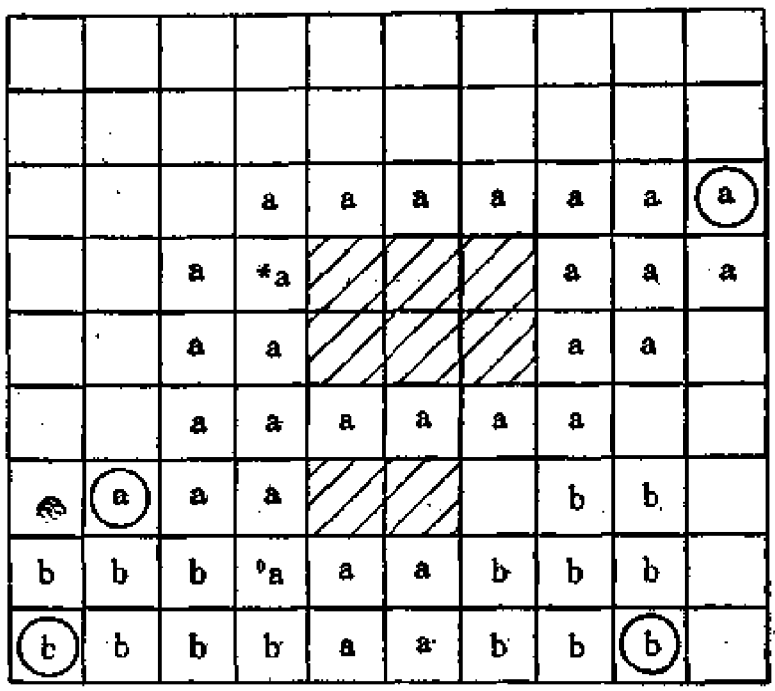


图 6.26　判定不是瓶颈单元方法示意图 2

更高的扩展优先度。

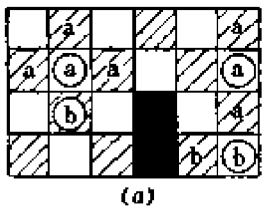
第二种：已布通线网的扩展优先度随其所占单元数的增加而减少。未布通线网的每个子线网随着它们之间的距离的缩短扩展优先度上升。

第三种：为了避免二个子线网具有相同的优先度，以线网名的字母顺序为扩展优先度序。

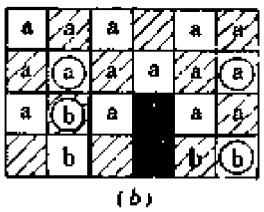
5．实际的布线过程

在上述布线方法中所要求的"同时"扩展，由于大部分计算机处理特性的限制，扩展时实际上是顺序进行的。但为了取得"同时"扩展的效果，我们可以把布线网格场着色成一个黑白相间的棋盘。首先扫描所有的黑格，进行所需的扩展，然后扫描所有的白格，进行下一步的扩展。图 6.27 中的六个图为图 6.22 布线问题的实际处理过程的图示(缩短了接点间的距离)。

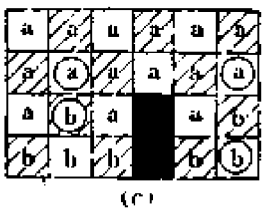
读者从上例中可看到线网 a 是如何收缩、变动而让出线网 b



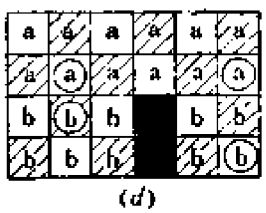
扫描全部黑格，此时线网 a 优先度高于线网 b



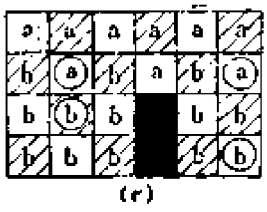
扫描全部白格，线网 a 实现了连接



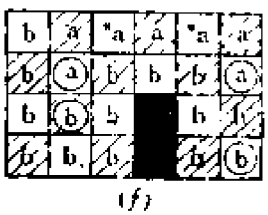
扫描全部黑格，线网 b 优先度高于线网 a



扫描全部白格



扫描全部黑格



扫描全部白格，打*单元为瓶颈单元，线网 a、b 都实现了联结

图 6.27　整体布线过程图示

实现布线所必须的通道，使线网 b 的布线也能够实现的了。

当应用上述方法处理图 6.23 的布线问题时，会发现线网 b 的布线仍无法实现。线网 b 的二个子线网分别充满布线场的左上角和右下角。然而，除非违反瓶颈单元不允许被扩展的限制条件，它们无法相互连通(注意，此时线网 a 优先度在未连通前高于线网 b)。

问题并不在瓶颈单元上，而是线网 b 必须把横在它们接点间的"墙"推到所有接点的一侧去。

解决这类问题的一种方法是，首先应用前述布线方法进行布线，对那些剩下的不能实现互连的线网，强制地规定同号的子线网中只有一个子线网可以扩展，而同号的其他子线网不允许扩展。这样就能自动地移动、推开成为"墙"的已布线网，为实现这些线网的联结提供必需的通道。图 6.28 为一个实际整体布线的示例图，图中所示的只是解决这类问题的方法之一，虽然对不少情况是有效的，但仍可能有些线网的布线还是不能完成。对这方面工作的进一步研究可望提出一些新的改进方法。

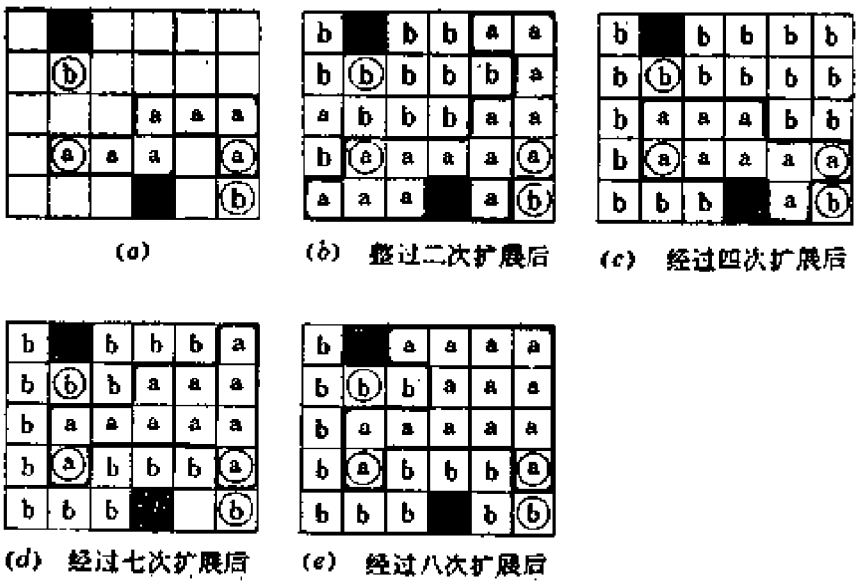


图 6.28 一个整体布线示例

J.Soukup 认为，这种布线思想原则上也可适用于任何多层的布线问题。

这种方法中最令人感兴趣的一点是：与以往的面向线网的布线方法不同，它不是“串行”的和仅考虑线网当前布线的局部优化，而是一种“同时”布线的方法，具有相当好的整体合理性。它在布线过程全部完成前，原则上任一线网的布线都是可调整、修改的。布线过程可望具有相当高的灵活性。应该说，这是一个很有特色的布线思想。

6.4 布线的顺序处理

李氏算法、线探索法等面向线网的布线方法很多都是一种“串行式”的布线方法，即按一定顺序依次对线网实行布线的方法。显然，为了获得尽可能高的布线完成率，在布线时，仅考虑使当前连线长度最短化是不够的，还必须考虑当前布线对今后布线的影响，即每次布线时，应考虑先布哪一条线网对今后布线较为有利。这就是布线的顺序问题。

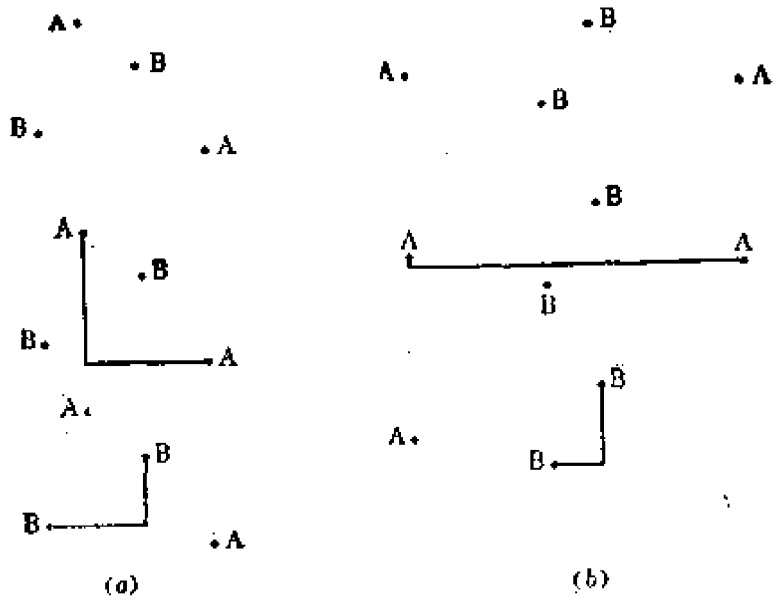


图 6.29 布线顺序对提高布线完成率的影响

图 6.29 说明了这个问题的重要性。在图 6.29(a)中，当 A 线网先布且拐角在下面，则 B 线的布线必然需"绕道"而行，使总的连线长度增加。但若 B 线网先布，则不论 B 线是哪条最短曼哈顿路径都对 A 线几乎没有影响，A 线仍可找到一条最小曼哈顿路径实现布线。在图 6.29(b)中当 A 线先布时，则 B 线将绕行较长的距离，总连线长将远大于 B 线先布时的情况。在实际应用中，顺序问题的处理是一个极其复杂的问题，对于复杂的布线问题，顺序处理将是提高布线完成率重要的手段。

一、一般的顺序处理方法

在要求不太高的布线问题中，大量地采用一些简捷的顺序处理方法。这些方法主要有：

1. 短线序法

这种方法采用了下述启发式原则：即接点间曼哈顿距离较小的连线先布后，对今后布线的影响较小。应用这种方法决定布线顺序时，首先定义各线网的尺寸 S_i，i 线网的尺寸 S_i 为覆盖该线网所有接点的最小矩形的半周长。布线时，每次选择当前未布线集中 S_i 最小者先布。

上述顺序处理原则实现方便，且对不少问题是有效的。显然，按上述方法处理时，S_i 越大的线网将越后布。而一般来讲，长的线网布线难度大，留到最后布，这些长线布线时就更困难了。因此，有人主张按"布线难度大的线先布"的启发式原则决定布线顺序，即每次布线时，选择未布线集中 S_i 最大者先布，即长线序。在有的问题中，长线序是有效的。因此，上述这些原则对不同的布线问题实际上效果是不稳定的。

2. 点干扰度序方法

点干扰度序方法[28]试图粗略地估计每个线网布线后受其影响的线网数，然后按其影响的线网数的多少来决定布线顺序。

A 线网的点干扰度 I_A 可定义为覆盖 A 线网所有接点的最小

矩形所覆盖的其它线网的接点数，布线时，每次选择未布线集中 I_i 最小者先布。图 6.30 为应用此原则决定布线序的一个例子，可以看到，图中 A、B、C、D 四条线网对应的 I_A、I_B、I_C、I_D 分别为 4，1，0，3，因此布线顺序为 C—B—D—A。图中并给出了最后的布线结果。

用这种方法处理得到的布线往往是互相嵌套的，即较短的水平和垂直线先布，然后布那些包围这些短线的较短的线网，最后布长线，它们总是布在其它线的外侧。正如其他启发式算法一样，这个原则也存在着反例。在有些情况下，还可能出现违反传递律的情况。如图 6.31 示为熟知的啤酒图案问题。图中 I_A，I_B，I_C 都等

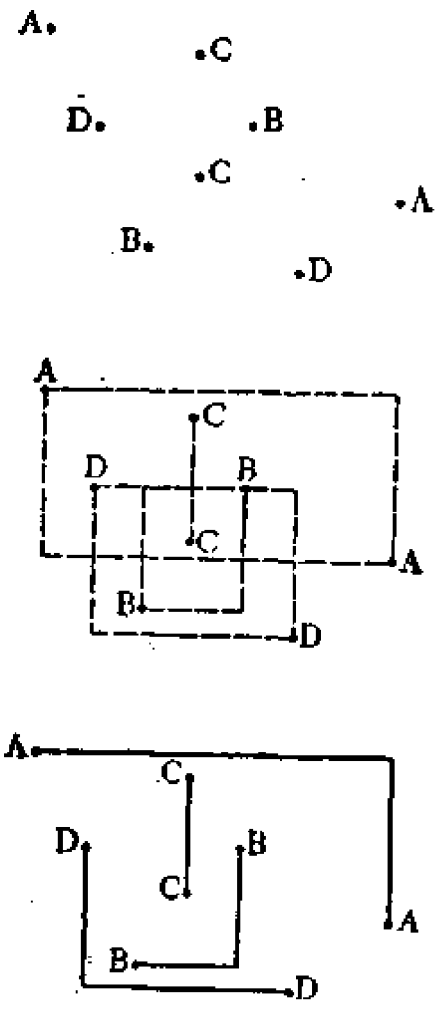


图 6.30　使用点干扰度最小者先布原则布线图

于1，而且实际上是A线要求B线先布，B线要求C线先布，C线要求A线先布。

3．线干扰度序方法

这种方法考虑的处理原则与点干扰度序很类似，但在度量每条线对今后布线的影响程度时，采用了线干扰度L_i。如图6.32所示，设线网B在以线网A的接点为顶点的矩形边上的投影分别为l_{vB}和l_{hB}。则L_A为$\sum_{Fi}(l_{Vi}+l_{hi})$。其中所有i线网都至少有一个接点处于覆盖A线网所有接点的最小矩形域内，布线时，每次选择L_i值最小者先布线，用这种方法处理时，可避免大部分违反传递律的情况。

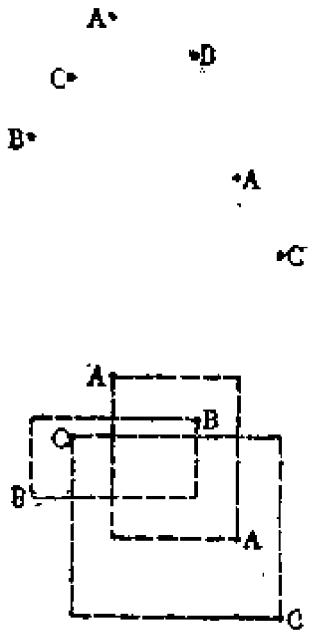


图6.31 啤酒图案问题

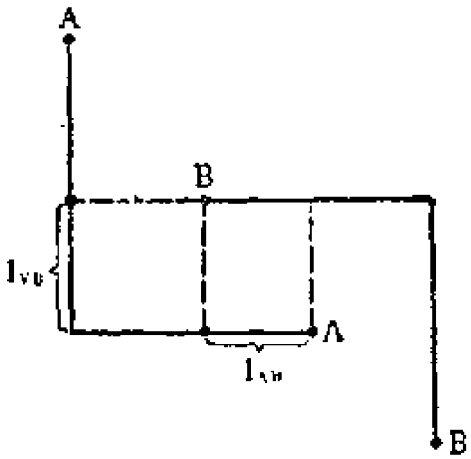


图6.32 线干扰度序方法示意图

上述各种启发式的顺序决定方法都具有处理简捷的优点，而且对不少实际问题是有效的，共同的问题是对不同的问题处理效果不稳定。对于复杂的布线问题，处理结果不理想。

4．布线的复杂性

关于布线问题的复杂性，有一个很有用的简单公式。对于一个已确定的布线问题，可很容易求得其接点间最小曼哈顿距离的

总和，设为 L。同时，对确定的布线区域也可方便地求得其可利用的布线通道总长。对于一个以单位长度（线宽＋线间距）划分的 $a\times b$ 的网格场，设其所有的通道都是可利用的，则通道总长 S 为 $a*(b+1)+b*(a+1)=2ab+a+b$。可以看出，比值 L/S 是描述布线问题复杂性或难度的一个有用的量。一般当 $R=L/S$ 大于 0.33 时，面临的将是一个比较困难的布线问题。当 R 值足够大时，如 $R>0.45$，一般布线方法的布线完成率仅为 80% 左右，需要采取一些更好的算法才可望进一步提高布线完成率。当 $R<0.33$ 时，是一个比较容易的布线问题，尤其当规模比较小时，较好的布线方法可获得 99% 以上的布线完成率。

二、一种基于全局分析的顺序处理方法

1972 年王守觉基于线间关系的综合分析，提出了一种布线顺序的决定方法。

在实际布线中，由于各线网接点的相互位置和对通道的要求，使各线网在布线时存在着各种顺序约束条件及布线方式的约束条件，为了获得尽可能高的布线完成率，必须对上述线网间的关系进行较严格的分析。

从本质上来讲，线网间的关系是极其复杂的。为了分析的方便，我们首先分析平面上二条两接点线网间的关系，也可称作单线间关系的分析。

平面上二线由于它们不同的接点相对位置，可有 548 种可能的情况。当分析线 A 与其他线的关系时，可限定先考虑线 A 接点 (x_{A1},y_{A1})；x_{A2}，y_{A2} 满足 $x_{A2}<x_{A1}$ 且 $y_{A2}>y_{A1}$ 或 $x_{A2}>x_{A1}$ 且 $y_{A2}<y_{A1}$ 的情况（另一类情况可方便地进行类推）。此时可能的情况有 274 种。

1．单线间关系的分类

下面被分析线 A 简称为本线，线 A 外的其它线称为另线。存在着三种最基本的关系：

(1) 无关情况

无关情况也称任意情况，其含义是当另线 i 无论走哪条最小曼哈顿路径时，都不影响本线以最小曼哈顿路径布线，如图 6.33 所示。

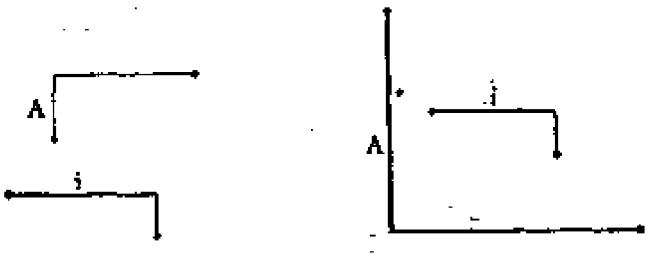


图 6.33 本线布线与另线路径无关的情况

(2) 约束情况

当存在布线约束条件的情况时，首先让我们对一些将用到的名词给出一些定义。连接线 A 二接点 A_1 和 A_2 的最多只有一个拐弯点的最小曼哈顿路径有二条。若其拐弯点的 y 坐标等于 $\mathrm{Max}\{y_{A1}, y_{A2}\}$ 则称该条路径为上行路径，若 y 坐标等于 $\min\{y_{A1},$

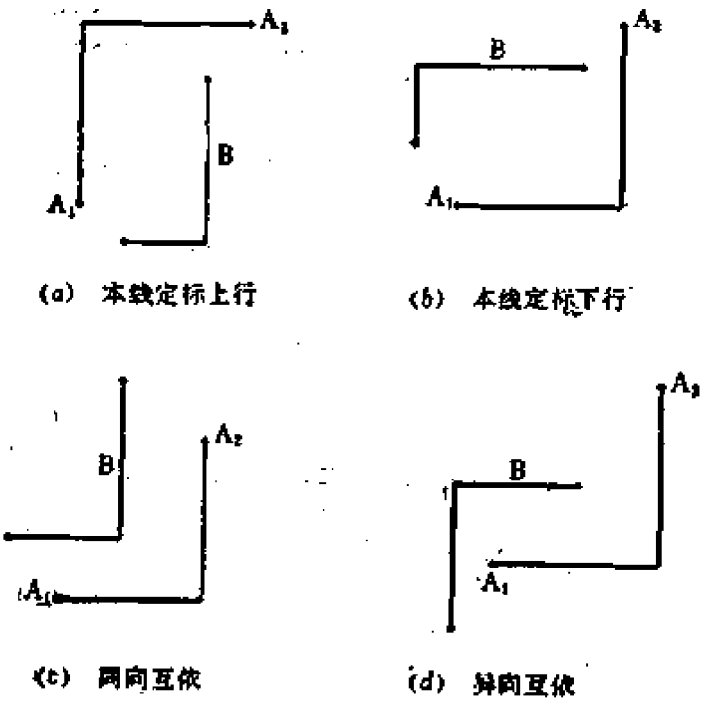


图 6.34 同向互依及异向互依的情况

$y_{A2}\}$，则称为下行路径。从上述二条路径中选定一条的过程称作定标过程，并可用一定标来描述这条路径。当定标值为二进制 01 时，为上行路径。当定标值为 10 时，为下行路径。

图 6.34 中给出了本线定标上行、定标下行以及本线与另线定标要求相同和要求相反(即同向互依及异向互依情况)的图例。

(3) 绕行情况

无论另线按哪条最短曼哈顿路径先布线，本线都将不可能按最短曼哈顿路径实现布线，图 6.35 给出了一些图例。

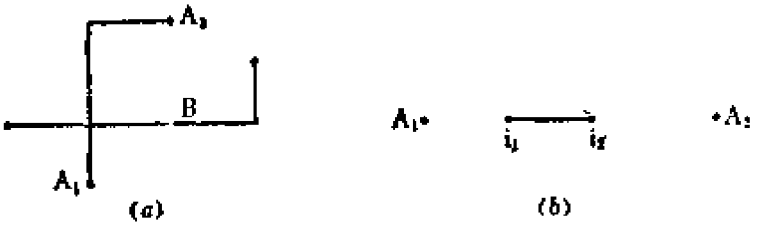


图 6.35 另线先布对本线的影响

实际上二线在进行分析时，常常是相互对对方有一定的要求。对各种不同的接点相互位置，我们都可列出其相应的要求来，如图 6.33 A 线与 i 线间的关系为"本任另任"，意思是另线 i 对 A 线为任意情况，而本线 A 对另线 i 的要求也是任意情况。同样，图 6.34(a)中 A、B 线的关系为"本上另任"。(b)中为"本下另任"等等。此外还存在"本上另下"(如图 6.36(a))及"本任另绕"(如图 6.36(b))等等情况。

2. 单线间关系的分析方法

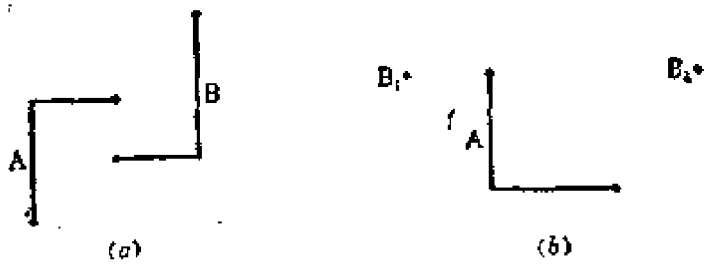


图 6.36 "本上另下"和"本任另绕"情况

可以看到：二线间的关系实际上是其接点相对位置的某种函数。因此，只要给出二线接点相对位置的严格描述，就可以根据一定的映射关系确定它们的线间关系。

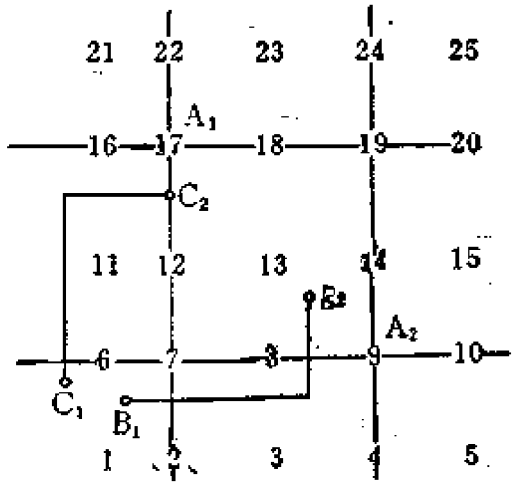


图 6.37　单线间关系分析法示意图

在平面上，当我们从本线 A 的二接点出发分别作与数轴平行的四条直线，可将平面分为 25 个区域，其中 9 个面域，12 个线域，4 个点域，如图6.37示。并可按一定顺序给各区域编号。定义各区域的域值 P_n 为 2^{n-1}，其中 n 为区域的编号。这样平面上线 A 之外的任一线的接点与本线 A 接点的相对位置可用其接点所在域的域值和来表示。如图中 B 线对应的域值和 $RPV_B = P_1 + P_{13} = 2^0 + 2^{12}$，同样 $RPV_C = P_1 + P_{12} = 2^0 + 2^{11}$。通过对 RPV 值的分析，其可能的值的总数 N 为 $N = C_{25}^2 + C_{25}^1 = 325$。其中，以 9，17 为接点的另线，是与本线直接相连的自连线情况，共有 $24 + 23 = 47$ 种情况，另线二接点同在 7，9，17，19 点域内为不可能情况，共 4 种，因此实际可能的值共有 $325 - 47 - 4 = 274$ 种（注意：当本线 A 退化为一条水平线段的情况时，实际上即为 6，……15 区退化消失的情况，处理过程仍可适用）。这样，对于另线每一种可能的接点相对位置我们都可以求得其相应的 RPV 值，通过归并译码，便可确定它和本线 A 的相互关系。如用测量值二进制 11111　11111　11111　11111　00000 与另线的 RPV 值进行逻辑乘，当结果等于 0 时，表明本线与该另线的关系为“本任另任”。而若用测量值二进制数 00000　11111　11111　11111　11111 与另线的 RPV 值进行逻辑乘，当结果等于 0 时，也表明本线与该另线的关系为“本任另任”。这二个测量值可用来判定 30 种情况。当另线的 RPV 值与上述二个测量值逻辑乘结果不为 0，而

与假定的测量值 10001 11111 11111 11111 10001 逻辑乘结果为 0，则表示它们的关系为"本绕另绕"，等等。所以，只要进行适当的归并，并设计相应的测量值和按一定顺序将另线的 *RPV* 值依次与测量值作逻辑乘，首先得到 0 结果的序次就对应着该另线和本线的关系。可见，分析的方法是相当简捷的。

3．线间关系的统计处理

设线网以某个序次排列(该序次可以是任意的)。依次取一线网为本线，并与所有序号大于该线的另线进行比较，分析，判定它们的线间关系。对于每条线网，分别统计三类信息：

(1) 定标信息

初始时，该信息记录为 0，当在关系判定时，出现定标要求，则把定标值与该线(本线或另线)的定标信息逻辑加。同时并记录每条线网、每类定标要求的数量。

(2) 绕行数记录

初始时，该记录内容为 0，当在关系制定时，出现绕行要求，则把相应线(本线或另线)的绕行数记录值加 1。

(3) 互依关系统计

当在关系判定时，出现互依要求时，记录有关的线网序号及它们互依的类型(同向还是异向)。

当所有的线分析完毕后，首先处理具有互依要求的线网。可用一无向图来描述线网间的互依要求，图中顶点对应着线网，当二线网间存在互依要求时，在它们对应的顶点间加一条边，并表明其互依类型。

第一步从无向图中删除所有绕行数不等于 0 的线网所对应的顶点。第二步对于每个连通片进行定标要求处理，当连通片中的若干顶点对应的线网具有一类定标要求时(即定标值结果为 01 或 10 时)，以这些顶点的定标要求推定与其邻接的顶点的定标要求，并将其相应的定标值逻辑加到邻接顶点的定标记录中去，并记录其各类定标要求的数量。当连通片中所有顶点都无定标要求或定

标值结果为11时，可对连通片中任一元素给定一定标要求（如上行），然后依次推定连通片内各顶点的定标要求，并予记录。

经过上述处理后，全部线网可分为三类：

① 与其它线都为无关情况或其它线对它只有一种定标要求（即定标值结果为00，01，10）的线网。

② 其它线对它有矛盾的定标要求的线（即定标值结果为11的情况）。

③ 与其它线有绕行关系的线，即其绕行数统计值不等于0的情况。

4．布线顺序的确定

根据上述统计结果布线顺序的确定是十分简单的。

① 先布定标值结果为00，01，10的线（且其绕行数统计值为0）。这类线的布线序与布线结果无关，当存在定标要求时，则直接依其定标要求布线。

② 再布定标值结果为11且绕行数统计值为0的线。设要求其上行和下行的线数分别为U_N和D_N，可定义其中每条线网的先行权$P=\mathrm{Min}\{U_N, D_N\}$，按先行权由小到大的顺序布线。

③ 最后布绕行数统计值不等于0的线。每条线网的先行权P可定义为其绕行数统计值，然后依先行权由小到大顺序布线。一般认为，布线完成率和完成布线时的连线总长之间有较好的对应关系。从这个思想出发，我们可定义一个与连线总长有对应关系的通道损耗的概念，设连线A的接点间最小曼哈顿距离为L_M，由于其它线网布线的影响，A线实际布线时的最短路径长度为L_A，则A线的通道损耗$S_A=L_A-L_M$。显然，连线总长最短化的目标和使线网布线后通道损耗总和最小化的目标是等价的。

作者基于上述平面上线间关系分析的方法，以通道损耗总和最小化为目标，具体实现了上述布线算法[103]，对于一些复杂的布线问题（最多线数为1049，$R>0.45$）布线完成率可比一般的定序方法（程线序，点干扰程序），可提高10%以上。虽然分析定序

时，将耗费一定的机时，但由于一部分线网布线方式的预先确定，使实际布线效率(以李氏法为实际布线算法)得到提高，而总的处理效率仍可与一般方法相当。这个方法当扩展到处理多点线网时，算法将需要作一定的修改和补充。

由于从本质上来讲，线网布线顺序问题是一个极其复杂的问题，而且除布线顺序问题外，还有不少其它因素，如布线通道的分配，布线区面积的利用合理性等都对布线完成率有很大的影响。因此，对于复杂的布线问题来讲，面向线网的布线方法在更进一步地提高布线完成率方面必然受到一定的局限。在第七章内我们将介绍另一类布线算法——面向布线区域的布线方法，以获得更理想的设计结果。值得注意的是，本节介绍的关于线间关系综合分析的思想(基于总体分析基础上的布线思想)在面向布线区域的布线方法中也同样具有重要的意义。

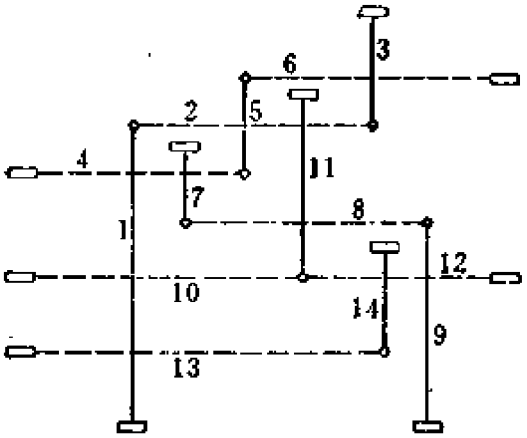
6.5 分层及通孔最小化算法

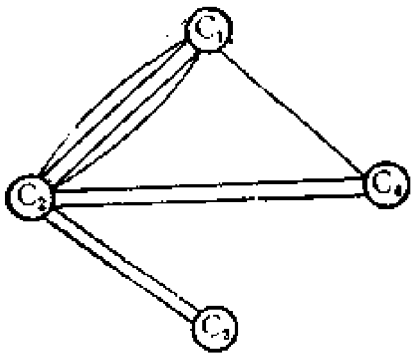
在集成电路的布线问题中，大量的是二层布线问题。当限定布线通过曼哈顿路径实现时，为了获得尽可能高的布线完成率，一种令人满意也是经常采用的方法是在一层上全部用来布连线所含的水平线段，而在另一层上全部用来布连线所含的垂直线段(即横竖分层原则)。二层间必须的连接用通孔来实现，这种方法对提高布线完成率是很有效的。6.3 节提到的线探索法中先布线后分层的原则，实质上也是以横竖分层原则作为其可行性的保证的。但是当实际结果采用严格的横竖分层原则时，由于很多线网将要求在二层上都布线而需要很多通孔。我们知道，在集成电路生产中，集成电路的成品率将随着通孔数量的增加而降低。这样，放在我们面前的需解决的是这样一个问题：对于一个已经完成的布线结果，如何在不改变所有线段的平面位置的条件下，求得一个合理的分层使所需的通孔数量最小化。

一、相交图与色图理论

图 6.38 为一个简单的布线问题的结果。采用横竖分层时共需 8 个通孔。

为了讨论的方便，我们首先引入一些定义和有关的理论结果。





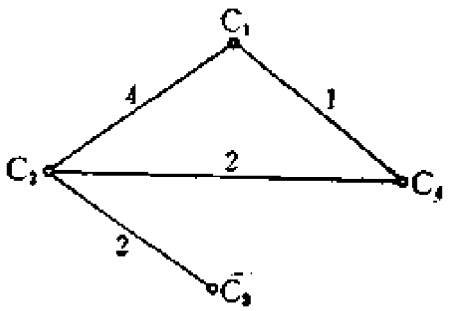


图 6.38　采用横竖分层的布线结果图

*D*1——相交对的定义。非同一线网的相交的垂直线段和水平线段对称为相交对。

*D*2——相交集的定义。令相交对 (i, j) 为相交集 C_i 的初始元素，则任何与相交集 C_i 中任一相交对有一个共同元素(线段)的相交对都是该集的元素。当 C_i 中包含了所有满足上述条件的相交对时，称 C_i 为第 i 个相交集。如图 6.38 存在四个相交集：

$C_1 = \{1, 4\} \cup \{1, 10\} \cup \{1, 13\} \cup \{4, 7\} = \{1, 4, 7, 10, 13\}$；

$C_2 = \{2, 5, 8, 11\}$；

$C_3 = \{3, 6\}$；

$C_4 = \{9, 12, 14\}$；

*D*3——连接对的定义。同一线网相连的一对水平线段和垂直线段称为一个连接对。如图 6.38 共有(1, 2)；(2, 3)；(4, 5)；(5, 6)；(7, 8)；(8, 9)；(10, 11)；(11, 12)；(13, 14) 等 9 个连接对。

*D*4——相交图模型的定义。相交图中顶点对应着相交集，当一个连接对的元素分属二个相交集时，在该二个相交集对应的顶点间连接一条边。也可简化为一个带权的无向图，此时顶点 i 和 j 之间的边权数等于元素分属 i，j 相交集的连接对的个数，如图 6.38 示。

*D*5——色数的定义。将图 G 的顶点涂上颜色，使每条边所联结的结点不具有相同的颜色，所需用的最少的颜色数称为图 G 的色数。已经证明：任何平面图的最大色数等于 4。

D 6——偶图的定义。如果能将图 G 的顶点集划分为二个子集 V_1 和 V_2，使得 $V_i(i=1, 2)$ 构成的子图中不存在联结本身顶点的边，则称图 G 为偶图或二分图。可以证明：当且仅当一个图不具有奇数长度的循环(回路)时，该图为偶图。其证明可采用凯尼格定理(Koerig's theorem)，即当且仅当一个图不具有奇数长度的回路时，这个图才可以用二种颜色来涂色，使每边联结的顶点都是不同的颜色。

D7——色边权、色度、割色度的定义。对图G顶点任意用二种颜色给以着色后，连接不同颜色的顶点的边称之为本征边，各边的色边权M_{ij}定义为：

$$M_{ij}=\begin{cases}+W_{ij} & \text{若边 } E_{ij} \text{ 为本征边时}\\ -W_{ij} & \text{若边 } E_{ij} \text{ 为非本征边时}\end{cases}$$

其中：W_{ij}为边E_{ij}的边权；

顶点i的色度d_i定义为$d_i=\sum_{\forall j}M_{ij}$，

其中j为与i顶点邻接的顶点。

图G的一个切割的割色度C_e可定义为它所切割的所有边的色边权的和。可证明下述的最佳着色条件定理：即当且仅当在图G中的一个着色方式使其每个切割的割色度都大于、等于0时，其非本征边的边权和为最小，也即为最佳着色。

该定理的证明是简单的。割色度大于、等于0意味着该切割的所有边中本征边的边权和大于、等于非本征边的边权和。若图中有一个切割不满足上述条件，且割色度小于0，则只需将该切割一侧的所有顶点涂以相反的颜色，这样，切割的所有边（并且仅仅是这些边）的本征性质将发生相反的变化，使非本征边的边权和减少，因此原图着色不是最佳着色。相反，若一个图的着色已满足上述条件，设改变其中一个顶点子集的着色，由于原着色中，与该顶点子集对应的割色度大于0（割色度等于0的情况实质上说明了该图最佳着色解的不唯一性），则改变颜色后，该子集对应的割色度将小于0，即非本征边的边权和将增加而破坏了最佳着色。

二、通孔最小化算法[104]、[105]、[106]

根据相交集的定义，若一个连接对的二个元素同属于一个相交集，则这个连接对所需的通孔是无法消除的，因此这种通孔也可称为本质通孔。

如果我们对相交图的顶点用二种颜色着色，红色顶点表示该

顶点对应的相交集中水平线段布在上层，垂直线段布在下层，黑色顶点表示该顶点对应的相交集中水平线段布在下层，垂直线段布在上层。这样与本征边对应的通孔都是可以消除的，而与非本征边对应的通孔是必需的。由于这些与边对应的通孔可以通过不同的着色(分层)来调整，因此，这类通孔也可称之为非本质通孔。

根据凯尼格定理，如果相交图是一个偶图，则所有的非本质通孔都可望通过简单的分层处理而全部消除。但不幸的是，在实际的布线问题中，相交图恰好是偶图的情况，其概率几乎等于0。因此，对实际的二层布线问题来讲，更实际的问题是如何对相交图进行着色(分层)，使非本征边的边权和最小(或者讲使所有顶点的总色度最大)，其实际意义是使不能消除的非本质通孔数最小。

显然，这个问题可以容易地转化为下述问题：如何移去边权和最小的一个边的子集，使相交图中不存在任何奇数长度的回路(或使相交图成为一个偶图)；如何找出一个具有最大边权和的偶子图；如何把顶点分为"发送机"和"接收机"使之能发送最多的信息等等。

这些问题的精确解是一个复杂的难题。存在着许多启发式的算法，试图以较高的处理效率求得令人满意的结果。

首先我们根据最佳着色条件定理，可给出一个形式上的算法。

1．最佳着色算法

① 对相交图G各顶点用二种颜色任意地着色。

② 对图G所有可能的分划进行检测，计算其割色度C_s。若$\forall C_s \geq 0$，已获得最佳着色，过程结束。否则将其$C_s < 0$且绝对值

2. 准最佳着色算法

该算法限定所检测的G的分划的最大元素为2。当顶点数$\leqslant 5$时，可获得相交图的最佳着色。否则其结果是准最佳化的。

① 对相交图G各顶点用二种颜色任意地着色。

② 计算图G各顶点的色度d_i，若$\forall d_i \geqslant 0$，转4；否则转3。

③ 将d_i中$d_i<0$且绝对值最大的顶点改变着色，转2。

④ 按d_i递升的顺序排列所有的顶点，计算d_{ij}。

$$d_{ij}=d_i+d_j-2M_{ij}$$

其中j为：

$$j\in\{\text{顶点号}\mid j>i\ \text{且}\ M_{ij}>0\}$$

若$\forall d_{ij}\geqslant 0$，过程结束，已求得准最佳着色。否则，若$d_{ij}<0$，改变顶点$i$，$j$的着色，转2。

3. 拟最大偶子图算法

① 将图G的各边依其边权大小由大至小排列。

② 以图G的所有顶点作初始子图G'。

③ 按边权序逐次将各边加入到G'中去。从第三条边开始，每次增加一条边时，检查G'中是否出现奇数长度的回路，若出现上述情况，该边不加入G'；若不出现新的回路或出现偶数长度的回路时，则该边加入G'图。上述过程一直进行到所有的边处理完为止，求得的G的子图G'是最大偶子图。

图6.39为图6.38中的布线结果经准最佳着色算法或拟最大偶子图算法处理后的分层结果，所需通孔数由8个减为1个。

在图6.39中，可以看到，相交图的边权和和非本质通孔数并不对应。这是因为非本质通孔A对应着二个连结对(10,11),(11,12)。只有当这二个连结对对应的通孔都是非本质的且其对应的边都为本征边时，此通孔才可被消除。一个通孔最多可对应着4个连结对。一个方便的处理方法是在定义边权重时，定义每个连结对的权重=1/(该连结对对应的通孔相关的连结对总数)，而当一个通孔为本质通孔时，其相关的各连结对边权都定义为0。

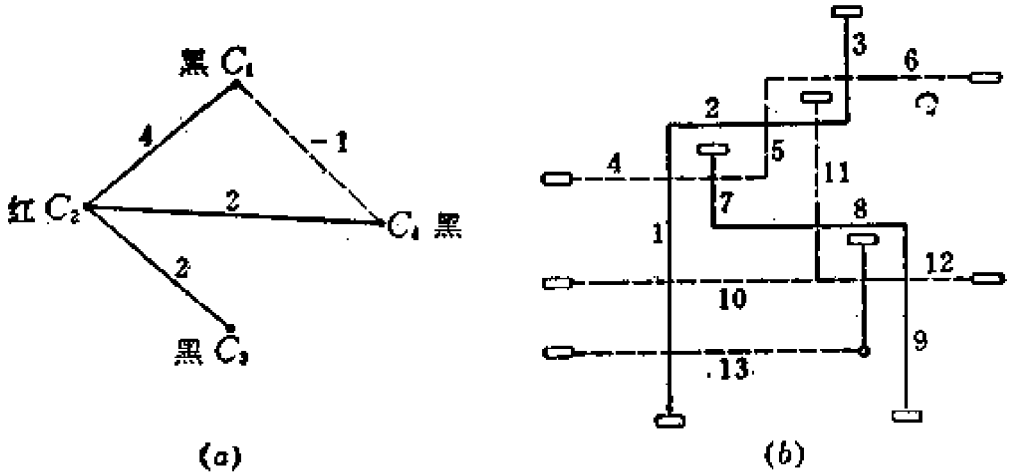


图 6.39　通孔最小化的处理结果

可以看到，本节介绍的通孔最小化问题不仅在应用面向线网的布线方法时存在，在应用其它布线方法时，也存在，而其基本的最小化方法具有普遍的适用性。可以说，通孔最小化方法是实际布线设计中的一个重要的方法和必须予以考虑的问题。

6.6　面向线网布线方法小结

本章介绍了一些主要的面向线网的布线方法以及有关的问题，大部分方法都能不同程度地成功地用于实际布线问题，实际应用中也常常把有些算法结合起来应用，常见的是把线探索法和李氏算法结合起来应用，以获得较高的布线效率和较满意的布线结果。

一、面向线网的布线方法的特点

本章介绍的这些方法都是从线网的角度出发设计的，因而在布线时，较好地考虑了每条线网的优化布线。同时，大多数方法都以连线长度最短化作为目标函数，因此，适用于对连线长度要求严格的一类布线问题。如某些高速电路的布线问题。此外，面向线

网的布线方法往往对各种工艺条件、电学要求有较强的适应能力。

二、布线设计目标的讨论

我们知道，在集成电路的布线设计中，布线完成率或布线设计后芯片的面积往往是设计者深为关注的。大量的布线实践告诉我们，连线总长最短化与上述目标虽然有较好的对应关系，但并不是完全对应的，尤其是当连线长度达到某一临界值后，更是如此。图6.40表示的是连线总长与布线完成率或芯片面积的一种经验关系。这里重要的原因是：布线完成率和芯片面积与布线的均匀性及布线区面积的利用率有着密切的关系。为此提出了很多其它的布线算法，我们将在下章详细地予以讨论。

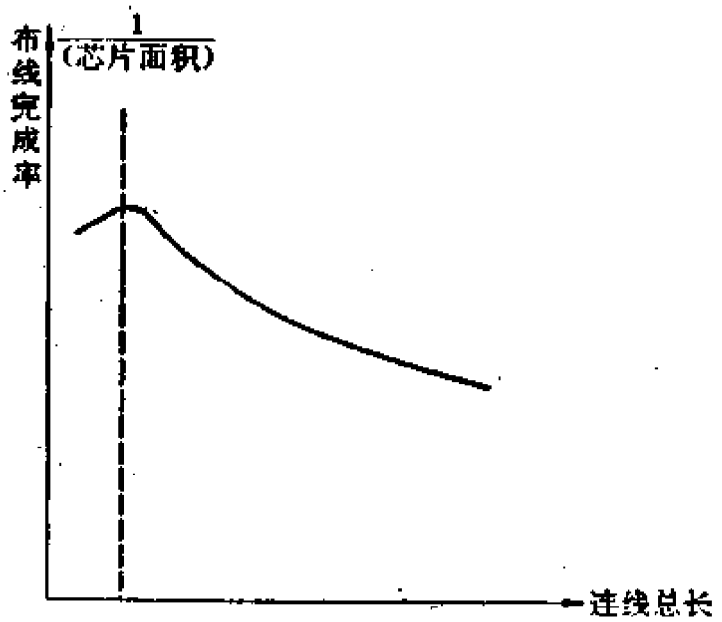


图 6.40　布线完成率或芯片面积与连线总长的关系

三、面向线网的布线方法存在的问题

1．“死线”问题

在规模比较大的较复杂的布线问题中，几乎所有的面向线网

的布线方法都难以保证 100% 的布线完成率。总有一些线网"无法连接"，有时称这些线网为"死线"。死线处理涉及到一些重要的而尚未得到很好解决的问题。这些问题是：

① 如何拆去一部分已布线才能使拆下的线网和死线都能完成布线。

② 重新布线时采用什么方法才好。

近年来，有人建议，应用人工智能的方法来解决"死线"重布问题。但目前通常的方法是由设计者通过和计算机对话，用人机交互的方式进行处理。因此具有一个良好的人机交互系统，包括图象显示设备是很重要的。

2．效率问题

布线效率低也往往是面向线网的布线方法在实际应用中存在的一个问题。

3．资源问题

通常，面向线网的布线方法在实际应用时往往要求计算机具有较大的、甚至是很大的存贮容量。对于 LSI 尤其是 VLSI 而言，这将使面向线网的布线方法的实际应用受到很大的限制。

第七章　面向布线区域的布线方法

在 LSI/VLSI 中，布线问题由于规模的增大（一般线网数为 10^3～10^4 的数量级）和复杂性的增加，使得面向线网的布线方法在解决所有线网在布线时的相互影响上遇到了很大困难。1971 年后，在布线问题上引入了“分级布线”(Hierarchical routing)的概念，即通过适当的划分，把整个可布线区域划分为若干规则的布线通道区（Channel），然后通过基于全局考虑的适当的布线分配（总体布线——Global routing or loose routing），把所有布线合理地分配到各布线通道区中去。注意，此时分配在各布线通道区的互连线往往仅是一些线网的一部分，而且仅仅是确定了这些子线网的布线通道区属性，而并没有在布线通道区中最后地完成布线。最后才按一定的顺序[215]解决各个布线通道区中的布线问题。这种布线思想把规模庞大的、复杂的布线问题分解为若干个规模都大大缩小了的子问题，从而为从全局考虑出发，更合理地解决布线问题提供了可能性。实践证明，这种设计思想是行之有效的。它具有布线效率高，设计精度高（布线完成率高或设计结果芯片面积小）和对计算机资源要求低等明显的优点。因此，在现有的 LSI/VLSI 布图设计自动化系统中广泛地采用了上述布线设计思想。如贝尔实验室的 LTX 系统，IBM 公司的 EDS 系统，以及 Hughes Aircraft 公司的 HAL 系统和西门子公司的 AVESTA 系统等等。

上述分级布线思想在实现时要求布线层次在二层或二层以上，目前一般为二层，原则上可适用于现有的任何一种集成电路工艺，尤其是适合目前在 LSI/VLSI 发展中极有前途的低功耗、高集成度的 MOS 工艺。这时，二个布线层可分别为铝线层和多晶硅

层，也可以是二层铝或其它金属层。

显然，采用上述分级布线思想实现布线设计时，主要考虑的问题是如何合理地利用布线区域完成全部布线。换言之，与大部分面向线网的布线方法不同，它的目标往往不是简单的连线总长最小化，而是如何实现布线使所需布线通道区面积和芯片总面积最小化(在多元胞和任意元胞模式中)，或者是如何合理利用已确定的布线区域尽可能获得100% 的布线成功率（在门阵列母片式模式中)。

因此在面向区域的布线方法中必须解决的问题有：

① 如何合理地把芯片上可布线区域划分为布线通道区？

② 如何合理地完成线网的分配即进行总体布线？

③ 如何实现各布线通道区中的最佳化布线？

下面我们将分节予以讨论。

7.1 布线区域的划分

由分级布线的思想可知，布线区域的划分应使划分得到的布线通道区比较规则，便于通道区中最终布线的实现，同时使总体布线易于进行。在目前的实际系统中，布线区域划分的主要方式有：

一、多元胞模式中布线区域划分方式

对于具有子块概念、且子块以行列式排列的多元胞模式[5]布局设计结果来讲，子块列之间以及子块列和芯片边界之间的区域可定义为垂直通道区。相邻子块间的水平区域以及子块行和芯片边界之间的区域可定义为水平通道区。这样划分得到的布线通道区都是一些矩形或近似矩形的区域(由于各子块长度的细微差别，垂直通道区往往不是严格的矩形)，而且这些区域的大小可根据布线的实际需要而加以调整(图7.1)。

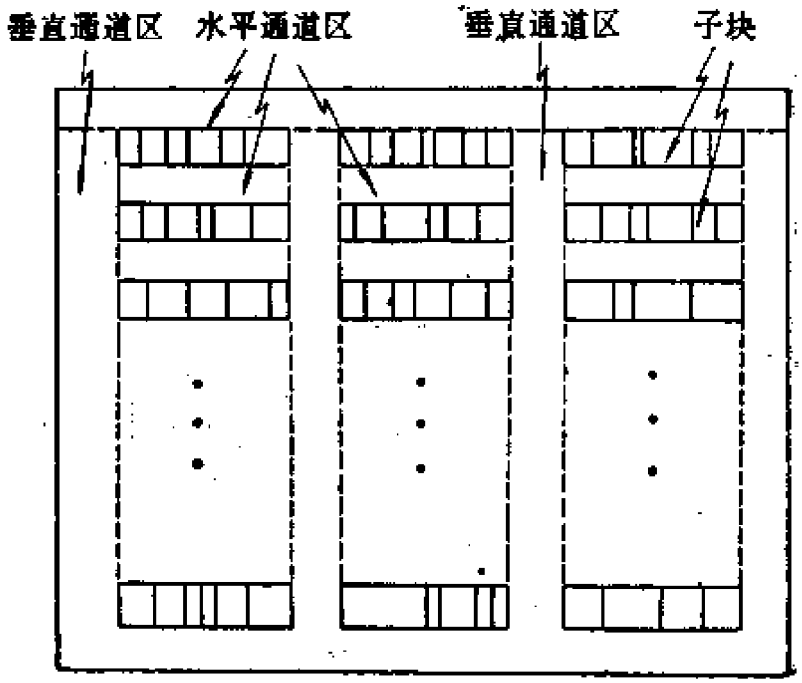


图 7.1　以行列式形式排列的多元胞模式

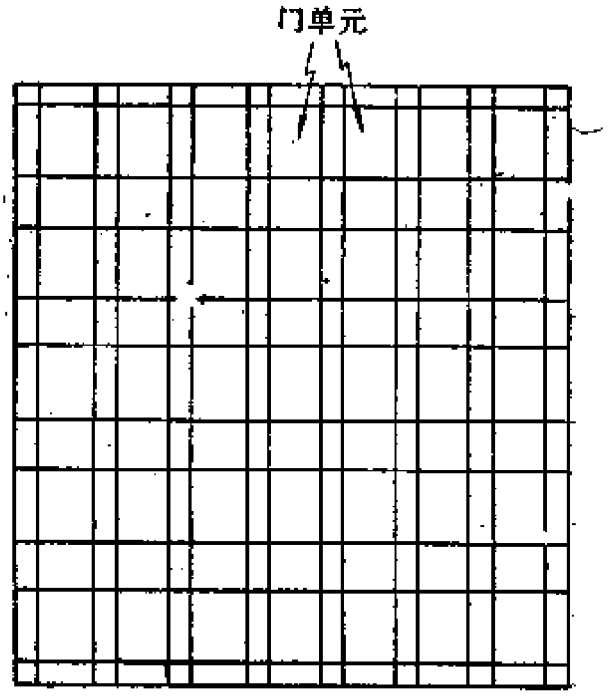


图 7.2　门阵列模式布线区域的划分结果

二、门阵列母片式中布线区域的划分方式

图 7.2 所示为一般门阵列模式中布线区域的划分结果[113]。它以门单元为基础，把芯片划分为一系列规则的矩形区域(也称总体布线单元)。

由于母片式的设计特点，这些矩形区域的大小和形状，在布线时都是不可改变的。

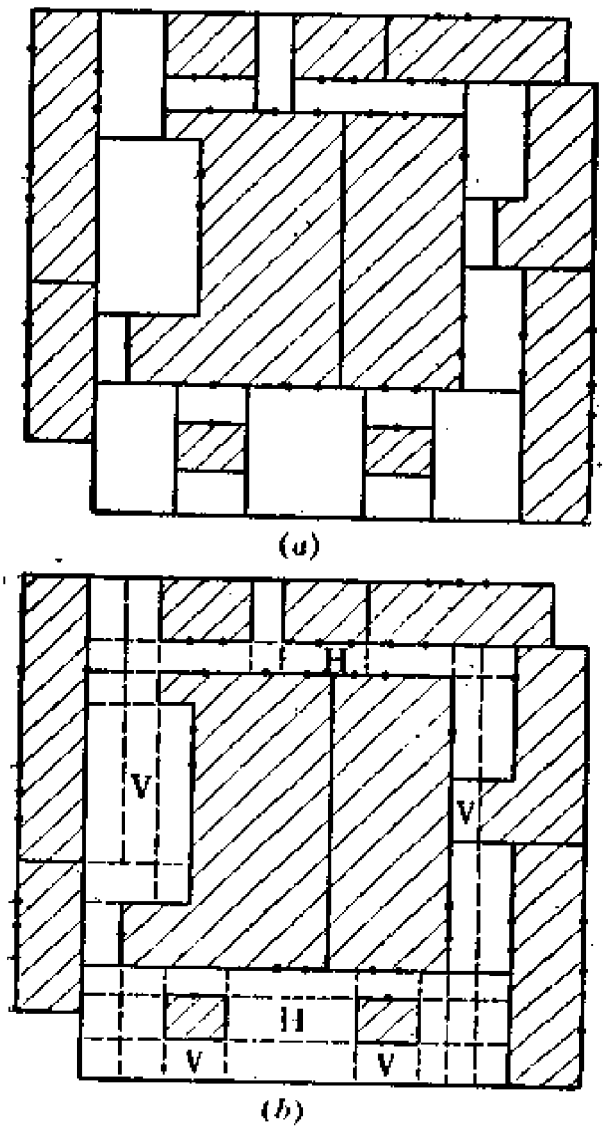


图 7.3 任意元胞模式布线区域划分方式

三、任意元胞模式中布线区域的划分方式

在任意元胞模式中，由于单元形状的不规则性，在不同的系统中，存在着各种不同的布线区域的划分方式。其主要的划分方式有下述二种[3]：

① 贝尔实验室一个新的系统首先把布线区域划分为一系列小的矩形域，然后通过人工或自动的方式把它们结合起来形成若干布线通道区，如图 7.3(a) 所示。

② 麻省理工学院 (M.I.T) 采用了另一种布线区域的划分方式，在他们的 PI 系统中，通过把单元的边界进行延伸，从而把布线区域划分为一系列小的矩形区域，如图 7.3(b) 所示。

对应于任意元胞的布局设计结果和布线通道区的定义，可画出其相应的 x 方向和 y 方向的位置图(可参看第五章)。显然，影响设计得到的芯片面积的主要参数是二个方向关键路径长度的乘积。因此在任意元胞的总体布线中，主要目标是如何合理分配线网，使关键路径中通道区完成布线所需要的通道数尽可能少。值得注意的是，随着线网分配的变化，位置图中关键路径也可能发生变化，而给线网的分配带来一定的困难。

7.2 总体布线方法

总体布线有时又称作为“概略布线” (Loose routing)，它是以使整个布线设计的布线完成率尽可能高(在门阵列模式中)或整个布线设计完成后芯片的面积尽可能小为目标，把每条线网的各部分合理地分配到各个布线通道区中去，并使各布线通道区中的布线问题得到明确的定义。而每个布线通道区中的最终布线，即对每条线网确定其在该通道区中的具体位置，将通过通道区布线来实现。由于总体布线时仅确定线网各部分的通道区属性，而未确定其在通道区中的具体位置，因此从这种意义上讲，它是一种“概略”

的布线。

然而，由于总体布线是从整个布线设计全局出发进行考虑的，因此，它可望在宏观上，从整体上保证布线的合理性，这也是分级布线——面向区域布线的主要优点之一。

一、多元胞模式中的总体布线

1．线网的分类

在图7.1所示的多元胞设计模式中，单元间的连线可以分为三种类型：

① 线网的接点都在同一个水平通道区的上、下边界上。有时，这类线网中也可包括通过等价接点的交换后，满足上述条件的线网。显然，这类线网一般并不存在布线通道区的分配问题。

② 线网的接点全部在同一个子块列内，但不在同一个水平通道区的边界上。

③ 线网的接点分属不同的子块列。

可以看到，对于第②和第③类线网来讲，存在着相应线网的布线通道区的分配问题。

为了讨论、分析的方便，我们可以把第③类线网定义为具有第②类线网特点的子线网的集合。这些子线网互连的垂直线段将要排在子块列间的垂直通道区中，互连的水平线段按排在子线网相关的水平通道区中。这样第③类线网的各个子线网都可以和第②类线网统一考虑了。

2．冗余通道和冗余通道分配的目标

(1) 冗余通道及其数量的确定

我们首先来讨论第②类线网的分配问题（包括第③类线网属于该子块列的子线网的分配问题）。我们知道，多元胞设计模式的双边单元中，除存在着可起某种转接作用的逻辑等价接点和电学等价接点外，单元中还存在着“冗余通道”(feed-through)或称“转

接通道"(Jumper)、过渡通道等，见图 7.4。此外，在单元之间和单元行的两端，在一定条件下也可允许有连线穿越单元行。这时，也可以想象在这些位置存在着一个"冗余通道单元"(feed—through—cell)或称"行内的冗余通道"。

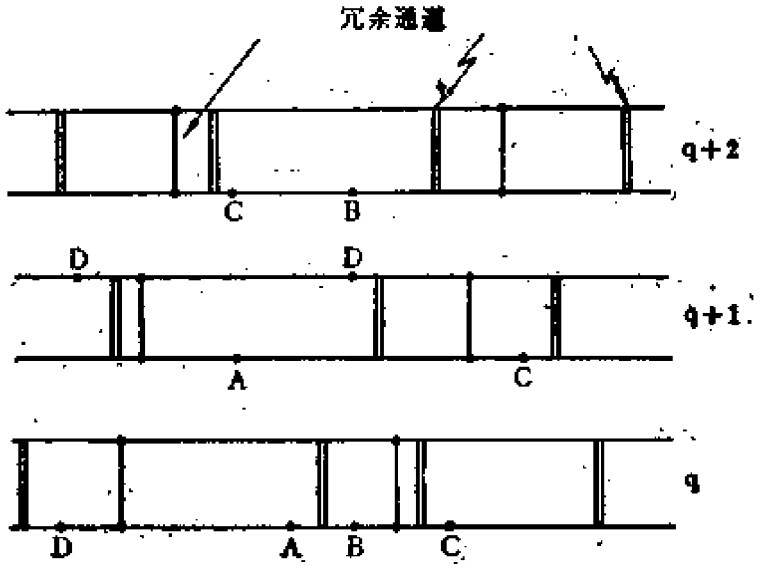


图 7.4　冗余通道示意图

对于每一条必须穿越单元行的第②类线网来讲，似乎有许多穿越位置可供选择。放在我们面前的问题是：从全局看，所有的第②类线网分别应从什么地方穿过单元行才能使布线设计结果最合理？

对于每一个单元行来讲，必须穿越该行的线网的数量是不同的。我们可以以每个单元行的水平中轴线作割线，对所有第②类线网（包括第③类线网的子线网）求最小割切值（可参看第五章中最小割算法）。设 k 列中各单元行的对应最小割值分别为 C_1，C_2，…C_i（值得提一下的是等价接点的交换会对这些值发生影响），同时设各单元行中单元累计长为 BL_i，单元中冗余通道总数为 S_i，则布线设计完成后，k 列子块列中单元行的实际长度下限值为 $ML=\max\{(BL_i+C_i-S_i)\}$。各单元行中冗余通道的总数为 $M_i=ML-$

BL_i+S_i，而“行内冗余通道”总数为$(ML-BL_i)$。

(2) 冗余通道分配的目标

决定列中第②类线网从何处穿越单元行的问题也就是各单元行中冗余通道的分配问题。一种简单的分配原则是：寻求一个所有第②类线网的冗余通道分配方案使所有第②类线网所需的水平连线总长最短[160]。自然，各单元行的冗余通道的分配结果实际上是相互影响的。在实际处理时，考虑到处理效率和实际需要，往往只要求一个局部最优解。

3．冗余通道的分配

当依一定的顺序，比如由下至上，逐个单元行进行冗余通道分配时，每个单元行的冗余通道分配问题可近似地转化为一个线性分配问题。

(1) 冗余通道分配的价格矩阵

对q行来讲，若该行共含P个单元，则可生成一个$m\times n$的价格矩阵$C(m, n)$。其中m为需要穿越该单元行的线网数，n为S_q+P+1，其中$P+1$列分别对应着q行的“冗余通道单元”。矩阵元素$C(i, j)$为线网i利用j冗余通道时，以连线水平方向长度度量的代价，一般采用水平连线长度比所需最短长度的增量值作为代价。为了方便“冗余通道单元”的宽度的调整，可以设想，“冗余通道单元”所对应的$P+1$列冗余通道是一些可以重选、容量可调整的冗余通道。这里，线网所需最短水平连线长度可用复盖线网所有接点的最小矩形的水平边长来度量。当线网布线时利用了j冗余通道时，可设想在j冗余通道处增加了一个线网接点，然后可用上述同样的方法求出线网的水平连线总长。

(2) 冗余通道位置的初步确定

由于在冗余通道的分配完成前，无法精确地确定各“冗余通道单元”的宽度即单元间的间隔和单元行的起始位置。因此在计算代价时将采取一些近似的方法。一种方法是事先定义各“冗余通道单元”的宽度为（“行内冗余通道”总数即$(ML-BL_i)$）/(行内单

元总数+1)。另一种方法是先对各"冗余通道单元"的宽度进行预分配，使各"冗余通道单元"的宽度与第②类线网的分布密度有一定的对应关系。第②类线网密度较高的区域内的"冗余通道单元"的宽度也较大。在计算连线长度时，"冗余通道单元"的通道位置都以其"单元"的 y 中轴线位置作为名义值。

(3) 线性分配算法的修改和实际的分配过程

在实际问题中，不少单元行中冗余通道总数将大于需穿越的线网数，并且"冗余通道单元"对应的列是一些可重迭、容量可调整的冗余通道。因此在利用线性分配算法时，可对一般的算法作适当的修改。下面列出的是一个修改过的改进行扫描法。

STEP1：将对应于单元内的冗余通道的列作上标记。

STEP2：按改进的行扫描法（参见 4.10）将 k 列分配给 q 行。

STEP3：如 k 列是作标记的列，则将该列复盖，否则进行计数。当累计已分配的"行内冗余通道"总数小于该行实际存在的"行内冗余通道"总数时，该列不复盖，否则覆盖全部未作标记的列。重复 STEP2，STEP3 直至所有的线网都分配到一个冗余通道后为止。

冗余通道的分配完成后，由于各"冗余通道单元"的实际容量得到了确定，从而可方便地确定行内每个单元的精确位置。当一个单元行的冗余通道分配完成后，可顺序解决上一个单元行的冗余通道分配问题。在解决当前单元行的冗余通道分配时，由于下面的单元行中，冗余通道的分配都已完成，因此，在计算连线水平长度时只需考虑该单元行相邻通道区中以及上面各通道区中的接点位置就可以了。

4. 冗余通道分配的改进方法

用上述方法解决列内单元行的冗余通道分配问题，实现时简单明确而且一般可以获得较满意的结果[160]。但是由于连线水平线段总长最短化的目标和使布线设计后芯片面积(芯片高度)最小

化的目标并不是完全对应的，因此在一些情况中，尤其是规模很大的问题中将可能对设计结果有较大的影响。

（1）冗余通道分配目标的改进

我们知道，芯片 y 方向的尺度与各水平通道区的高度有着严格的对应关系。设每个单元的高度为 h，j 列中各通道区高度为 h_i，则 j 列的高度 $H_j = qh + \sum_{\forall i} h_i$（其中 q 为 j 列中单元行数），因此芯片 y 方向高度 $W = \mathrm{Max}\{H_j\}$。可以看到，影响 W 值的主要因素是那些具有 $\mathrm{Max}\{H_j\}$ 的列。而在这些列中，影响 H_j 的主要因素是该列中各通道区的高度。

因此，单元行中冗余通道分配一个更好的目标是完成冗余通道的分配使 $\sum_{\forall i} h_i$ 最小化。

图 7.5 说明了冗余通道分配对通道区高度的影响。在图 7.5 (a)中，如果使线网 A 利用冗余通道 t_3 时可使相关的通道区高度和最小化。而如果利用其它冗余通道都将使相关的通道区高度和增加 1。图 7.5(b)所示情况说明了相关通道区高度和和水平连线最短化的不对应性。图中 A 线若以水平连线最短化原则利用 t_2 冗余通道，显然将使下面的通道区高度增加。图 7.5(c) 所示为利用多个冗余通道使相关通道区高度和为最小的例子。

（2）有关的定义和性质

为进一步讨论的方便，我们首先引入一些相关的定义和性质。

定义 1：通道区各列的局部布线密度(Local density)和通道区布线密度(density of channel)。

图 7.6 所示为一典型的水平通道区。可用 4 个数列[T, B, L, R]来表示通道区的互连要求，其中 T, B 为接点的有序集合（允许有 0 接点即空接点存在）：

$$T = (t_1, t_2, t_3, \cdots t_p)$$

$$B = (b_1, b_2, b_3, \cdots b_p)$$

表示了通道区上，下边上接点自左至右的有序排列。L, R 为需由

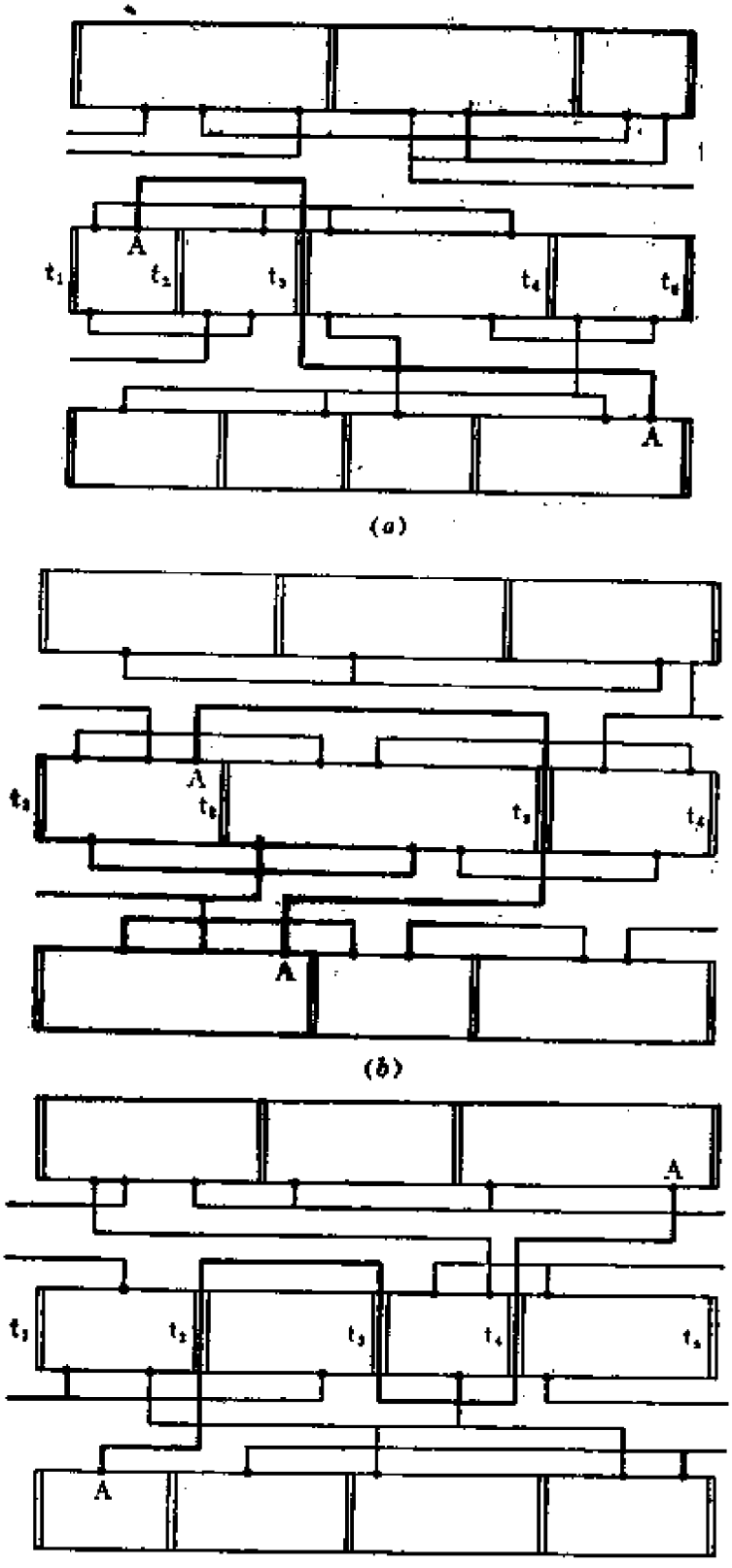


图 7.5　冗余通道分配对通道区高度的影响

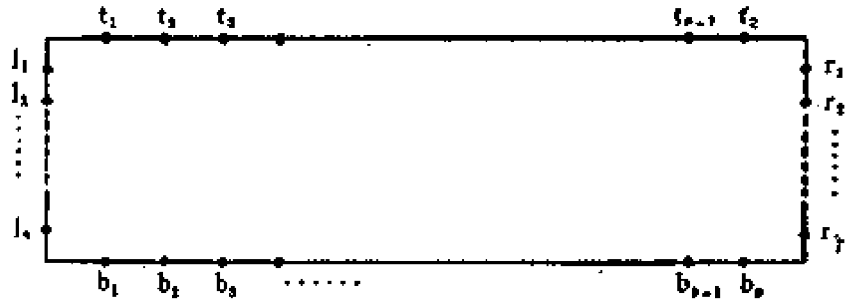


图 7.6 水平通道区

通道区左，右端引出的线网的接点集合：

$$L=\{l_1, l_2, \cdots l_q\}$$

$$R=\{r_1, r_2, \cdots r_s\}$$

L 列和 R 列可看作是通道区的第 0 列和第 $P+1$ 列。

对通道区的任一 i 列，存在着二个接点集合，即 i 列(包括 i 列)的左边接点集合 VL_i 及 i 列(包括 i 列)的右边接点集合 VR_i。分别为：

$$VL_i=\{tk\,|\,tk\neq 0 \text{ and}(k\leqslant i)\}\cup\{b_k\,|\,b_k\neq 0 \text{ and}(k\leqslant i)\}\cup L$$

$$VR_i=\{t_k\,|\,t_k\neq 0 \text{ and}(k\geqslant i)\}\cup\{b_k\,|\,b_k\neq 0 \text{ and}(k\geqslant i)\}\cup R$$

则 $C_i=VL_i\cap VR_i$ 称为与 i 列切割的线网集，其元素数 $VL_i\cap VR_i$ 称为通道区 i 列的布线密度，记作 d_i。各列布线密度的极大值称作通道区布线密度，记作 D。

定义 2：通道区布线高密度列的广度(有时简述为通道区的广度(span))。

通道区内列布线密度等于通道区布线密度的列数，称作通道区布线高密度列的广度，记作 S。

性质 1：对于一个通道区，任何一个可行的布线都至少需要 D 个水平布线通道。

性质 2：当在通道区的上边界(或下边界)任二个接点间插入一个接点时，通道区各列布线密度的值最多变化 1，即 $d_i-1\leqslant d_i'\leqslant d_i+1$。

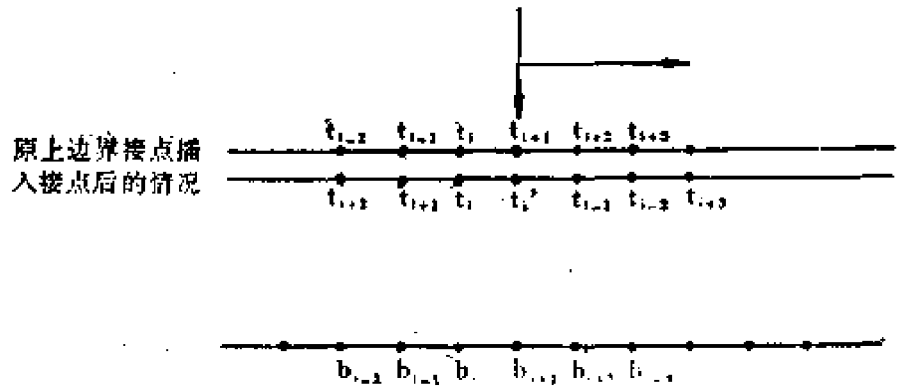


图 7.7　在通道区上边界插入接点时对列布线密度的影响

如图 7.7，在上边界原接点 t_i 和 t_{i+1} 间插入一个接点 t_i'(可以是空接点)，同时 t_k $(k>i)$的所有接点右移一个单位距离。

显然 $d_k'(k\leqslant i)$各列的布线密度将不发生变化，而 $d_k'(k>i)$各列的变化如下述：在自左至右地定义本通道区的线网接点序列时，对于第 k 列来讲，若原 $k-1$ 列上接点为线网终止接点，可定义其特征值 $W_{k-1}=1$，其他性质接点 $W_{k-1}=0$，若原 k 列上接点为线网起始接点，则其特征值 W_k 定义为 -1 其它性质接点 $W_k=0$，则插入后，$d_k'<d_k+W_{k-1}+W_k$ $(k>i)$。

其中，当 $k=i+1$ 时，W_{k-1} 为插入接点 t_i' 的特征值。显然 $d_k-1\leqslant d_k'\leqslant d_k+1$。

插入的其它情况下，各列布线密度的变化可以类推之。

(3) 改进的冗余通道分配方法

从图 7.5 可以看到，如果其它线网的分配都已完成，为了使相关的通道区高度之和尽量小(根据性质 1，这个目标和使相关的通道区布线密度之和尽量小有着极好的对应关系)，分配给线网 A 的冗余通道应使 A 线网布线时，水平线段尽量不经过相关通道区布线密度最高的列。

因此，实现子块列内冗余通道分配的另一种方法是：首先用线性分配的方法对各子块列中的单元行逐行进行冗余通道的预分配；然后对于实际长度小于该子块列中单元行最大实际长度的行依一定方向逐行地在单元间和行的二端插入适当的空接点，使其

总长不大于单元行的最大实际长度，同时使相邻的通道区布线密度之和最小化；最后以每个单元行相邻的二个水平通道区为基础，逐个线网或子线网地调整它们的冗余通道分配结果，使相邻的水平通道区的布线密度之和最小化。当调整时进行的单元移动后，应考虑性质2所述对各列布线密度变化带来的影响。

在调整线网的冗余通道分配时，显然，我们最为关心的是那些具有当前 $\max\{H_i\}$ 的子块列中的冗余通道的分配。

在单元行的冗余通道分配调整中，与列布线密度等于 D 的列相交的二类线网或三类线网的子线网是优先调整的对象。相交数愈多的线网，调整优先度愈高。

5．其它线网的分配

对于接点分属不同子块列的线网的分配，除了其在每个子块列中的子线网冗余通道分配外，还存在着它们的部分垂直线段在垂直通道中的分配问题。这个分配问题实质上也就是决定这些线网在何处穿越水平通道区而实现互连。其分配、调整原则非常类似于水平单元行中冗余通道的分配、调整。不同的是在考虑相邻的垂直通道区的布线密度分布的同时，还必须考虑各子块列中单元行长度不同带来的影响。

二、门阵列模式中的总体布线[114]、[113]

1．门阵列总体布线的目标

与多元胞模式不同，门阵列模式中各单元的位置是固定不可移动的，因此相应的布线区域也是固定的。即对于每一个总体布线单元其布线可利用的资源也是固定的。这些资源可描述为：

一定数量的水平通道 HN_i（在一层上）；

一定的水平通道长度 HL_i；

一定数量的垂直通道 VN_i（在另一层上）；

一定的垂直通道长度 VL_i；

所允许存在的通孔数 M_i [115]；

一定数量的引出(入)端 IOS_i。

当一条线网进行布线经过每个总体布线单元时，必然要使用它们的一些资源。为了使布线完成率达到100%，其必要条件是在实现布线时，每个总体布线单元所需要的资源不超过其固有的资源。若定义每一总体布线单元的溢出量 F_i 等于所需资源量 N_i 和固有资源 S_i 之差。则门阵列模式总体布线的目标可叙述为，寻求一个合理的布线分配使各总体布线单元的总溢出量 $\sum_{\forall i} F_i$ 最小化，或者是使各总体布线单元中最大的溢出量 $\mathrm{Max}\{F_i\}$ 最小化。

2．门阵列总体布线的主要步骤

一般门阵列的总体布线可分为三个主要步骤：初布、拆线、调整重布。后二个步骤实际进行时往往需反复多次才能获得较满意的结果。

(1) 初始布线

在初始布线阶段，可把每个总体布线单元看作是一个大网格。忽略线网接点及走线在网格单元中的具体位置，然后以连线长度最短化为目标，在大网格场中进行布线。此时，为了避免一次布一条线所造成的问题和由于任意选择布线顺序而造成的对后面布线的影响，在初布时将忽略其它限制条件(设想各网格单元资源足够多）而把线网全部以最短连接路径直接布上。对每一条线网的布线可使用求最小斯坦尼树的准最优解的方法[116]来实现布线。很明显，正如可预料的，初布的结果将使有些网格单元变得相当拥挤，即对资源的需要量将大大超过固有的资源量。而另一些网格单元中布线则很稀疏，因此必须把一部分通过溢出量大于零的网格单元的线网拆去重布。

为了拆线、调整重布的需要，在初始布线完成后应对各网格单元各项资源的利用情况（可用溢出量作为标志量）进行定量的统计。

(2) 拆线

在一个复杂的布线问题中，决定拆去已有布线中的哪些线网，

使这些线网重布后能满足总体布线的目标是一个相当复杂和困难的问题。虽然已经提出了不少启发性算法，但要严格地评价它们的效果也是一个困难的问题。

① 一种决定所拆线网的方法是：对每条线网定义一个权重 W_j（有时也可分别对各子线网定义权重，并以子线网为单位进行拆线重布）。W_j 可等于 j 线网所通过的网格单元的溢出量 F_i（$F_i>0$）之和，即 $W_j=\Sigma F_i$ $F_i>0$ 且单元 i 在 j 线网的联结路径上。W_j 的另一种定义是令其值等于 j 线网所通过的网格单元的溢出量的最大值，即 $W_j=\mathrm{Max}\{F_i\}$ 通道区 i 在 j 线网的联结路径上。在拆线重布时，权重 W_j 大的线网先选来拆去，并进行重布。

② 决定所拆线网的另一种方法是由线网接点的相互位置来作为选择的出发点。在图 7.8 中，设网格单元(3,3)的水平通道溢出量为 1，如何选择被拆线网呢？为了尽量使重布时所需的总资源数增加得最少，以有利于今后其它线网的拆线重布。考虑到各类线网在所需总资源数最少的情况下（即连线总长最短的情况下）实现布线的可能路径数，我们可以对不同接点位置的线网赋予不同的拆线重布权重。

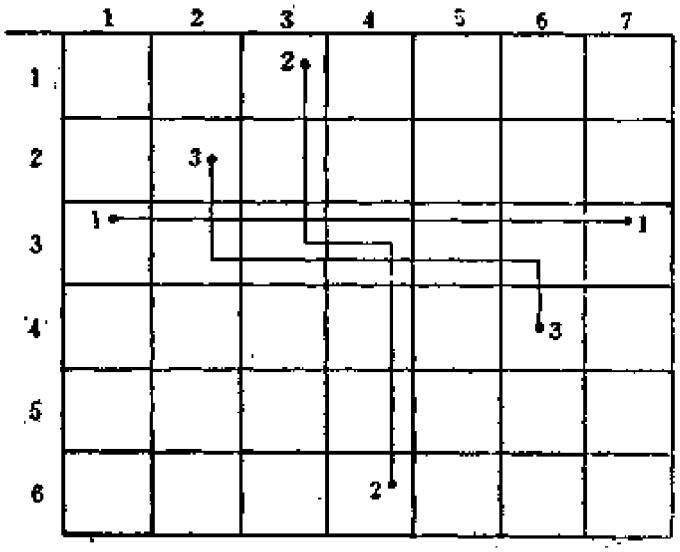


图 7.8 拆线网选择

当线网的两个接点处于同一水平网格单元行中或同一垂直网格单元列中，其拆线重布权重较小(拆线优先度较低)。

当线网的两个接点不在同一水平(垂直)网格单元行（列）中时，具有较高的拆线重布权重。一般讲，线网在大网格中以最短曼哈坦距离实现布线的可能路径数愈多，则其拆线重布优先度愈高。当着眼于解决所越过的网格中垂直(水平)通道溢出的问题时，线网接点间的水平(垂直)方向的长度愈长，则其拆线重布优先度愈高。

在图7.8中，线网2的拆线重布优先度高于线网3，而线网3的拆线重布优先度将高于线网1。

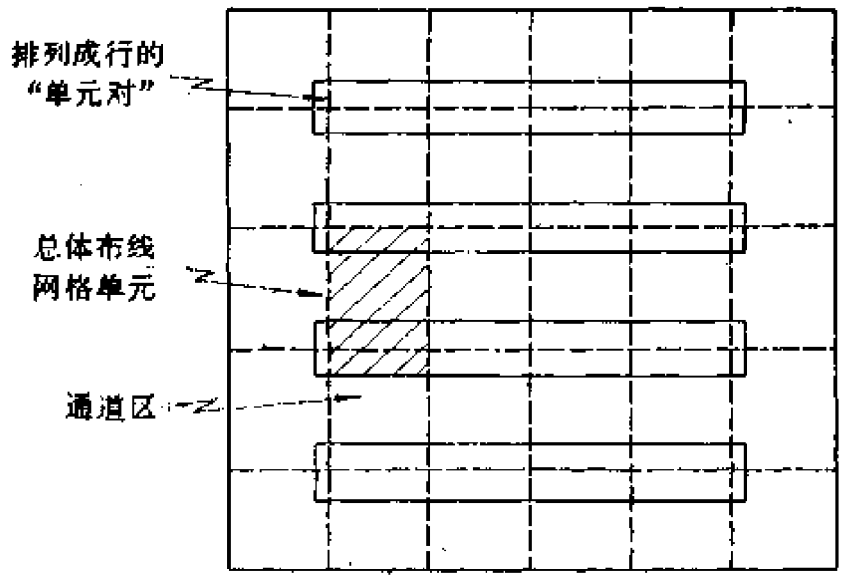


图7.9 硅栅门阵列总体布线

③ 文献[117]介绍了一种用于高密度硅栅CMOS门阵列的总体布线方法。门阵列上的基本单元由“背对背”(back to back)的“单元对”构成，并如图7.9所示进行划分、定义总体布线所用的大网格。对于网格单元的左、右边界来讲，其固有的资源量和需要量可用相应的水平通道数来描述。而对上、下边界来讲，其固有量和需要量则可用相应的垂直通道数来描述。因此，在初布后可依网格单元的边界分别计算出各边界上的溢出量。为了减少以至消

除各边界的溢出情况，必须拆去并重布一些越过溢出量大于零的边界的线网。需要注意的是，拆去一些线网时不要“拆过”了头，即使有些原溢出的边界反而变成了一些利用得很差的边界。要不然的话，只要拆去所有与溢出边界有关的线网就可以了。

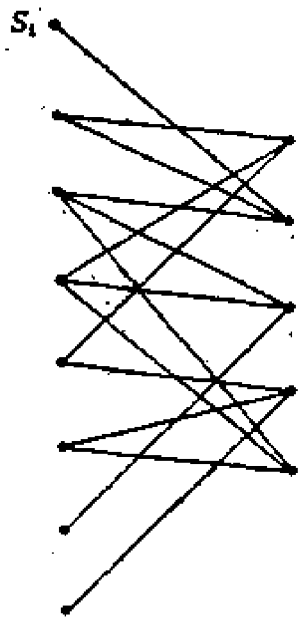


图 7.10 二分图

我们可以用一个二分图来描述上述拆线选择问题。在图 7.10 所示的二分图 G 中，左边顶点 S_i 对应着越过溢出边界的线网，右边顶点 T_j 对应着溢出量大于零的边界，当且仅当线网 S_i 越过 T_j 边界时，在它们对应的顶点间存在一条边。图中每个 T_j 顶点都具有一个相应的权重 W_j，W_j 即为 T_j 边界的溢出量。那末上述拆线选择的目标可叙述为：求一个线网的顶点子集，以获得一个对所有 T_j 顶点的复盖，并使每个溢出边界被复盖 k_j 次，$k_j = W_j$。显然这个复盖应是一个最小的基本复盖。

当每一个溢出边界顶点的权重都为 1 时，这个问题即是一个求最小集的复盖问题。并已由 cook 和 karp[19][20]证明是一个 NP——完备问题。

一般情况下，即 $W_j \geqslant 1$ 的情况下，上述问题也可描述为一个整数规划问题。

设二分图左边顶点数为 m(越过溢出边界的线网数)，二分图右边顶点(溢出边界)数为 n，则可定义一个关联矩阵 $A(n \times m)$。

其中，$A(k,l) = 1$ 如果 S_l 对应的线网越过 T_k 对应的溢出边界时；

$A(k, l) = 0$ 不满足上述条件时，

则上述选择问题就可转变为下述问题：

$$\min\ (x_1 + x_2 + x_3 \cdots + x_m)$$

目标为：$A \cdot X^T \geqslant W^T$

其中：$W = (W_1, W_2, \cdots W_n)$

$X = (X_1, X_2, \cdots X_m)$

W_k：为 k 边界对应的溢出权重

X_i：为 0 或1

这个问题也是一个 NP——完备问题 。但在实际处理时，并没有必要去寻求真正的最小解。只需找一个满足一定要求的解就可以了。

在采用迭代方式重布时，关心的常常是方法的简捷和有效，因此可以把拆线选择问题分成二个选择过程，首先选择一个溢出边界的子集，着重注意使拆线后，该子集内的边界溢出量能得到减少。

在确定这个子集后，再选择一个线网的子集来进行重布。

a）溢出边界子集的选择　一种方法是每次选择一条具有当前最大溢出量的边界。这将有利于使具有溢出量最大值的边界的溢出量优先地得到减少。

另一种方法是以与度数最大的线网顶点相关的溢出边界作为所选择的边界子集。这时，将这条线网重布将可使尽可能多的溢出边界上的溢出量都得到减少。

第三种方法是把所有具有当前最大溢出量的边界作为所选择的边界子集。这种方法将试图在一些线网的重布后，使整个网格边界集的最大溢出量得到减少。

b）线网的选择　选择与 *a*）中所选边界相关的线网进行重布以减少或消除这些边界的溢出量。

一种简单的方法是选择那些通过尽可能多的选定边界的线网进行重布。

在图 7.11 中画出了选定的溢出边界（作标记的边界）。这时可令芯片的外周边界也都是溢出边界以便进行统一的处理。如果存在着一个由溢出边界形成的回路（loop），那么仅越过回路边界

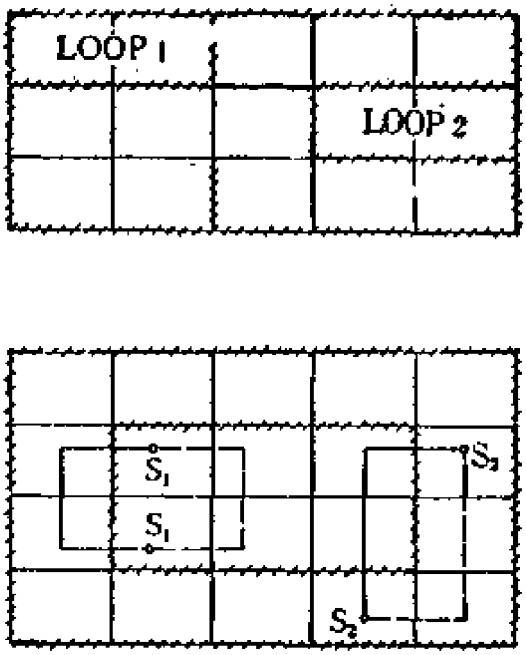


图 7.11　选定溢出边界

一次的那些线网不可能在重布后使该回路上的总溢出量有所减少。当线网重布时，如果存在着全部路径都在回路的内部或外部的通路，则一定能使回路上总溢出量有所下降。

因此，为减少回路上总溢出量，应该选择满足下述条件的线网：

i．该线网与回路的交叉数大于 1。

ii．该线网存在着不与回路交叉的重布路径。

(3) 重布

在决定了拆线、重布的线网后，接下来的问题是如何将拆去的线网在总体布线的大网格上重新实现布线才能减少或消除溢出。

这里存在着一些具体实现的方法。

① 一种简单的方法是逐条线网利用修改过的李氏算法重新布线。重布前，由各总体布线单元的溢出量的大小确定各布线大网格的动态价格。溢出量值愈大，网格的布线动态价格愈高。运用修改过的李氏算法进行重布，即寻找一条实现互连且路径上各网格动态价格和最小的路径。每条线网重布后，应修改有关的网格的动态价格，如果保证每条线网重布后，其路径上网格的动态

布线价格之和比重布前是下降的，则算法的收敛性是有保证的。这种方法的缺点是重布的结果与重布线网的布线顺序有密切的关系。

② 另一种方法当选定重布的线网集后，该线网集合中的每条线网的重布都是独立的，即在每条线网重布后不进行各网格布线价格的修改。这种重布方法与重布的顺序无关，但不能保证算法是收敛的。为了保证算法是收敛的，当选定的重布线网集全部重布后，需检测其总的溢出量是否比重布前下降了，只有当总的溢出量是减少的，才承认此次重布的结果。如果收敛的速度很低时，采用这种方法实行重布将花费相当多的机时。

由于门阵列的母片在设计、选用时已较充分地考虑到设计成功率的要求，往往留有相当的余量以满足布线的要求，因此，在一般情况下，经过若干次的拆线重布，原则上可使各通道区(或各边界)的溢出量小于、等于零。

在总体布线中，由于在每个总体布线单元中忽略了线网接点的基本位置，因此，即使各总体布线单元概略统计的溢出量都小于、等于零，也并不能保证在最终具体实现各线网的布线时能获得100%的布线完成率。

文献[117]考虑到上述情况在计算溢出量时引进了余量(slack)的概念。即定义溢出量为：

$$\text{溢出量} = \text{需要量} + \text{余量} - \text{固有量}$$

在总体布线开始时，令余量等于零，当经过第一次总体布线的初布——拆线——重布，使各总体布线单元溢出量小于、等于零后，分别令余量为1，2等，重新计算各单元的溢出量，重新进行总体布线的拆线重布。如果能够获得满意的结果，则将能更好地满足最终实现布线的要求。当然，对于设计要求比较严格的电路，如果要求余量大于零，那么总体布线将会存在着更大的困难，耗费更多的机时。

总体布线完成后，通过每个网格单元上、下边界(水平边界)的

线网已被确定，但每个线网通过边界上哪条通道(垂直通道)还未得到确定。此时，可应用与多元胞模式冗余通道分配类似的方法(见上一节)对垂直通道进行分配。还可以通过迭代来改进初始的分配，使通道区的最大密度最小化。

当垂直通道的分配完成后，相邻的单元行之间就成为一个明确定义的水平通道区，而可方便地应用通道区布线算法实现最终的实际布线(见7.3)。

IBM公司曾对100个双极逻辑芯片(门阵列模式)应用总体布线实现布线设计的结果作过报道[114]，这些芯片平均每片含125门(Gates)和136个线网，图7.12所示为统计的结果。以线网为对象，平均的布线完成率为98%，若以连线长度为基础，则完成的连线占总连线长度的99.9%。其中35%全部自动地实现布线，余下的65%需人工进行少量的干预。近年来随着系统和算法的发展，平均的布线完成率约为99.8%左右，50%的几百门的芯片可自动地完成全部连线。平均每个芯片仅有1.4条线网无法自动完成布线。

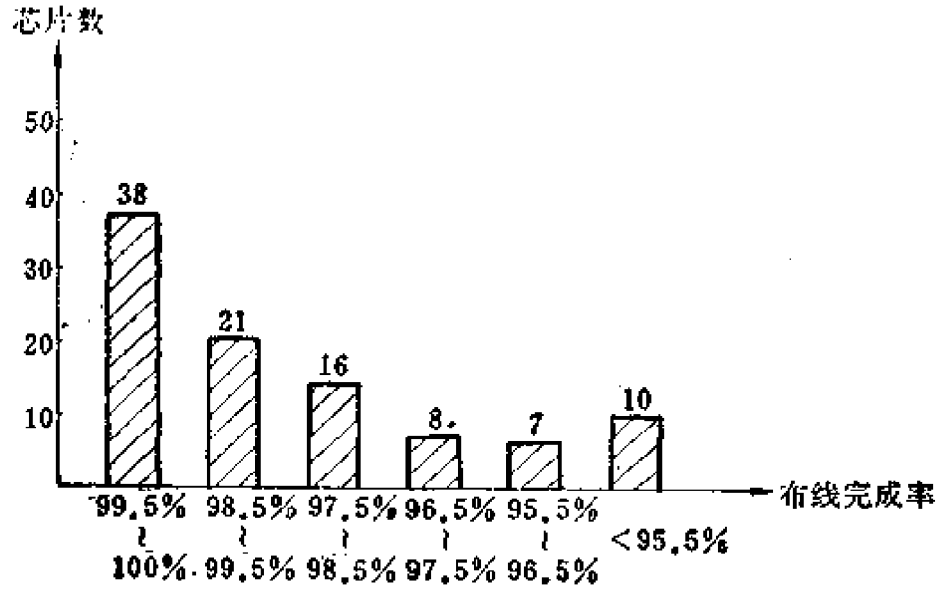


图7.12 门阵列模式总体布线设计结果统计

同时，我们也可以看到，由于门阵列设计模式的特点以及算法方面存在的问题，为了完成门阵列的布线设计，人工干预(交互式的布线设计)仍是必须的，尤其是对于规模比较大的电路或设计要求比较严的问题更是如此。

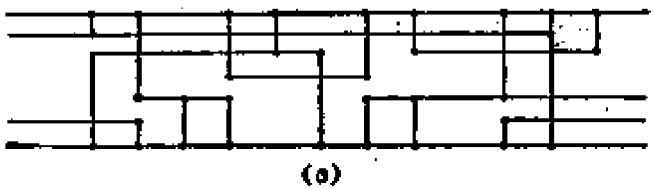
正因为如此，除进一步探索更有效的布线方法（包括总体布线)外，剩线的自动处理也是人们关心的一个研究课题[109]。

7.3 通道区布线算法

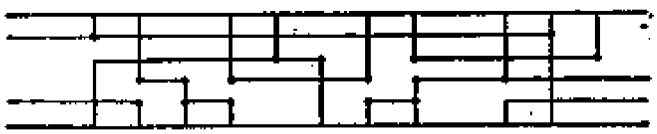
通过总体布线，整个芯片的布线问题被分解为若干相邻单元或相邻单元行(列)之间的子区域的布线问题的集合。只要按一定的顺序 [216] 分别实现各子区域内的实际布线也就可以最后地完成整个布线设计任务了，各子区域的布线问题通常称为通道区(channel)布线问题。

在实际问题中，通道区布线问题存在着多种不同的形式。

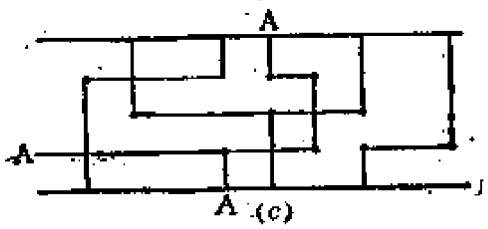
1．一般平行边界的通道区布线问题[129]、[130]



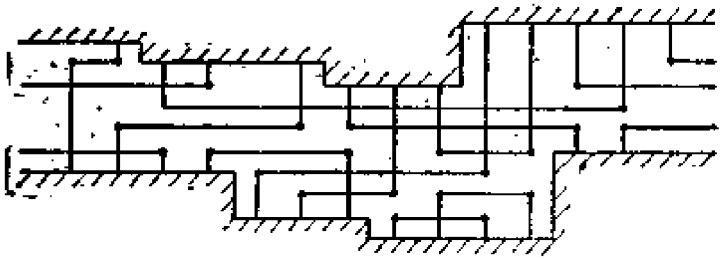
(a)



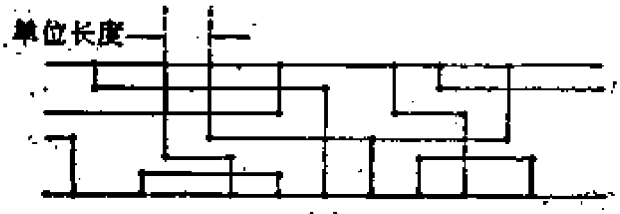
(b)



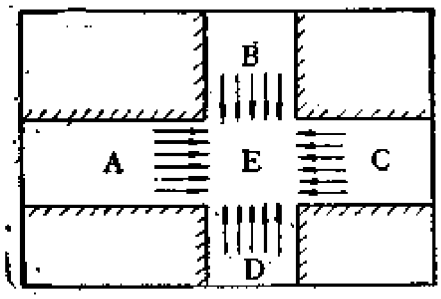
(c)



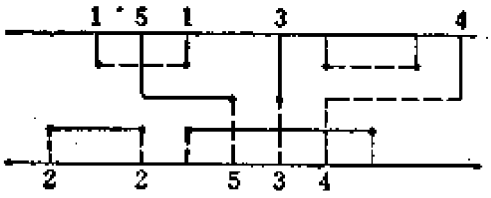
(d)



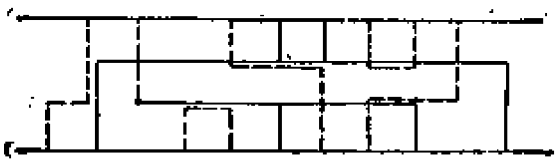
(e)



(f)



(g)



(h)

图 7.13　平行边界通道区布线示意图

如图 7.13(*a*) 所示，接点排列在矩形通道区的对边的边界上，其左、右边界的引出线位置可由算法确定，通道区内不存在任何布线障碍。布线时所有水平线段布在一层上，所有垂直线段布在另一层上，二层间必须的连接用通孔实现。当接点之间的间距等于单位长度(一般为连线实际宽度和连线间最小间距之和)的整倍数且上、下接点按列排列时，即为一般平行边界通道区的布线问题。在布线方式上，若规定每条线网只允许有一条水平线段时，又称为非曲干布线或直干布线(without dogleg)。当允许每条线网的相邻接点之间的水平线段可不处在一条直线上时，又可称为曲干布线(dogleg)，如图 7.13(*b*)所示。当允许线网的水平线段实际长度大于其直干布线时所需的最小长度时（如图 7.13(*c*) 中线网 *A*)，可称为迂回布线方式[131]。

2．不规则边界的通道区布线问题[132]、[133]

如图7.13(*d*)所示，通道区的上、下边界成不规则的折线时(折线每个拐角为 90° 或 270°)，称为不规则边界的通道区。在实际设计中，常常存在一条边界为不规则的折线的情况，如多元胞设计模式中的单元列与左、右边界形成的通道区，我们称之为单边界不规则通道区。

3．可变接点列的通道区布线问题[135]

如图 7.13(*e*)，在实际设计中，有时通道区上、下边界上接点间距并不是单位长度的整数倍，同时上、下边界的接点不按列排列，这种情况下的通道区布线问题我们称作可变接点列的通道区布线问题。

4．多层布线的通道区布线问题[137]

在这种通道区布线问题中，可以有多于二层的布线层用来实现布线。如三层布线情况。此时，可以用二层来安放那些水平线段(如第一层，第三层)，而用另一层安放那些线网的垂直线段(第二层)即所谓 *H*—*V*—*H* 方式。自然，另一种方式可以是 *V*—*H*—*V* 方式。原则上也可以存在更复杂的分层方式。

5. 三边或四边通道区 (switching box) 的布线问题 [123]、[139]、[140]

如图 7.13(*f*)所示，如果设想 *A*、*B*、*C*、*D* 四个通道区中已有三个通道区已完成了布线。则对于通道区 *E* 来讲，它的三条边界上的接点位置是已确定了的，只有一条边界上的引出线位置可由算法确定，这种通道区布线问题称作三边通道区布线问题。一般来讲，对这种通道区实现最优化的布线时，难度将高于一般通道区的布线问题。当在图 7.13(*f*)中，A、*B*、*C*、*D* 四个通道区都完成布线后，再进行 *E* 通道区的布线时，则 *E* 通道区四条边界上的接点位置都已固定，这时，*E* 通道区的布线问题可称作为四边通道区的布线问题。这是一个非常困难的布线问题，同时也是一个引人入胜的问题，至今尚没有令人十分满意的解决方法，在实际设计时，一般通过 7.2 节中所阐述的垂直通道分配或通过适当地选择芯片上各通道区的布线顺序[124]，使布线时不产生三边或四边通道区的布线问题。

6. 非横竖分层的通道区布线问题[142]

如图7.13(*g*)、(*h*)所示，通道区内线网的布线分层已不再是一般通道区中那种横(水平线段)、竖(垂直线段)分层的方式了。图 7.13(*g*)中，依线网接点位置规定了如图的五种连线、分层的方式，并以此为基础发展了相应的布线方法。在图 7.13(*h*) 中采用了更自由的分层方式，在电路性能要求允许的条件下，有时这种方式的通道区布线可获得更好的布线结果，将可使实现布线所需的布线区面积更小。

通道区布线的种类繁多，在近十几年中提出了各种通常区布线算法。为了讨论的方便，本书将以一般的平行边界通道区布线问题为基础，来讨论各种布线方法。

一、通道区布线的算法目标和算法模式

1. 通道区布线的算法目标

在实际的布图设计中，求解一般的平行边界的通道区布线问题的目标可以有二种主要的情况。下面以水平通道区为例来说明这二种情况。

(1) 通道区高度可调整的情况

在通道区高度可调整的情况中(如多元胞设计模式，任意元胞设计模式中)，通道区布线的目标是实现通道区内所有线网的布线并使所需的布线区面积最小化。由于通道区的长度是一个常数，因此布线区面积的最小化也可等价地描述为使通道区的高度或通道区内水平通道数的最小化。

(2) 通道区高度固定的情况

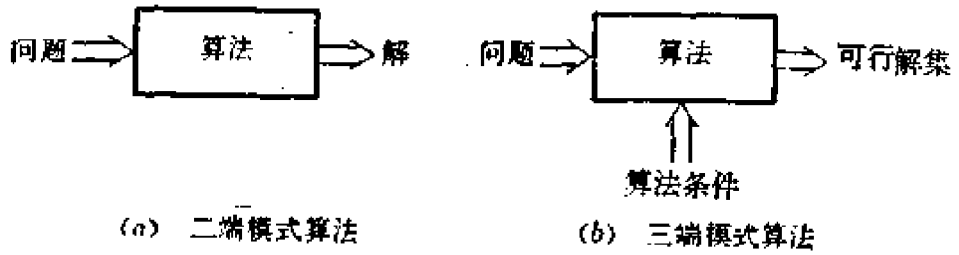
在通道区高度固定的情况中(如门阵列设计模式中)通道区布线的目标则可描述为尽可能在每个固定大小的通道区内100%地完成所有线网的布线。可以看到，这个目标和完成布线时使所需的水平通道数最少是等价的，因此，在通道区布线算法中，其算法的主要目标可统一地描述为：实现通道区内所有线网必需的连接，并使所需的水平通道数最少化。

在实际应用时，通道区布线算法必须考虑电路性能及工艺规则的要求。比如图7.13(*h*)所示的布线方式中，布线结果可能使连线的寄生电容值变得较大而影响电路的性能，因此，有些高速电路就不能采用这种方式布线，同时由于芯片上通孔的数量对芯片的成品率有很大的影响、互连线的长度对一些高速电路的性能或电路中一些高频信号的传输有较大的影响，所以在完成布线时，使所需的通孔数尽可能少，连线长度（尤其是一些关键线网的连线长度)尽可能短也是布线设计的目标之一。在门阵列模式中，由于在相当数量的通道区中，通道区所固有的水平通道数将大于完成布线所必需的水平通道数，此时尽可能减少所需的通孔数和使连线长度尽可能短将成为更重要的目标。

2. 通道区布线算法模式

已经证明，求解一般的平行边界的通道区布线问题，使所需的

水平通道数最少化是一个 NP— 完 备 问 题 [144]。因 此 在 算 法中，求解的精确性和求解的计算效率之间存在着尖锐的矛盾。在实用系统中，一方面要求算法有足够高的求解精度且对不同的布线问题有相当高的求解稳定性(精度稳定性)，另一方面要求算法有足够高的处理效率且对不同的布线问题具有很好的效率稳定性。自 1971 年以来，不少软件研究人员为此作出了很大的努力，提出了各种算法。这些算法基本上可分为二种主要模式(如图7.14(a)、(b)所示)。



(a) 二端模式算法　　(b) 三端模式算法

图 7.14 算法模式

(1) 二端模式算法

我们可以把算法想象为一个黑盒子，如果从黑盒的一端装入需求解的问题，则以黑盒的另一端经过一段时间后就可取得问题的解。这类算法，我们称之为二端模式算法。

在实际算法中，二端模式算法的求解精度和稳定性都相当好，需解决的主要问题是处理效率及其稳定性问题。典型的算法如 1973 年"optimnm"算法[129]，在直干布线时，它具有很好的求解精度和稳定性，对不少问题也具有相当高的处理效率，但对有些问题，则计算时间相当长，处理 BELL 实验室提出的"difficult example"布线问题，该算法处理 4 小时，计算过程仍未结束。

(2) 三端模式算法

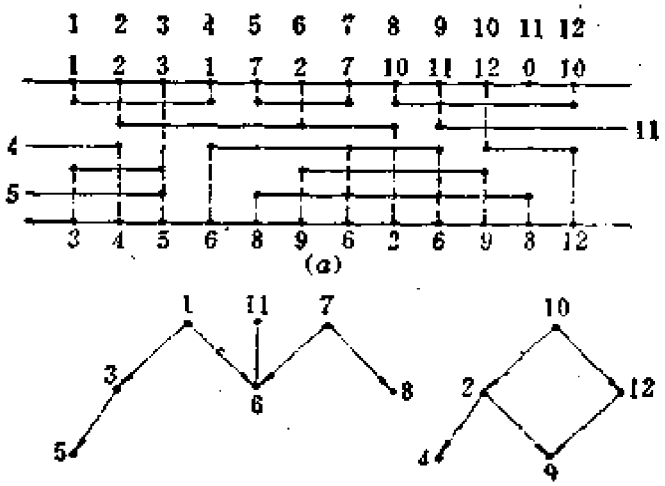
这类模式算法则要求从黑盒的一端装入所求解问题后，再装入(也可以由软件自动产生)一些不同的算法条件，经过一段时间后可从黑盒的另一端得到一个可行解的集合。一般对应于每种算法条件或算法条件的组合都将产生一个可行解，然后可以从该可

行解集中按算法目标选择一个最好的解作为实际的算法结果。

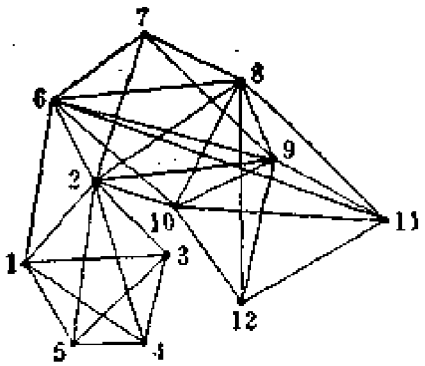
在实用算法中有不少算法属于三端模式算法。这类算法一般使用一些启发式的原则来解决通道区的布线问题，因此通常具有相当高的处理效率和效率稳定性，但在不同的算法条件下将得到不同的解。因此为了获得足够满意的解，往往需要通过改变算法条件，求出一个相当大的可行解集。通常人们关心的问题常常是：一个算法通过改变算法条件能不能找到足够满意的解，以及为找到足够满意的解，所要求产生的可行解集的大小。显然，这里可行解集的大小将直接影响算法的实际处理效率。典型的算法如 D. N. Dentsch 的曲干(dogleg)算法[130]和 T.Yoshimura 和葛守仁提出的匹配算法(matching)[150]。匹配算法通过选择不同的起始列(starting column)和匹配方向对 "difficalt example"布线问题可取得比"dogleg"算法更好一些的结果。

二、通道区布线的基本限制关系及其描述方法

图 7.15(a) 所示为一个典型的通道区布线问题。接点的标号为接点所在的线网号，相同标号的接点需在通道区内实现互连，



(b) 垂直限制图



(c) 水平区间图

图 7.15 通道区布线基本限制关系

"0"标号接点表示该点为一个空接点。线网接点实现互连时，一个基本的限制条件是不同线网在同一层的线段间不允许发生交叉或重迭。当把所有的水平线段放在一层上，而把所有的垂直线段放在另一层上实现布线时，上述限制条件可分别描述为垂直限制条件和水平限制条件。

1．垂直限制条件及其描述方法

(1) 垂直限制条件和垂直限制有向图

在任何一个可行的布线解中，不同线网的垂直线段不允许发生重迭。为了满足上述限制条件，当通道区第 k 列上 $t_k \neq 0$、$b_k \neq 0$ 且 $t_k \neq b_k$ 时，则 t_k 线网的水平线段在布线中，其位置必须高于 b_k 线网的水平线段。在图7.15第 3 列上，即要求线网 3 的水平线段必须布在线网 5 的水平线段的上面。否则线网 3 的垂直线段在第 3 列上至少有一段将和线网 5 的垂直线段发生重迭。这种垂直限制条件可方便地用一个有向图来描述，如图 7.15(b) 所示。图 $G(V, E)$中顶点 V_i 是 i 线网的映射，顶点 V_i 和 V_j 之间的有向边 E_{ij} (由 i 指向 j) 表示线网 i 的水平线段必须高于线网 j 的水平线段。图 7.15(b) 所示的就是通道区布线问题(图 7.15(a))的垂直限制有向图 G。

定义：最大限制链长度 L_{max}。在不存在回路的垂直限制有向

图中，各入度为零的顶点到各出度为零的顶点间的最长的有向路径长度加1，称作 G_r 的最大限制链长 L_{max}。显然，通道区布线时，L_{max} 是完成布线所需水平通道数的一个下限界值。即：任何可行的布线所需的水平通道数必须大于、等于 L_{max}。

(2) 垂直限制有向图中的回路及一般处理方法

有时，在垂直限制图中可能存在着回路(如图7.16)，在直干布线和典型的曲干布线(只允许在接点所在列作曲干处理)中，回路中至少有一条线网将无法实现可行的布线。为了解决这个问题，除在布局时予以适当考虑，尽量避免在垂直限制图中产生回路外，一般可采取下述方法去完成布线：

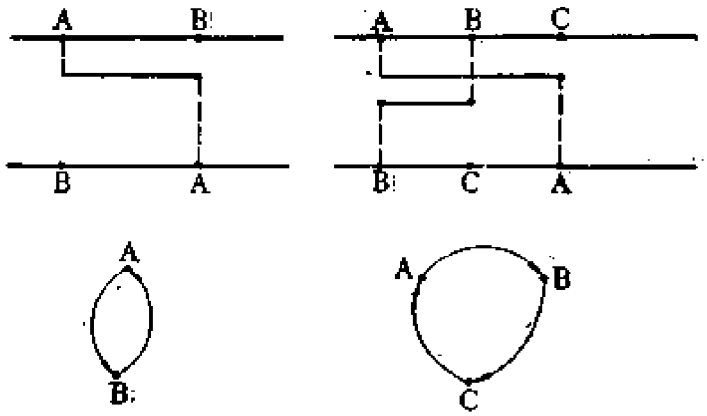


图 7.16　垂直限制图

① 利用可能存在的等价接点，修改线网的连接表来消除垂直限制图中的回路，如图 7.17(a) 所示，图中 A_2 和 A_2' 为等价接点。

② 在线网布线的局部采用其它的分层原则实现布线，图7.17(b)是其中的一个例子。

③ 采用广义的曲干或迂回布线。这里所谓广义的曲干布线是指线网水平线段的"弯曲"的位置不受该线网接点所在列的限制，而在需要并可能的位置都允许水平线段发生"弯曲"，如图7.17(c)所示。迂回布线方式见图 7.17(d)。

当采用后二种方法时，不少情况中将使完成布线所需的水平通道数增加，因此在复杂的通道区布线问题中，当垂直限制图存在

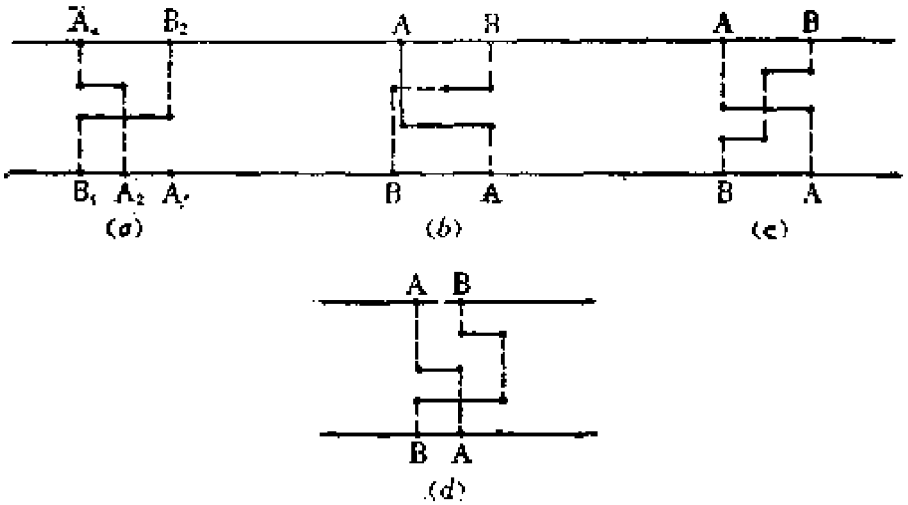


图 7.17　消除垂直限制图中的回路

着多个回路时，存在着一个处理过程如何优化的问题。

(3) 消除回路时的优化处理

优化问题首先可描述为：当垂直限制有向图 G 存在着回路时，如何求一个顶点的最小子集，使删除该子集及相关的边集后，图 G 中不再存在回路。其实际意义是如何处理最少数量的线网而使图 G 中不再存在回路，不幸的是，这个问题也是一个 NP——完备问题。作者在引文[151]中提出了一种求准优解的方法，这种方法分为二个主要步骤：

① 求优化的初始有序子集

a) 求优化的初始有序子集的方法

STEP1：从图 G 中删去所有非回路的相关顶点及非回路的边，得到 G 的子图 G'；记 G' 中各顶点的度数为 IDi，令 $k=1$。

STEP2：G' 图中是否存在回路？若不存在(G'图为一个空图)，处理过程结束。若存在(G' 图非空图)，令当前 G' 图中各顶点度数为 CDi（初始时，$CDi=IDi$)，根据选优函数 $F_i=CDi\times PW+IDi$，其中 $PW\gg IDi$。选择当前具有最大 F_i 值的一个顶点作为结果顶点集中第 k 个元素；

$$k=k+1。$$

STEP3：从G'图中删去该顶点及其关连边，并删去由此产生的所有非回路的顶点和边，形成新的G'图。转STEP2。

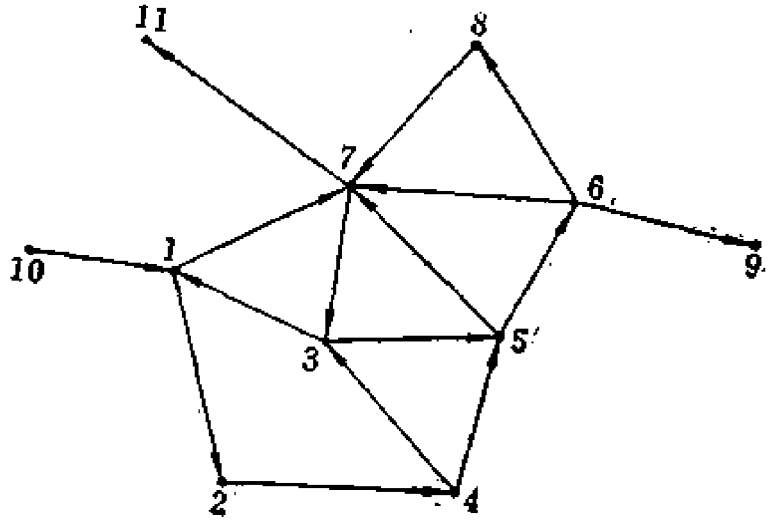


图7.18 消除回路时优化处理例

如图7.18所示的有向图，用上述方法求得的初始有序子集为(7.3)。

b）逆向删冗法的过程

求得初始有序顶点子集后，恢复初始的G'图。首先逆向地删去该子集的最后一个顶点V_P及其关联边，判定V_{P-1}点是否是G'图中回路的相关顶点，若是，则继续从图G'中删去V_{P-1}及其关联边；若不是，则从初始有序顶点子集中删去V_{P-1}点；重复上述过程，直至顶点子集的全部元素都被处理后为止。经过删冗的初始顶点子集就是求解的结果。在实际问题中，相当多的问题可求得最优解。图7.18例中，经删冗后最后解为(3)。

② 用逆向删冗进行解的进一步优化后采用广义的曲干布线或迂回布线来实现选定线网的布线时，它们的水平线段"弯曲"的位置如何选择的问题，这也是一个相当困难的问题。在实际应用时，为了避免产生附加的更复杂的垂直限制关系，一般选择上、下接点都是空接点的列进行线网的曲干处理或迂回处理。当存在着多个候选位置时，可考虑下述分级的优化目标：

a）尽可能使通道区布线密度D不增大

b）尽可能使增加的水平线段复盖通道区列布线密度较小的列；

c）尽可能使增加的水平线段长度最短化。

图 7.19 所示为一个垂直限制图中含有多个回路的通道区，大部分线网采用直干布线的结果。

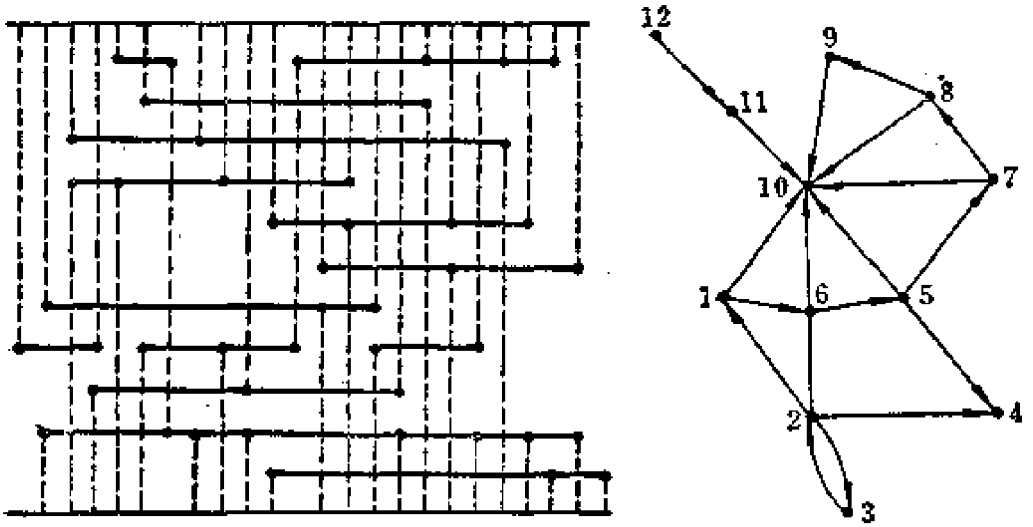


图 7.19 一个垂直限制图中含有多个回路的通道区

当采用曲干布线时，对于多接点线网可分别定义为一些由相邻接点组成的子线网集。一个具有 m 个接点的线网可定义为 $m-1$ 条由每二个相邻接点组成的子线网。这时在相应的垂直限制图中，顶点将与每一条子线网相对应。需要注意的是属于同一线网的子线网间不存在垂直限制关系。

图 7.20 中通道区布线问题若采用直干布线 L_{max} 等于 3，图中给出了曲干布线时的垂直限制图，可以看到，相应的 L_{max} 等于 2。

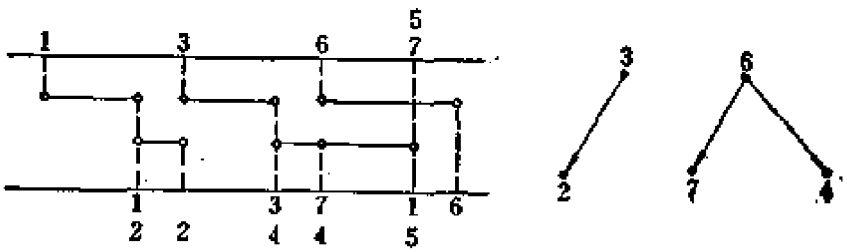


图 7.20 曲干布线时的垂直限制图

在含有较多数量的多点线网的通道区布线问题中，采用曲干布线通常可使通道区的最大限制链长有所下降，并且可望减少限制图中产生回路的可能性。

2. 水平限制条件及其描述方法

(1) 水平限制条件和水平区间图

通道区布线的水平限制条件可描述为：在任何一个可行的布线解中，不同线段的水平线段不允许发生重迭。为了满足上述限制条件，当线网的水平线段由线网最左端的接点和最右端的接点来确定时，设 $S(i)$ 为与通道区列 i 相交的线网子集，则 $S(i)$ 中任何二个元素的水平线段都不允许布在同一条水平通道上。上述限制条件可方便地用水平区间无向图 G_I 来描述[150]，图 G_I 中的各顶点对应着一条线网，当二条线网都必须和同一个通道区列发生交叉时，在它们对应的顶点间连接一条边。图7.15(c)就是图7.15(a)所示通道区布线问题的水平区间图 G_I。

无向图的一个子图如果是一个完全图，即子图中任二个顶点之间都存在着一条连接它们的边，则该子图称作为是一个团图(cliques)。如果一个团图中增加任一个顶点都将破坏其团图的性质，则该团图称作为最大团图。图7.15(c)的最大团图为顶点1、2、3、4、5；1、2、6；2、6、7、8、9；2、6、8、9、10；6、8、9、10、11；8、9、10、11、12；组成的6个最大团图。

由水平区间图的实际意义可以看到，最大团图的元素数的最大值即为通道区的布线密度。

(2) 带区的概念

为了描述水平限制关系，我们也可以利用“带区”(zone)的概念[129]。带区是通道区中一个尽可能大的连续的列的集合，通过带区(包括通过带区中某些列)的任何二条线网都不允许布在同一条水平通道上，很明显，与带区相关的线网子集一定是水平区间图中的一个最大团图。

图7.21给出了图7.15所示通道区布线问题的带区的划分和

列	通过的线网子集	带区
1	1 3 4 5	
2	1 3 3 4 5	1
3	1 2 3 5	
4	1 2 6	2
5	2 6 7 8	
6	2 6 7 8 9	3
7	2 6 7 8 9	
8	2 6 8 9 10	4
9	6 8 9 10 11	5
10	8 9 10 11 12	
11	8 10 11 12	6
12	10 11 12	

(a) 带区的划分

	1	2	3	4	5	6
1			7	10		
2					11	
3	6					12
4	8					
5	9					

(b) 带区的描述

图 7.21 通道区布线带区划分和描述

带区的描述方式。

带区的划分可用下述算法实现：

设通道区共有 P 列，初始带区号 $n=1$，带区终止指针 $Q=1$。

划分过程为从通道区第一列到第 P 列，自左至右的扫描过程。在每一列作如下处理：

若第 j 列上存在着线网的起始接点且此时 $Q=1$，则第 n 个带区的起始位置为 j，第 $n-1$ 个带区的终止位置为 $j-1$(当 $n\neq1$ 时)，并置 $Q=0$。自然，最末一个带区的终止位置为 P。

若 j 列上存在着线网的终止接点且 $Q=0$ 时，则将 Q 置成 1 并使 $n=n+1$。

当扫描过程结束时，也就完成了带区的划分。

在实际的布线算法中，运用带区的概念常常可使处理效率得到一定的提高。

当采用曲干布线时，显然属于同一线网的子线网间不存在水平限制关系。

三、左边算法和曲干算法

1. 左边算法

左边算法(Left-edge)[145]是 Hashimato and Stevens1971 年提出的通道区布线算法，也是最早的通道区布线算法，这是一个应用启发式算法原则的三端模式算法。算法首先选定各水平通道布线的顺序，可选择的顺序有由下至上各水平通道依次布线，由上至下各水平通道依次布线等；在每一条水平通道进行布线时，可自左向右逐段进行或自右向左逐段进行；布线的顺序和方向就是该算法的算法条件。下面我们将以自下至上，自左至右的布线方向(简称左下)为例来说明算法的实现过程。

(1) 左边算法的主要思想

在通道区布线的垂直限制图中，出度为零的顶点所对应的线网称作活动线，显然，为满足垂直限制条件，只有当前的活动线才允许被布到当前的水平通道中去。在图 7.22 中，在开始布线时，只有线网 2、3、4、5 是活动线，并可布在第一条(最靠近通道区下边界的)水平通道中。

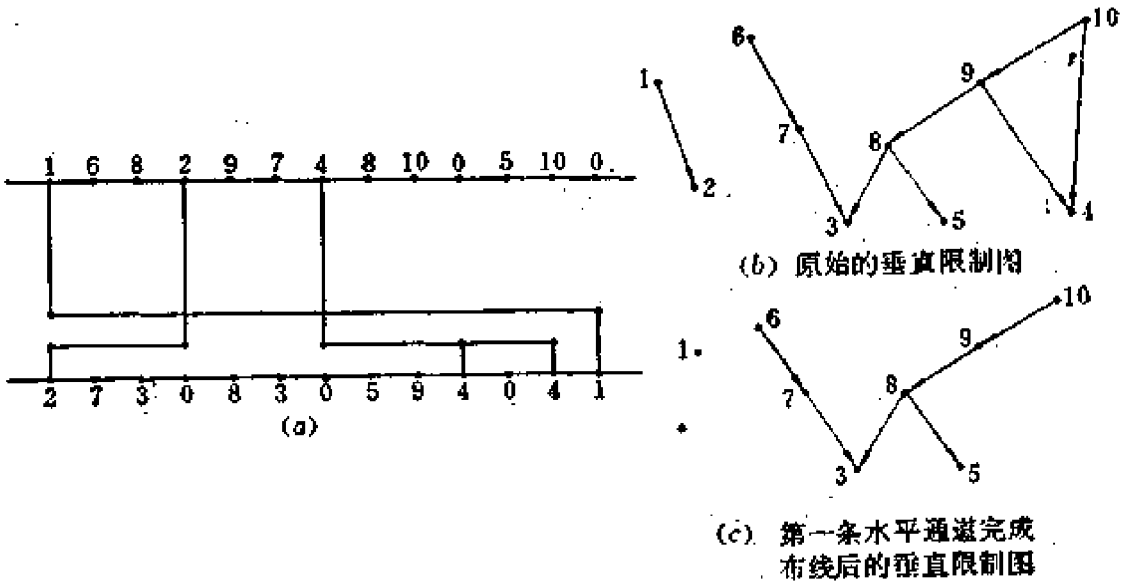


图 7.22 左边算法布线示意图

为了使完成布线所需的水平通道总数最小化，算法每次从不和已布通道段重迭的活动线中选取其左端点最靠左(即“左边”原则)的线网进行实际的布线。如图7.22所示，开始应选取线网2首先进行布线，然后选取线网4进行布线(此时，线网3与已布通道段发生重迭，故不可能再布在该水平通道中)。当完成了一条水平通道的布线后，可把已布线网对应的顶点和关联边从垂直限制图中删去，此时，将可能有一些线网成为新的活动线。图7.22(c)为布完第一条水平通道后的垂直限制图，线网1成为新的活动线。按上述处理原则进行第二条水平通道的布线……，直至所有线网都实现了布线为止。

可以看到：“左边”原则实际上是试图尽可能地利用当前的布线通道，从而使整个布线所需水平通道数最小化的一个启发性原则。可以证明：当通道区布线问题不存在垂直限制关系时，应用该算法布线，可获得所需水平通道数最少的最优解。文献[152]通过特定的工艺设计规则，增加了通道区的相对长度，使通道区上、下边界的接点都不处在同一个通道区列上，应用左边算法进行布线，获得了100%的布线成功率和通道区高度的最小化。

(2) 左边算法的改进

在存在着垂直限制关系时，左边算法常常不能得到最优的布线解。但从该算法的处理原则可以看到，该算法具有极高的处理效率，因而可以通过选择不同的布线方向，如左下，左上，右下，右上等，来获得一个布线的可行解集，并从中选择一个最好的解作为实际的布线解。需要注意的是当采用其他布线方向时，活动线概念和实际布线线网的选择原则将作一点修改。以“右上”为例，即水平通道的布线由上至下依次进行，每条水平通道中布线自右向左逐段进行，此时，活动线是指垂直限制图中入度为零的顶点对应的线网。选择实际布线线网则依右端点最靠右的原则进行。

不幸的是，对于有些通道区布线问题，左边算法即使改变算法条件也难以获得足够满意的结果，图7.23为这种情况一个简单的

实例。

① 增加一些启发性原则　改进的方法之一是布线时增加另一些启发性原则。引文[153]考虑到垂直限制图中最长限制链长对完成布线所需水平通道总数的影响，启发性地认为在每一条水平通道布线后，应尽可能地使垂直限制图中的L_{max}有所下降，因此，在每条水平通道开始布线时，首先从当前的活动线中选择处在最长限制链上的线网进行布线，然后在该线网的左边采用"右边"原则布线，在该线网的右边采用"左边"原则进行布线。在一些垂直限制关系比较简单的布线问题中，上述改进可望得到较优的结果。

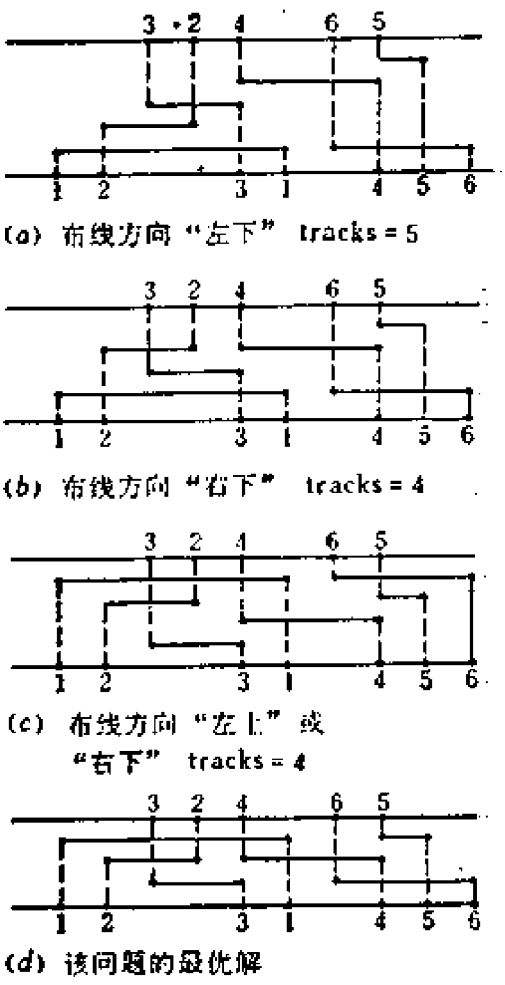


图7.23　左边算法不适应的情况

② 采用"曲干"布线　改进的另一种方法是采用"曲干"布线[130]。

2. 曲干算法

Deutsch 76年提出的曲干算法(dogleg algorithms)引入了子线网的概念[130]。一方面对含较多的多接点线网的通道区布线问题可望缩短其垂直限制图中的最大限制链长和减小其中可能存在的回路数，另一方面由于采用子线网的概念，可望更细致地进行水平通道段的分配，而将使布线更加合理。

采用曲干布线带来的问题是可能使布线所需的通孔数有所增加，因此在曲干算法中除布线方向这个算法条件外，还有一个控制通孔数量的子线网形式长度参量(Range)，这二个算法条件的具

体描述如下述：

① 布线方向 “曲干”算法中布线交替地由上、下(或下、上)边界向通道区中轴线逐个水平通道地进行。因此布线方向可能有8种可能的选择，即上左——下左、上左——下右、上右——下左、上右——下右、下左——上左、下左——上右、下右——上左、下右上右等。

② 子线网的形式长度参量R 子线网的形式长度可用子线网所含原线网连续的相邻接点的个数来度量，参量R的取值范围可由1至10。R为子线网所含原线网连续的相邻接点数减1，当$R=1$时，即原线网每二个相邻接点定义为一条子线网。

选择这二个算法条件的不同组合，理论上可产生80个可行的布线解，然后从中选择所需水平通道数最少且通孔数也较少的解作为实际的布线结果。由于曲干算法的处理原则是极其简捷的左边算法的原则，因此可以极高的处理效率求得每一个可行解，而且总的处理效率仍是令人满意的。

四、合并算法和区配算法

一种更好的三端模式通道区布线算法是1982年Yoshimura和葛守仁提出的合并算法和区配算法（merging and matching algorithm)[150]。

1. 合并算法

(1) 合并算法的主要思想

我们知道，根据通道区布线的水平限制条件，同一带区的线网将不允许被布到同一条水平通道中去，但不同带区的线网在满足垂直限制关系的条件下可利用同一条水平通道去进行布线，这种二条线网使用同一条水平通道实行布线的方式我们称之为“合并”。合并算法在布线时，不是把各个线网直接分配到各个水平通道中去，而是首先考虑各个线网之间的关系，从相邻的二个带区开始，尽量把能“合并”使用同一个水平通道的线网成对地合并起来，

以减少布线所需的通道数，然后依次地与其他相邻的带区进行“合并”处理，直至处理完全部带区，最后才根据这些线网间的“合并”信息实行实际的布线。

(2) 线网的合并条件

为了取得满意的结果，线网的合并处理必需考虑下述条件。

① 被合并的线网必须满足水平限制关系，即必须分属于二个不同的带区。为获得优化的解，应充分利用可利用的水平通道，同时为了处理的方便，应首先进行相邻带区中的线网的合并，然后依次地与其他相邻带区进行进一步的合并处理。

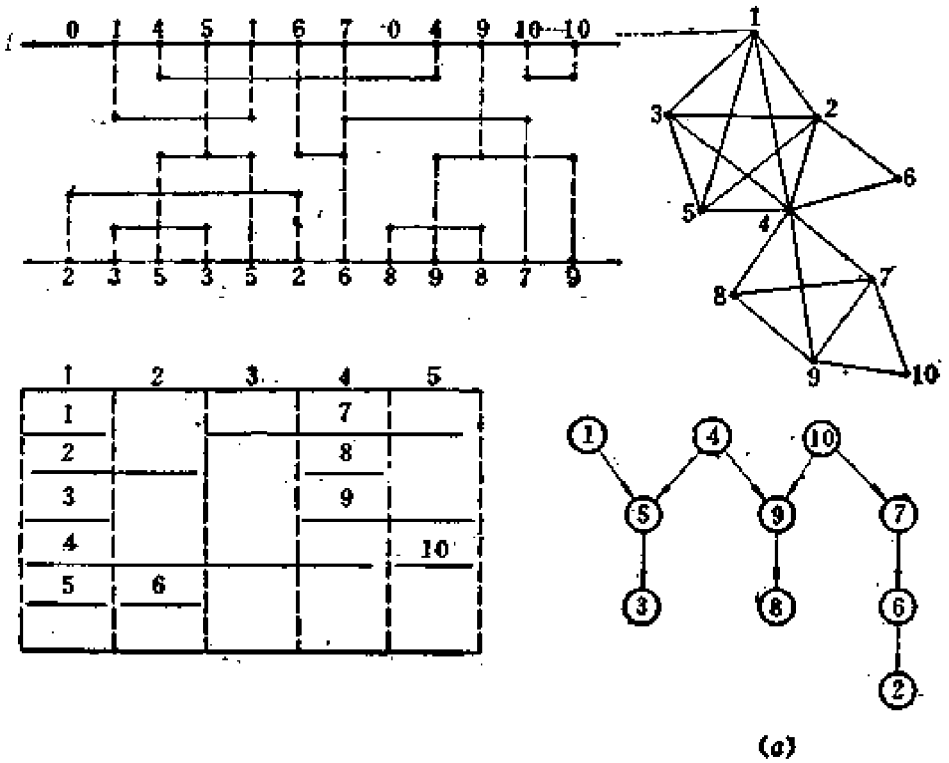
② 线网的合并不会改变通道区的布线密度，但线网的合并即相当于垂直限制图 G_V 中这些线网对应的顶点的合并，从而将使图 G_V 发生变化。为了避免合并后，在图 G_V 中产生回路，被合并的线网在垂直限制图 G_V 中对应的顶点间应不存在任何由有向边构成的路径，即对应的顶点应互不为上辈顶点和下辈顶点。

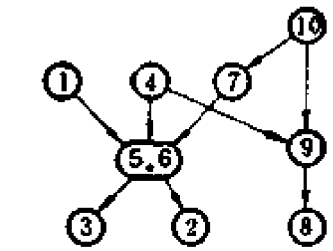
③ 由于最大限制链长 L_{max} 在不少情况中将严重地影响布线所需的水平通道数，因此线网合并后应尽可能使垂直限制图 G_V 中的 L_{max} 增加最少，这也是合并算法中主要要解决的问题之一。

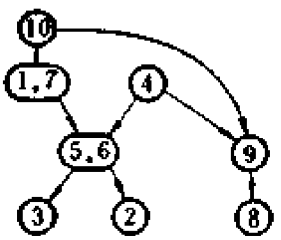
(3) 合并算法的布线过程

图 7.24 图示了一个简单的通道区运用合并算法布线的过程。从上述合并过程可以看到，条件 3 对保证最后布线解的精度是极其重要的。设想在第一次合并时，被合并线网为线网 1 和线网 6，则将使 G_V 图 L_{max} 变成 5，并将使最后结果所需水平通道数增加。考虑到第 3 个条件，合并算法在选择被合并线网时采用了一个启发式原则。该原则可描述如下。

设 Q 为通道区一个带区的相关线网集。P 为与其邻接的下一个带区的相关线网集。我们可以在垂直限制图中增加二个虚拟的顶点 S 和 t。分别表示有向树的源和目标点，如图 7.25 所示。可定义任一顶点 n 到源点的最长路径长度为 $u(n)$，到目标顶点的最长路径长度为 $d(n)$。图 7.25 中，$u(1)=1$, $u(3)=3$, $u(6)=2$, ···







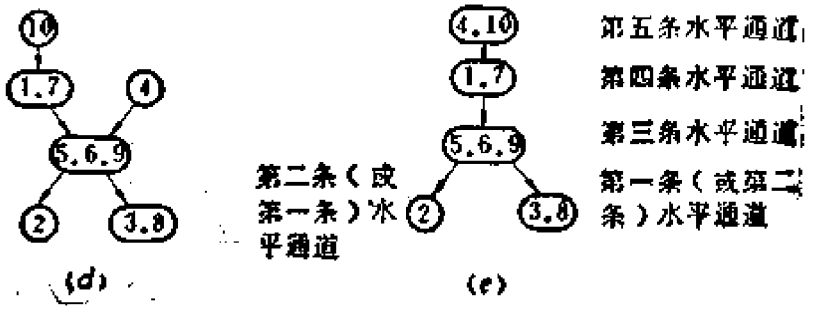


图 7.24 合并算法的处理过程

$d(1)=4$, $d(3)=2$, $d(6)=2\cdots$。

① 首先从Q集中选择被合并线网。选择原则是选取一条线网，使它对应的顶点在垂直限制图中尽可能处于最长的限制链上，并使其离虚拟的源或目标顶点最远。这些可通过利用一个选优函数$f(m)$, $m\in Q$来实现。

$$f(m)=C_\infty\times\{u(m)+d(m)\}+\mathrm{Max}\{u(m),\ d(m)\}$$

其中，$C_\infty\gg 1$。

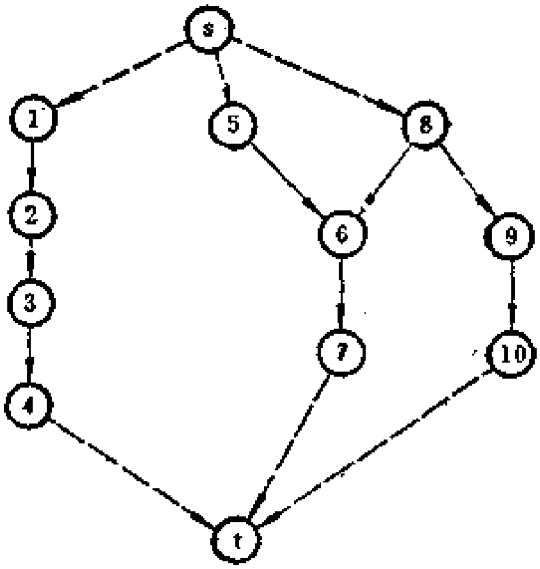


图7.25 增加源和目标点后的有向树

在Q集中选择具有$f(m)$最大值的线网作为被合并线网。

② 然后从P集中选取合并线网n。显然，合并线网应满足合并条件2，选择的合并线网应使实行合并后，垂直限制图中L_{max}增加最少。当符合这个条件的线网不唯一时，应选择处于最长限制链上且与m线网在限制链上位置具有对应性的线网优先进行合并，即使$u(m)/d(m)$尽可能等于$u(n)/d(n)$且$u(n)+d(n)$最大。这可利用另一个选优函数$g(n, m)$, $n\in P$来实现。

$$g(n,m)=C_\infty\times h(n,m)-\{\sqrt{u(m)\times u(n)}+\sqrt{d(m)\times d(n)}\}$$

其中，$h(n,m)=\mathrm{Max}\{u(n),u(m)\}+\mathrm{Max}\{d(n),d(m)\}-\mathrm{Max}\{u(n)+d(n),u(m)+d(m)\}$

其实际意义是合并后最大限制链长的增量。

$\sqrt{u(m)\times u(n)}+\sqrt{d(m)\times d(n)}$项是$u(n)+d(n)$和$u(n)/d(n)\approx u(m)/d(m)$二个条件通过平方根近似后的等价表示方式。

在P集中选择具有$g(n, m)$最小值的线网作为合并线网。当n线网和m线网合并后，修改垂直限制图，即把顶点n和顶点m合并为一个顶点(n, m)，然后从Q和P集中选择第二对合并线网

……，直至再也不能继续进行线网的合并为止。

在邻接的二个带区的合并完成后，可依次再与其它邻接带区进行进一步的合并，此时，已合并处理完的二个带区可看作是一个带区。

当通道区中线网的合并处理全部完成后，水平通道可根据修改过的垂直限制图很容易地进行分配而完成所有线网的实际布线如图 7.24(*e*) 所示。

(4) 合并算法存在的问题

合并算法具有相当高的处理效率，但算法的处理结果与选择的开始合并的带区有关，且与合并的方向(即先与右邻的带区合并还是先与左边的带区合并的选择)有关，因此往往通过选择不同的起始带区和合并方向以产生一个可行解集，再从中选择一个最好的解作为实际的解。在实际应用时，合并算法一般能获得令人满意的解，但对于复杂的通道区布线问题，求得的解将有较大的误差。主要的原因在于合并算法在合并线网时是逐对地进行的，因此就可能出现这样的情况，从当前来看一对线网的合并是合理的，而实际上，这对线网的合并将使其他线网丧失了可能合并的机会。如图 7.26 所示，如把线网 *a*, *d* 和线网 *b*, *e* 进行合并后，将发现线网 *f* 已无法和其它线网再进行合并了，结果将使布线的解需要 5 条水平通道，而实际上最优解仅需要 4 条水平通道。

2. 匹配算法

(1) 匹配算法的主要思想

匹配算法对上述问题作了改进，它的改进主要有二点：

① 在相邻的带区中，作线网合并处理，只有当线网的终点出现在邻接的待合并处理的带区中时，才考虑该线网的合并处理。从而使线网的合并能有更好的总体考虑。

② 用二分图"最佳"匹配来处理线网的合并，使尽可能多的线网能得到合并。

(2) 匹配二分图的建立及其应用

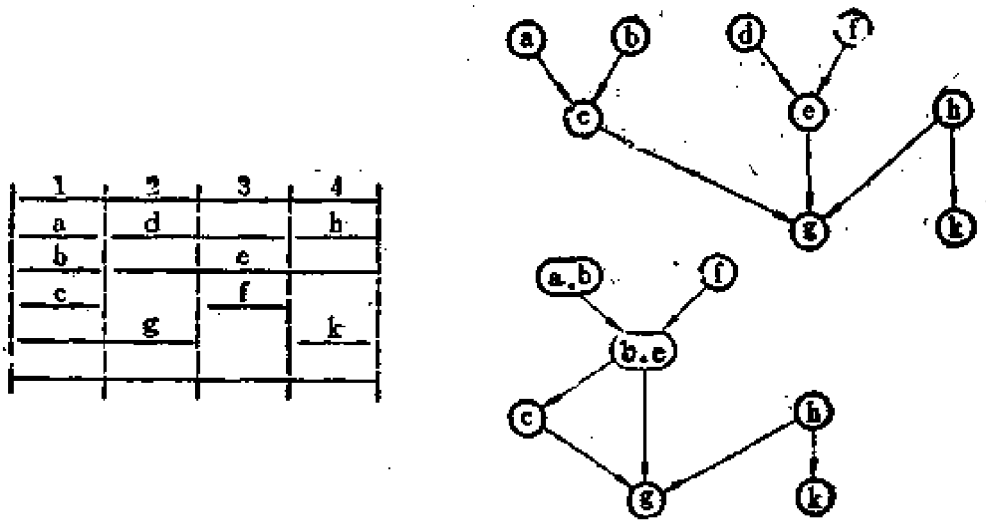


图 7.26　线网合并的一个假想的布线例子

用于“最佳”匹配的二分图 G_H 可用下述方法建立。设起始带区为 i 带区并向右开始匹配处理，对每一条终点在 i 带区的线网，加一个对应的顶点在 G_H 图的左边，其顶点集为 N_L；对每一条起点在 $i+1$ 带区的线网，加一个对应的顶点在 G_H 图的右边，其顶点集为 N_R；若 N_L 的元素 n_{Li} 和 N_R 的元素 n_{Rk} 可以合并，则在它们之间加一条边。为了简化处理的过程，将在建立图 G_H 时限制边的数量，一般限制每个顶点相关的边不多于 3 条。边的选择原则可采用合并算法中所采取的原则，即按 $f(m)$ 和 $g(n, m)$ 函数选择“最佳”的前三条边。

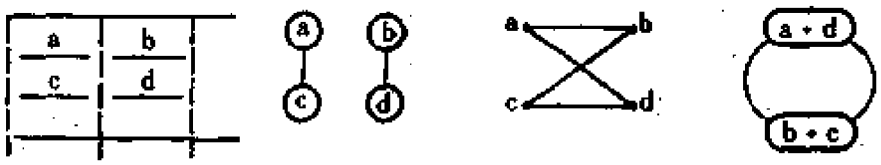


图 7.27　线网合并造成垂直限制图中出现回路的情形

这时产生的一个新的问题是：虽然 G_H 中每条边所表示的线网合并都是可行的，但当多对线网进行合并后，却可能使修改后的垂直限制图 G 中出现回路。图 7.27 就是一个简单的例子。

为了避免出现这种情况，引文[150]提出了一个算法，该算法

可如下述：

当给定二分图 $G_H(N, E_h)$和垂直限制图 $G_V(N, E_V)$后：

a_1：令 $E_x=0$，即结果集 E_x 为空集；当 N 集非空集时进行下面的处理。

a_2：令 N_0 为当前 G_V 图中入度为零的顶点集（即无上辈的顶点集）；令 E_0 是 G_H 图中的边子集，即

$$E_0=\{(i, j)|i, j\in N_0 \text{ 且 } (i, j)\in E_h\}$$

从 G_H 图中删去 E_0 集的所有元素。

a_3：如果 G_H 图中出现度数为零的顶点，转 a_5 处理，否则按 a_4 处理。

a_4：从 N_0 集中选择一个在 G_H 图中具有最小度数的顶点 V，并将当前与顶点 V 相关的边加入到 E_x 集中。

a_5：从图 G_H 和 G_V 中删去所有 G_H 图中度数为零的顶点或 a_4 中求得的 V 顶点及其关联的边。

重复上述 $a_2\sim a_5$ 的处理过程，直至 G_H 图成为一个空图。图 7.28 图示了该算法处理的过程。在图 7.28 的实例中，结果 E_x 集为$\{(a, h)\}$。

可以证明：当且仅当经上述算法求得的 E_x 集为空集时，图 G_H 的任何一个匹配所对应的合并都是可行的，即都不会使合并后的 G_V 图中出现回路。

一个重要的推论是：当 E_x 非空集时，图 G_H 的一个子图 $G'_H=(N, E_h-E_x)$中任一个匹配所对应的合并都是可行的。

由上述推论得到的一个“最佳”匹配算法如下述：

STEP1：对于已建立的二分图 G_H，可按上述算法求出 E_x 集。

STEP2：求 G_H 的一个子图 $G'_H=(N, E_h-E_x)$。

STEP3：在 G'_H 图中选择当前度数最小的顶点，优先进行匹配，然后从 G'_H 中删去已匹配的顶点及其关联边。

重复上述过程（STEP3）直至 G'_H 图不再存在可匹配的顶点对为止。图 7.29 为一个“最佳”匹配的实例。

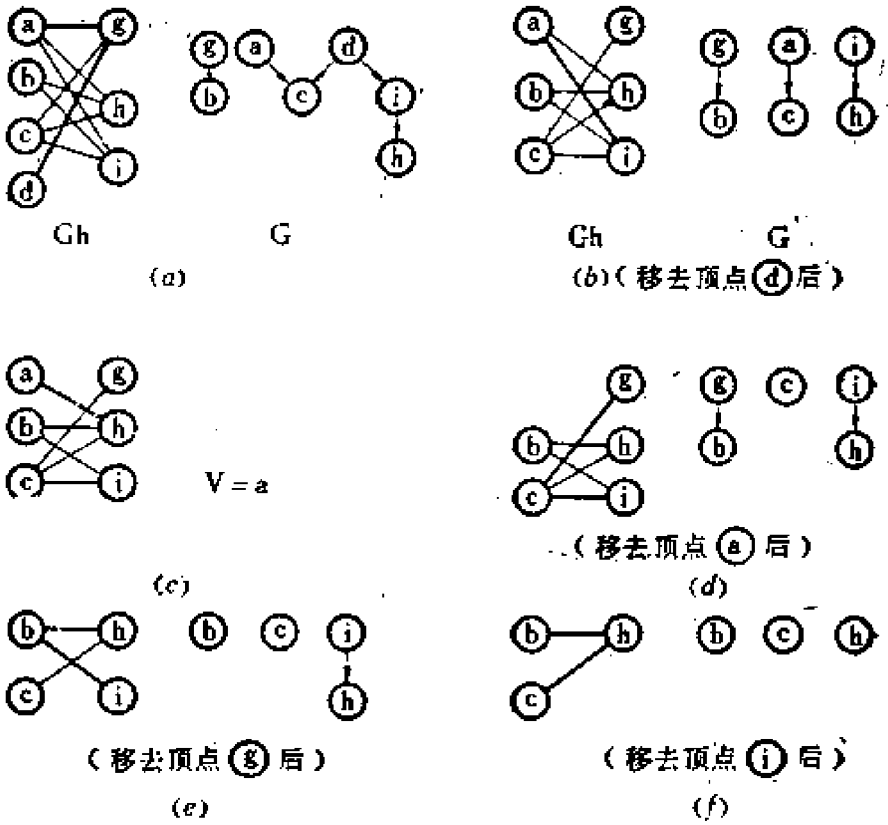


图 7.28　避免线网合并造成垂直限制图中出现回路的例子

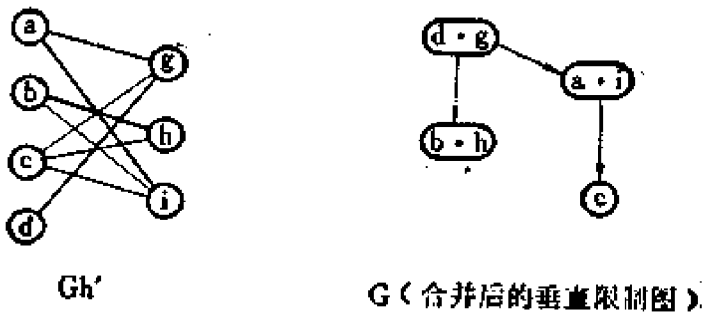


图 7.29　一个最佳匹配

相邻的二个带区的匹配完成后，可按其匹配结果实行布线的合并，将其相应的顶点从图 G_H 的右边删去，将 G_H 图左边的顶点修改为合并后的顶点，并修改垂直限制图 G_V，然后进行下一个带区的匹配处理。这个过程逐渐向远处扩展，直至全部通道区的带区都处理完为止，最后根据合并结果分配水平通道，完成实际的布线。

匹配算法是一个相当好的三端模式算法，通过选择不同的起始匹配带区和匹配方向，对于大多数通道区布线问题可求得实际上的最佳解。由于每次的求解过程具有相当高的处理效率，因此总的处理效率是令人满意的。

这二个算法都可以方便地扩展为曲干布线形式的布线算法。通过尽量(在满足其他优化条件的基础上)使属于同一线网的子线网优先匹配，可使布线的通孔数总量得到明显的减少。

五、“最佳化”算法和曲干“最佳化”算法

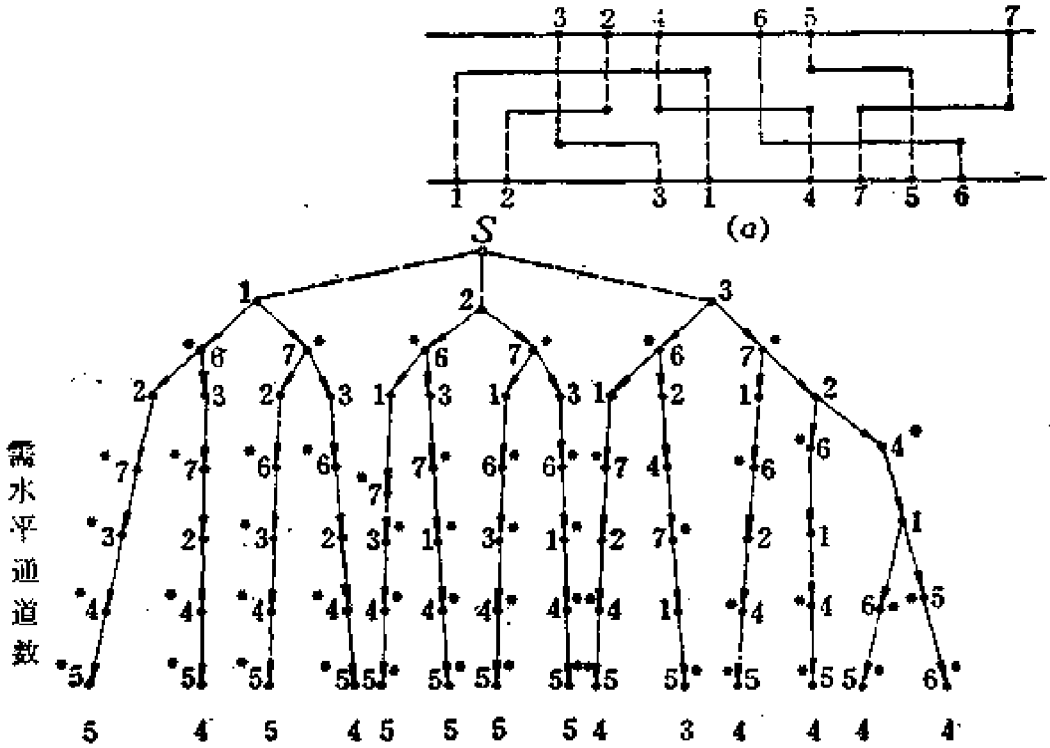
1. “最佳化”算法的主要思想

“最佳化”算法是一个典型的二端模式算法[129]。

对于一个垂直限制图中不存在回路的通道区布线问题，理论上总是可以穷举其所有的可行解，然后从中选优而得到最佳解。图7.30是一个简单的通道区布线问题和它的可行解搜索树。树上每一条由虚拟的树根到叶的有向路径都表示了一个可行解中各线网的布线顺序(由下至上，由左向右)，图 7.30(a) 为最优解的实际布线结果。穷举选优的方法仅对一些简单的通道区布线问题是实际可行的，当问题比较复杂，线网数比较多时，可行解搜索树将变得极其庞大，而无法以经济的计算时间获得最优的结果。

1973 年 Kernighan 等成功地应用分支限界法[129](branch and bound) 构造了所谓的“最佳化”算法。其主要思想可阐述如下：

开始时，可用其它方法(左边算法或改进的左边算法等)求得



(b)"•"号处表示一条水平通道已布满

图 7.30 最优解实际布线结果

一个较好的初始解（令其所需的水平通道总数为 V），虽然初始解的精度不会影响最后结果的精度，但它对分支限界过程的处理效率有严重的影响。初始解精度愈高，分支限界过程的处理效率亦越高。

在自下至上，自左至右逐条水平通道、逐段地进行布线时，将当前垂直限制图中的活动线网依其左端点的位置从左至右排列成一个序列。每条水平通道布线时，将未尝试过布线的活动线网在满足水平限制关系的条件下自左至右地布到当前的水平通道中去。为了计算的方便，只在当前的水平通道布满时（即搜索树上标"*"处），计算所有未布线网全部完成布线时所需的水平通道数的下界值 b。令当前已布水平通道数为 t，则可根据 $t+b$ 的值决定后续的处理过程。

当 $t+b \geqslant V$ 时，则从该"*"处顶点以下的各分枝对应的可行解子集中不可能获得一个比 V 更佳的解，因此这些分支可以被剪去

而不用继续搜索了。在图7.30的布线问题中，设初始解所需水平通道数为4，当第一条水平通道中已布线网为线网1和6（搜索树上左边第一个分支）时，若求得余下的线网实现布线所需的水平通道数下界为3，则 $t+b=1+3=4=V$。因此，第一分支对应的二个可行解不可能优于初始解而不用再继续在此分支上继续搜索了。在这种情况下，我们可拆去当前通道上最后一条线网并用下一个最左边的活动线网代替它进行布线，在图7.30中，即用线网7去代替线网6进行布线，在搜索树上即为转到另一条新的分支上去继续搜索。如果已不存在可替代线网时，则回到当前水平通道的前一条线网（线网1）上去进行迭代重布。如此递归，在需要时可一直退回到第一条水平通道的最左端。经过上述处理，我们或者可以找到一个替代对象而继续进行分支限界过程并找到一个比当前解更优的可行解，或者可以排除所有的可行解而证明当前解就是最优解。

当 $t+b<V$ 时，说明该分支对应的可行解中可能存在着一个比当前解更优的解，可将 $t=t+1$ 继续进行下一条水平通道的布线。如果全部线网都已完成了布线，我们就找到了一个比当前解（或初始解）更优的解。可令该解为当前新的布线解，然后回到最后布线的那个线网上，用它的替代对象去继续进行迭代、搜索更优的可行解，直至所有的可行解都被搜索过或被排除为止。最终得到的结果一定是所需水平通道数最少的最佳解。

可以看到，“最佳化”算法实质上是一个深度优先搜索的分支限界过程。不同的是并不是在每个分支点上进行限界的处理，而只在每条水平通道布满时（即搜索树上用“*”号标出的点处）才进行限界的计算和处理。此时，余下的线网的布线问题可以被看作是一个修改过的通道区布线问题，而给限界的计算带来了方便。

2. 限界的定义及计算方法

在上述算法中，限界的计算精度和效率是一个关键问题。如果限界的误差很大（即比实际所需通道数小得太多）时，上述算法

就成为一个穷举的过程，而使算法丧失实际意义。

在通道区布线问题的限制关系讨论中，我们知道：从垂直限制关系来看，通道区布线所需水平通道数必将大于、等于最大限制链长 L_{max}；而从水平限制关系来看，通道区布线所需水平通道数必将大于、等于通道区布线密度 D；因此 $B_1 = \mathrm{Max}\{L_{max}, D\}$ 将是通道区布线所需水平通道数的一个下界。然而由于 L_{max} 和 D 都只是分别地考虑了垂直或水平的限制关系，而未能统一地考虑它们相互的影响。因此，在不少情况下 B_1 只是一个较"松"的下界。如图 7.31 所示，通道区布线问题的 L_{max} 和 D 分别等于 2，但实际上该问题所需的水平通道数至少等于 3。

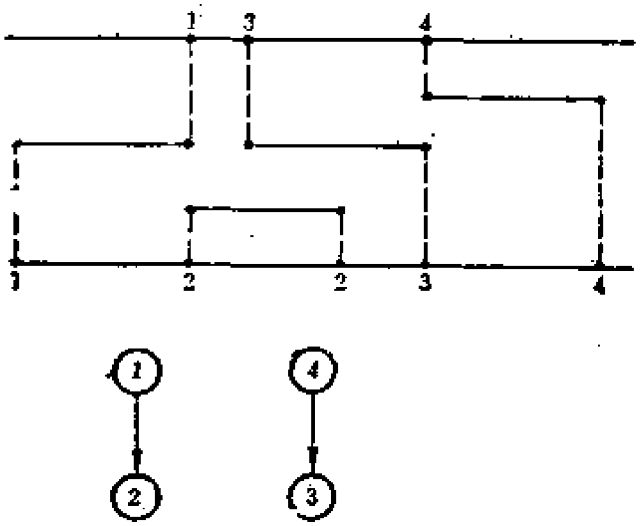


图 7.31　限界计算法例

在"最佳化"算法中提出了一种计算更"紧"的限界的方法。

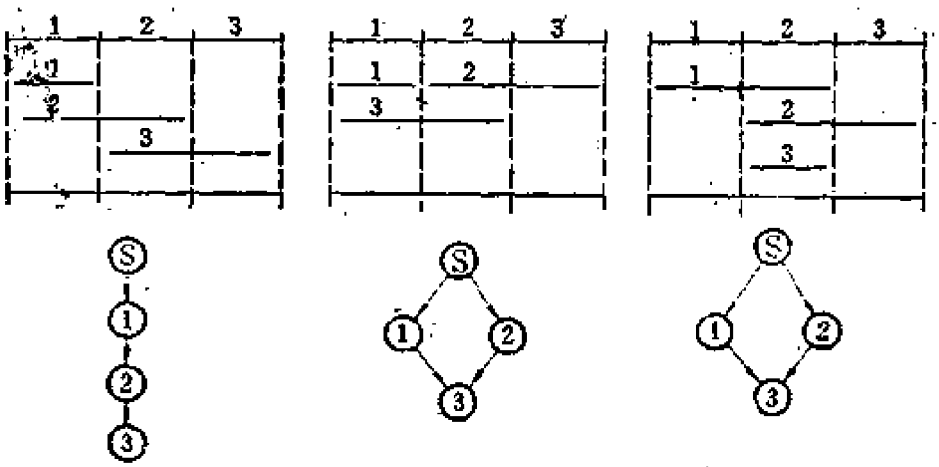
算法中引入了二种限界的定义，即定界和动界 (static bound and dynamic bound)。

(1) 定界和定界的计算

定界是对每一条线网分别定义的。每条线网当且仅当考虑它和它的所有上辈线网时，完成布线所需的水平通道数的下界定义为该线网的定界。

严格的定界计算将相当复杂，但定界的简单近似值可等于该线网在垂直限制图G中对应的顶点至虚拟源顶点的最长距离（定义各边权重为1)B_S。

这种计算方法在有些情况下得到的定界将是一个“松”的下界。如图7.32(c)中实际线网3定界应为3，而计算结果为2。在不少情况下，这种误差在动界计算中可得到补偿和修正。从计算效率考虑，实际算法中将采用上述近似计算方法。



(a) 线网3定界 = 3，B_S = 3 (b) 线网3定界 = 2，B_S = 2 (c) 线网3定界 = 3，B_S = 2

图7.32 定界计算法示例

(2) 动界和动界的计算

动界是未布线网集完成布线时所需水平通道数的下界。

“最佳化”算法用一种综合考虑垂直限制关系和水平限制关系的动界算法来求得一个更“紧”的下界值。动界算法可如下述：

首先对当前的未布线网集定义一个二维数组$M(I, J)$，其中行对应着线网的定界值，列对应着通道区的带区或通道区列，各数组元素初始值都为零，然后把各未布线网逐条装入$M(I, J)$，令K线网定界为S，其右端点在带区R中，左端点在带区L中，装入时则将$M(S, L)\cdots M(S, R)$各元素分别加1。

装入过程全部完成后，可依下法求出各列的列动界$b(j)$：令j列线网实际最大定界值为S_j，初始$b(j)=0$。

for $i=1$ to S,

$b(j)=\mathrm{Max}\{i, b(j)-M(i, j)\}$;

可以看到 $b(j)$ 的计算是一个递归过程，它较好地综合考虑了垂直限制关系和水平限制关系之间的相互影响。

未布线集的动界 DB 等于各列列动界的最大值，即 $DB=\mathrm{Max}\{b(j)\}$。

实验结果表明，“最佳化”算法在直干布线时，对垂直限制图中无回路的通道区布线问题有相当高的求解精度。对相当多的问题也具有足够高的处理效率，但对有些问题则计算时间相当长，典型的问题如 Bell 实验室提出的 “Difficult example”通道区布线问题，处理 4 小时后分支限界过程仍未结束，当采用曲干布线方式时，效率问题将更加严重。

3．曲干“最佳化”算法

1981 年 M. M. Wada 在提高 “最佳化” 算法的处理效率方面作了一些改进并将该算法扩展为曲干布线方式[149]，一般可提高处理效率 4～7 倍。

Wada 所作的三点改进是：

① 对替代线网进行了合理的限制。由通道区布线的水平限制关系出发，当线网 a 需拆去时，只有那些左端点被 a 线网覆盖的未布活动线网作为它的替代者才具有实际意义，因此，当线网拆除时，可限制其替代对象仅仅是那些左端点被其覆盖的未布活动线网，从而减少了搜索次数，提高了处理效率。

② 被拆线网的决定。在每条水平通道布满后计算动界时，对各列列动界进行记录并求出 $b(j)=DB$ 且最左边的列的位置 J。为使迭代布线后 DB 值能更快地收敛下降，我们可以从右至左一次拆去直到 J 列(带区)的本水平通道中的所有线网，然后再从该部分开始迭代布线，继续分支限界的搜索过程。显然，这也将使总的处理效率得到提高。

③ 虚拟初始解的运用。在实际应用时，为获得较高的处理效

率，可采用虚拟初始解——搜索验证的方法。在分支限界过程开始时，以某个所需水平通道数的下界值 V（如通道区布线密度 D 或 L_{max} 也可以是初始动界值）作为假想的初始解所需的水平通道数。在分支限界搜索过程中，当 $t+b \leqslant V$ 时，都需继续搜索。如能获得所需水平通道数等于 V 的可行解，显然已求得了最优解。如果分支限界过程结束，不存在等于 V 的可行解，则 $V=V+1$ 作为新的虚拟初始解，重复上述过程，由于初始解极其"优化"而将使分支限界过程可相当快地剪去大部分的分支，迅速验证它的实际存在性，从而可获得较高的处理效率。当虚拟初始解修正后变得愈来愈大时，验证的时间也将变得愈来愈长，因此，通常限制每个虚拟解搜索验证的次数，当超过该次数时，即认为该虚拟解不存在着实际的可行解。

上述措施在实践中有相当的有效性，但在有些情况下，由于限定搜索的次数，将可能丢失最优解。

经过改进的"最佳化"算法处理效率有明显的提高，对"difficult example"通道区布线问题，在直干布线时，求得解为 28 条水平通道，花费机时 430 秒，在曲干布线时，求得解为 21 条水平通道（已证明，至少存在着只需 20 条水平通道的更优解）。

六、通道区布线的通道损益分析法[151]

通道区布线的通道损益分析法 [151] 是作者根据王守觉同志关于从总体分析着手解决布线问题的思想，在 1982 年提出的一个新的通道区布线算法。这也是一个二端模式算法。该算法从活动线网的通道竞争出发，运用有效覆盖的总体分析方法，直接以完成布线所需水平通道数最少化为目标进行选优布线。由于该算法在布线时具有较好的总体分析并避免了迭代布线，从而对已发表的布线实例在无迂回布线的情况下都以相当高的处理效率取得了令人满意的结果。

1. 若干定义和性质

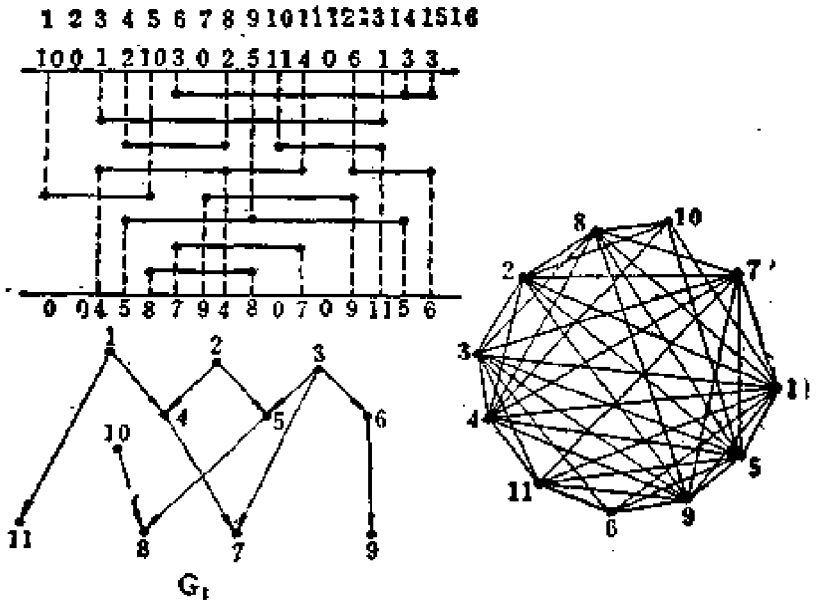


图 7.33 一个实际的通道布线示意图

为了阐述的方便，我们首先给出一些定义并讨论一些有关的性质。

图 7.33 为一实际的通道区布线问题及其垂直限制图 G_V 和水平区间图 G_I。

定义 1：在不存在回路的 G_V 中，如从顶点 A 至少存在着一条到达顶点 V 的路径，则称 A 线网为 V 线网的释放线。线网 V 的所有释放线的集合称为 V 的释放线集。

如图 7.33 线网 1 即为线网 4，7，11 的释放线。而线网 2、3、5、10 即为线网 8 的释放线集。

定义 2：设活动线相关集 S 为 $\{L_a, L_b, \cdots L_i\}$，则 S 满足：

① S 集诸元素都是当前的活动线；

② S 集诸元素水平线段的交集 $T \neq 0$；

③ 若 $\mathcal{G}$ 为活动线且 $\mathcal{G} \notin S$，则 $\mathcal{G} \cap T = 0$。

其中 T 称为竞争通道，设 L_a 线网二端 x 坐标为 x_{a1}，x_{a2}，且 $x_{a2} \geqslant x_{a1}$。则 T 是这样一个闭区间 $[X_1, X_2]$，其中：

$$X_1 = \mathrm{Max}\{x_{a1}, x_{b1}, \cdots x_{i1}\}$$

$$X_2 = \min\{x_{a2}, x_{b2}, \cdots x_{t2}\}。$$

在图7.33中，当前的活动线网为线网11、8、7、9，其中线网7、8、9为一个活动线相关集。T为通道上第7列和第9列之间的闭区间。

显然，根据水平限制条件，在每一条竞争通道上只能安置活动线相关集中的一个元素。当活动线相关集元素数大于1时，就存在着一个选优安置(布线)的问题。

定义3：在不存在回路的垂直限制图中，设各边权重为1，线网P的先行级定义为由垂直限制图虚拟源顶点到达顶点P的最长路径长度。

为了描述的方便，我们在讨论分析布线的方法时，采用由通道区下边界开始逐个水平通道地向上布线、在每一个水平通道中自左向右逐段地布线的方式。需要说明的是，原则上布线的方向不会影响最终的结果。

定界(static bound)的概念：在引文[129]中对线网定界作了定义和讨论。所谓线网V的定界也可理解为当仅考虑线网V及其释放线集时，布线所需通道数的下界。这时可简单地把V线网的先行级作为其定界。但由于忽略了水平线段的重迭限制条件及其它因素的影响，其结果在许多情况下是不够精确的(即是比较"松"的下界)。我们也可用下述算法来计算定界，可使大部分线网的定界计算是精确的。

① 以先行级为计算线网定界的序。即先计算先行级小的线网的定界。先行级为1的线网其定界也为1。

② 对当前需求界的线网的释放线集，建立一数组$M(i, j)$。其中行对应着线网定界，列对应着$ZONE$或布线区列号。将每一释放线$[L, R]$对应的$M(SB, L)\cdots M(SB, R)$各元素分别加1。其中SB为该释放线定界。

③ j列限界可依下法求得：

$$Sb(j) = 0$$

$FOR\ i=1\cdots$该列实际定界最大值

$$Sb(j)=\max\{i,\ (Sb(j)+M(i,\ j))\},$$

④ 各列限界的极大值加1即为该线网的定界。

上述方法所得结果并不是绝对精确的。如图7.34(a)线网5计算得到的定界值为4，而实际应为5。但上述方法对图7.34(b)，(c)等情况来讲，都是精确的。

colvmn		1	2	3	4	5	6
定界	1	1	2	2	1	1	1
	2			1	1		
	3				1	1	
net 5 定界			3	4	4	4	2

column		1	2	3	4	5
定界	1	1	2	3	3	2
net 4 定界			3	4	4	3

column		1	2	3	4	5	6	7
定界	1	1	2	2	2	2	1	1
	2			1	1			
	3				1	1	1	
net5定界		2	3	4	5	4	4	

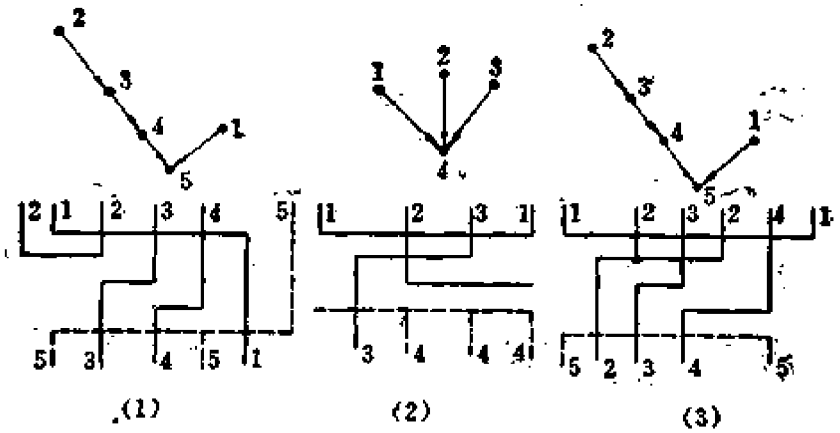


图7.34 定界计算

动界(dynamic bound)的概念：在引文[129]中，动界是作为当第n层通道布完后，预估完成全部布线尚需通道数的下界。其计算方法是对所有未布线建立$M(i, j)$数组，求各列限界的最大值即为当前的动界。设$[j_{L_1}, j_{L_2}]$为列的一个连续子集，则j_{L_1}到j_{L_2}各列限界的最大值即为$[j_{L_1}, j_{L_2}]$区间内所需通道数的下界。

定义4：设J列限界等于当前动界值。线网A覆盖J列且若删除线网A，则J列限界将减少1，则称线网A为有效线网。若A同时为活动线时，则称其对j列的覆盖为有效覆盖。

性质1：在完成布线时，若在每一层通道上，都存在一个不相关活动线的子集对当前所有等于动界值的列为有效覆盖，则得到的解所需通道数等于初始动界值，且一定为可行的最佳解。

由性质1推理可得性质2。

性质2：设当第n层通道布线时，当前任何不相关活动线的子集都不能对所有等于动界值的列实现有效覆盖，则完成布线后解的通道数一定大于或等于初始动界值加1。

在图7.34(a)中设线网1，2，3，4为未布线时，此时不存在任何不相关活动线的子集对所有等于动界值的列实现有效覆盖（注意：线网1不是有效线网，因此它对第5列的覆盖不是有效覆盖）。因此，线网1，2，3，4布线所需通道数的下限应为$3+1=4$。

由性质1进一步推理，可得性质3。

性质3：若在完成布线时，至少在k条通道上出现性质2所述情况，则给定问题的最佳解的通道数为初始动界值加k。

性质2、3说明用动界作为解的限界时，这样的限界并不一定是一个"紧"的限界。这也是用分支限界法求解时，对一些问题计算时间相当长的主要原因之一。在本算法中，动界值不再被认为是解的限界，而只是用来作为判定在本通道中是否存在一个不相关活动线的子集对所有等于动界值的列为有效覆盖的定量分析值。即如果存在一个不相关活动线的子集在本通道布线后，未布线的DB值比本通道布线前的DB值减少1，则据定义5，该子集

在本通道上一定完成了有效覆盖。此外，本算法是在相关活动线竞争选优时采用动界值分析，因此，这时的动界值可以仅看作是一个相对值，动界值和实际解所需通道数的绝对误差在相对比较时在很多情况中被抵消了。从而提高了分析的精度，保证了求解的精确性。

2. 通道损益分析法的布线过程

在通道损益分析法中，在每一个通道段进行布线前，将对竞争通道相关的活动线相关集中每一个元素进行预后分析。即分析每一个元素(线网)若布在当前的竞争通道段上后对完成全部布线时所需水平通道数的影响。

(1) 活动线的预后分析过程

设当前布线层次为 n，当前动界值为 DB_n。本通道已布线右端点为 j_r，与 j_r 紧邻的竞争通道对应的相关活动线集为 S。L 为 S 的一个元素，L_i 的预后分析分二部分进行。

第一部分分析通道段 $(j_r, j_i]$，其中 j_i 是 L_i 的右端点列号。此时只需设 L_i 已布，依动界定义，求 $(j_r, j_i]$ 区间各列限界最大值，设结果为 DB_1。

第二部分分析通道段 $(j_i, j_{end}]$，其中 j_{end} 为通道区最右端列号。此时，为获得最佳可行解而允许布到第 n 层通道 $(j_i, j_{end}]$ 段的线网应满足：

① 这些线网应是不相关活动线；

② 这些线网对 $(j_i, j_{end}]$ 段所有列限界等于 DB_n 的列应尽可能实现有效覆盖。

设 $N(i)$ 为候选活动线集。P_{in} 为装入指示字。$\mathrm{Max}DB_S$ 为子段限界极值。$(j_i, j_{end}]$ 段分析前，$N(i)$ 为空集，$\mathrm{Max}DB_S=0$。据前文所述定义和性质，j 列 $(j_i<j\leqslant j_{end}]$ 实际限界的求法如下述：

a) 如 $j-1$ 列为 $N(i)$ 中一元素终点时，则从 $N(i)$ 中删去该元素。并令 $\mathrm{Max}DB_S=0$。

b) 当 $\mathrm{Max}DB_S \geqslant DB_n-1$ 且 $N(i)$ 非空集时，装入指示字 P_{in}

$=1$。

c) 当 j 列为一有效活动线起点时，且 $P_{in}=0$ 则将该线网号装入 $N(i)$ 中。

d) 据动界算法，求 j 列限界 $DB^*(j)$。

e) j 列实际限界 $DB(j)$ 为：

当 $N(i)$ 为空集时，$DB(j)=DB^*(j)$；

当 $N(i)$ 非空集时，$BD(j)=DB^*(j)-1$。

f) 当 $DB(j)>\max DB_S$ 时，$\max\ DB_S=DB(j)$。

求出 $(j_i, j_{end}]$ 段各列限界最大值，设结果为 DB_2，则活动线 L_i 的有效覆盖定量分析值 ECA_i 为 $\max\{DB_1, DB_2\}$。

(2) 布线过程

通过上面的讨论和准备，通道损益分析法的布线过程可如下述：

对给定的布线要求，设算法的分析和布线顺序规定为由下至上，由左至右，逐层逐段进行。并设在垂直限制有向图 G_V 中不存在回路。

布线通道号 n 初始值为 1。

① 建立布线要求对应的 G_V 图，求各线网的定界。

② 计算当前 DB_n，并令已布线右端点 $j_r=0$。

③ 求与 j_r 紧邻的竞争通道对应的相关活动线集 S。

④ 当 S 为空集且未布线集为空集时，布线过程结束。

当 S 为空集而未布线集非空集时；n 加 1，转第 2 步。

当 S 集元素数等于 1 时；对该线进行实际布线。$j_r=j_i$，转第 5 步。

当 S 集元素数大于 1 时；分别设其中一元素为已布线，求其有效覆盖定量分析值 ECA_i。

对具有最小的 ECA_i 值的线网 L_i 进行实际布线。$j_r=j_i$，转第 5 步。

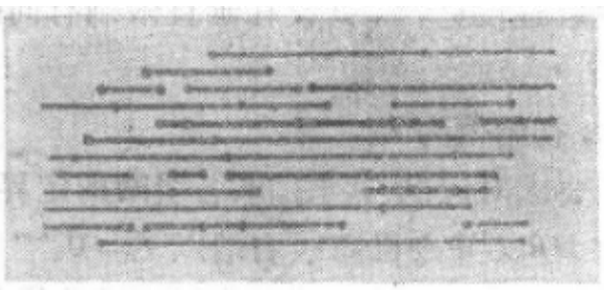
⑤ 从垂直限制有向图 G_V 中删去该线网映射的顶点及其关联

的边。转第3步。

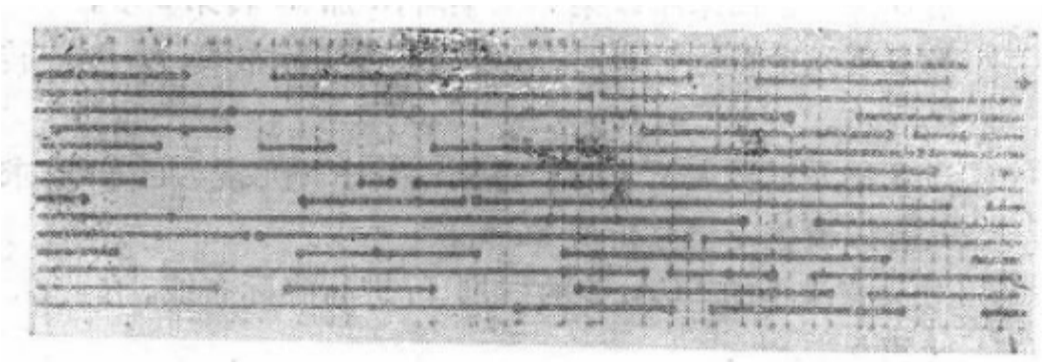
由上可见，算法过程是相当简捷的。需要说明的是：在计算 DB_n 时，若所有未布线定界都等于1，则自动转向用左端算法[145]来处理当前未布线集。据性质1，3，为了获得令人满意的结果，当 S 集中 ECA 值最小的线网不唯一时，应选取先行级或定界大的线网先布。这一点当采用先行级作为线网定界时，尤为重要。

上述算法也可方便地用于 dogleg 方式的布线。此时，所有线网都用其相邻接点构成的子线网来代替。需注意的是，这时原属同一线网的子线网的水平线段允许重迭。同样，属于同一线网的子线网的接点间不存在垂直限制关系。虽然由于引入了子线网的概念而需采取一些相应的措施，但其分析和布线的方法与非 dog-leg 情况原则上是相同的。

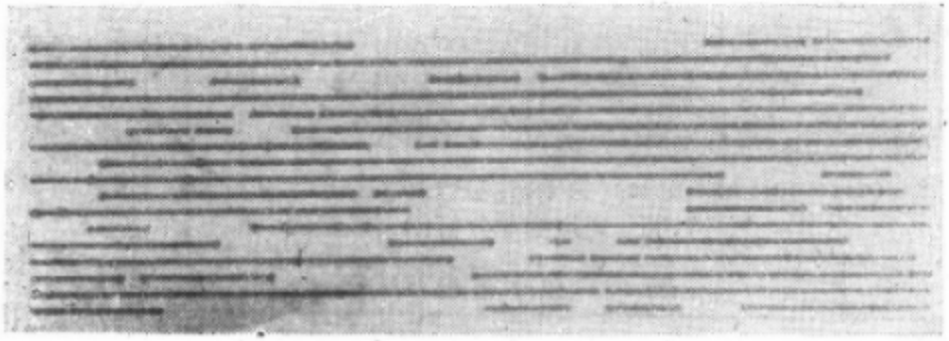
图7.35为通道损益分析法的一些布线实例照片，有关的实验数据可参看表7.1和表7.2。



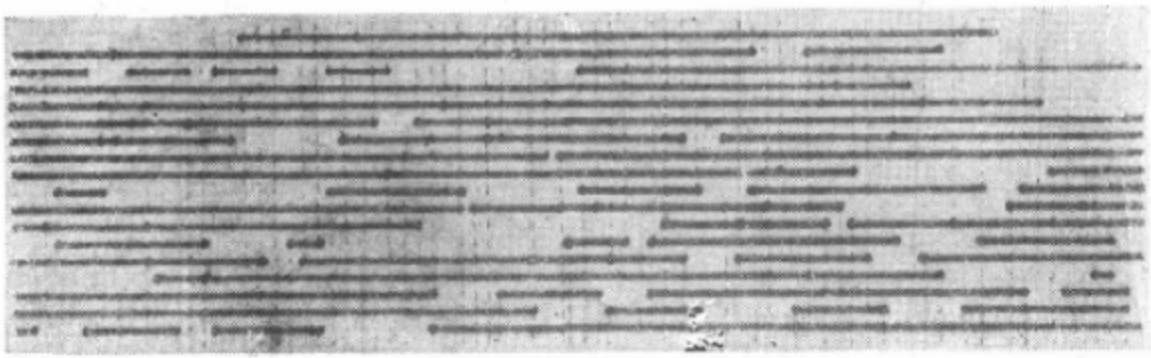
例1 (ex.1)通道区密度=12 所需通道数=12



例2 (ex.3a)通道区密度=15 所需通道数=15

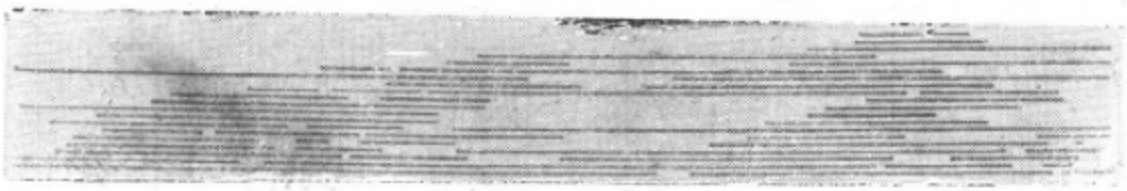


例 3 (*ex.3b*)通道区密度=17　所需通道数=17

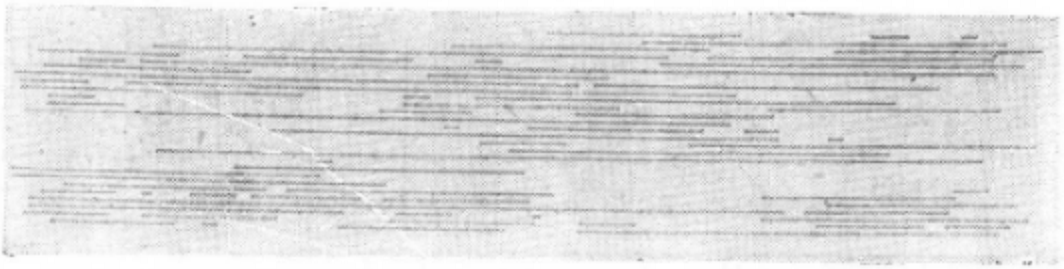


例 4　(*ex.3c*)通道区密度=18　所需通道数=18

例 5　(*ex.4b*)通道区密度=17　所需通道数=17



例 6　(*ex.5*)通道区密度=20　所需通道数=20



例 7　(difficult example)(直干布线)通道区密度=19　所需通道数=28

例8 (difficult example)(曲干布线)通道区密度=19 所需通道数=20

图 7.35 通道损益分析法布线实例

表 7.1 直干布线方式

	问题	题号	ex.1	ex.3a	ex.3b	ex.3c	ex.4b	ex.5	dif. ex
		布线密度	12	15	17	18	17	20	19
发表日期		线网数	21	45	48	53	55	62	72
1971	左边算法	所需通道数	14	18	20	19	23	22	39
1973	"最佳化"	所需通道数	12	15	17	18	17	20	23
	算法	所需机时	0.03s	5.1s	50s	0.5s	112s	0.5s	4小时*
1981	wada's 算法	所需通道数							28
		所需机时							430s
1982	合并算法	所需通道数	12	15	17	18	17	20	30
		所需机时	0.05s	0.07s	0.17s	0.16s	0.23s	0.22s	0.40s
1982	匹配算法	所需通道数	12	15	17	18	17	20	23
		所需机时	0.07s	0.43s	0.57s	0.72s	0.77s	0.88s	1.60s
1983	通道损益分析法	所需通道数	12	15	17	18	17	20	28
		所需机时	0.25s	0.67s	0.61s	0.89s	0.85s	1.48s	1.92s

* 该算法在 HP—2100 机上 4 小时后分支限界过程仍未结束。

表 7.2 曲干布线方式

发表日期	算法名称	"dif·ex"布线问题结果	
		所需通道数	所需机时
1976	"曲干"算法	21	
1981	"曲干"最优化算法	21	7430s*
1982	合并算法	21	1.0s
1982	匹配算法	20	2.1s
1983	通道损益分析法	20	13s

* 为估计值。

七、通道区布线算法小结

1. 通道区布线算法在布图设计中的地位

经过人们十几年的努力，通道区布线算法得到了很大的发展。目前，通道区布线算法已被广泛地应用于各种自动设计模式的布图设计中，并得到人们越来越大的重视。这主要是由于以下的二个原因：

① 在 LSI/VLSI 自动布图设计中，芯片中约 50% 的区域将是布线区。因此，为了提高设计的芯片集成度和提高电路的成品率，除完善其他部分的设计算法如布局、总体布线等算法外，有一个高精度的实际布线算法也是极其关键的。而通道区布线目前已有的算法已可保证使绝大部分通道区布线问题获得最佳解或准最佳解。

② 在 LSI/VLSI 自动布图设计中，随着设计规模的迅速增大，设计效率也成为人们越来越关注的问题。而在能够利用通道区布线的任何场合，人们可以发现，通道区布线方法将比其它布线方法具有高得多的处理效率。

表 7.1 和表 7.2 为一些通道区布线算法处理一些典型的布线问题时的实验结果。

值得注意的是，表列的算法求解时间，由于算法模式的不同而具有不同的意义。对于二端算法来讲(如"最佳化"算法，曲干"最佳化"算法即 Wada's 算法和通道损益分析法等) 表列的时间即为实际求解时间。而对于三端算法求解来讲(如左边算法，曲干算法、合并算法和匹配算法等)表列时间为一次处理所需机时。

2．通道区布线算法的进一步发展

在实际布线时若采用多层布线（如三层布线通道区布线[137])、迂回布线(如 1983 年 10 月 Burstein 利用整数规划实现的分级布线算法[131])、非横竖分层的布线方法(或称"非正常"方式布线——wrong way jog[117])，以及改变集成电路工艺规则，使通道区上、下边界接点都不处于同一列的无垂直限制条件下的布线 [152] 都可望获得所需通道区面积更小的布线解。其中一些手段对于固定通道区的门阵列设计模式具有更现实的意义。

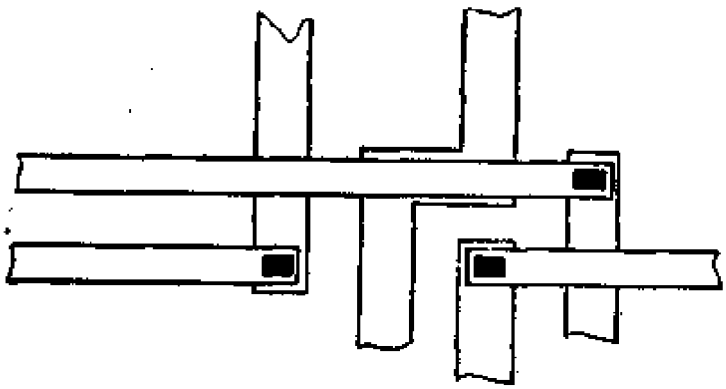


图 7.36　非横竖分层布线方法

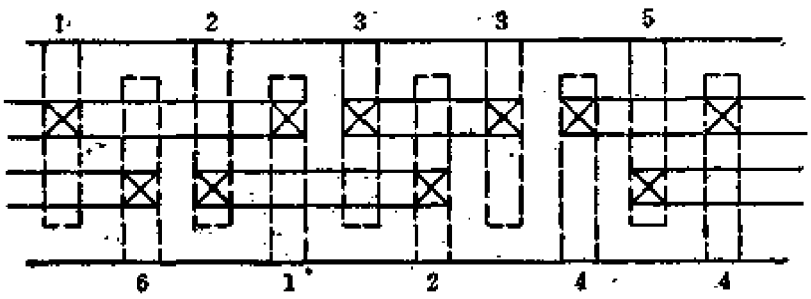


图 7.37　改变集成电路工艺规则的布线方法(如使多晶硅条足够细的情况)

虽然通道区布线算法在十几年中有了很大的发展，但是结合具体设计模式和具体工艺要求发展多目标优化（如除考虑所需水平通道数外，考虑所需通孔数、连线长度以及考虑到接触孔实际尺寸下的通道区面积的优化等）的算法仍有许多工作有待进一步的研究和发展。

除此之外，在通道区布线问题中，令人注目的另一个课题是三边通道区及四边通道区(stwitching box)问题。这是一个困难而引人入胜的问题。随着所应用的设计模式的复杂化，这个问题也正在引起人们更大的兴趣和关注。以往的算法[140][141]在这个问题上只取得有限的成功。近年 Burstein 等作了一些很好的工作[131]。他们的算法对一些简单的问题都取得了成功。图 7.38 是他们实验所用的一个四边通道区实例。[131]中的算法布线时除线网 24 外全部完成了连结。1984 年天津大学刘美轮、陈绍群和作者合作提出了一种新的四边通道区算法[123]。该算法在总体分析

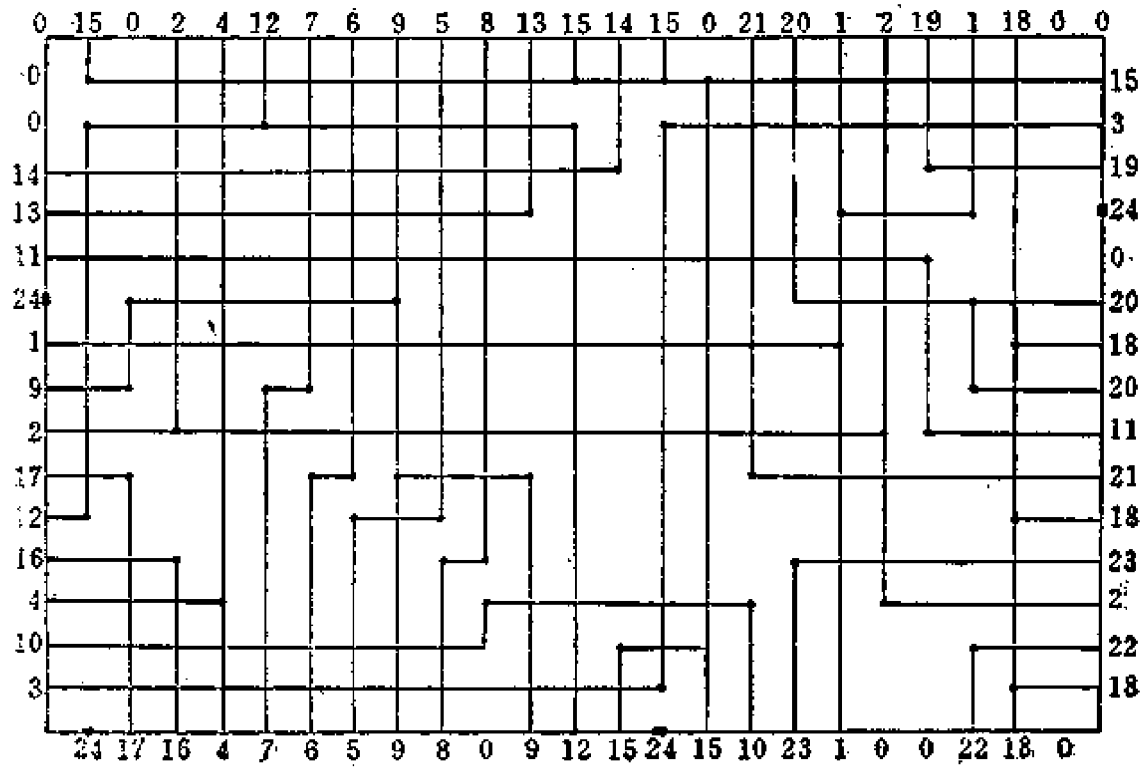


图 7.38 一个四边通道区布线实例

的基础上，通过对冲突线网的合理调整和同层邻行邻列布线，可100%地完成图7.38所示实例的布线(如图7.39所示)，并对已发表的其他四边通道区布线实例都取得了满意的结果(见图7.40)。

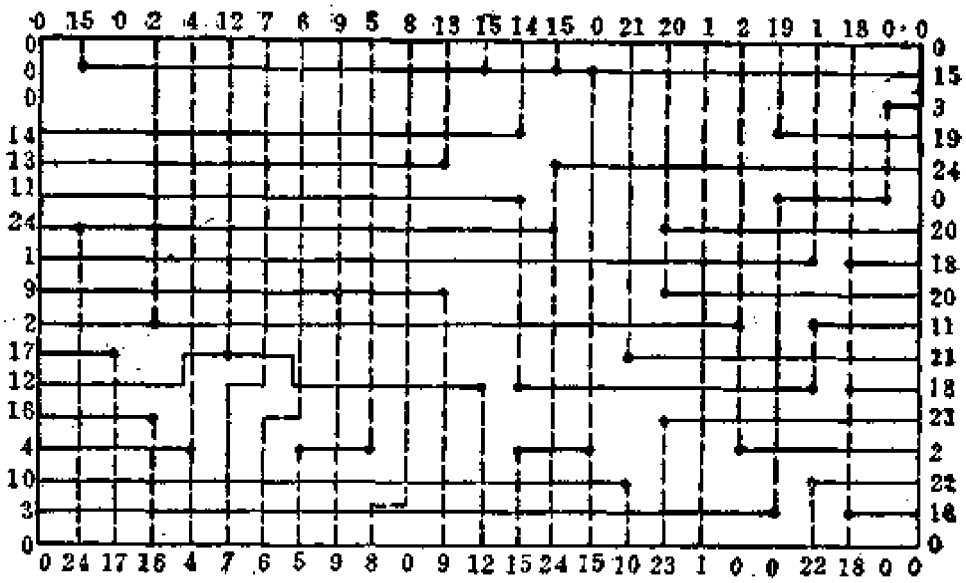


图7.39　与图7.38的对照图

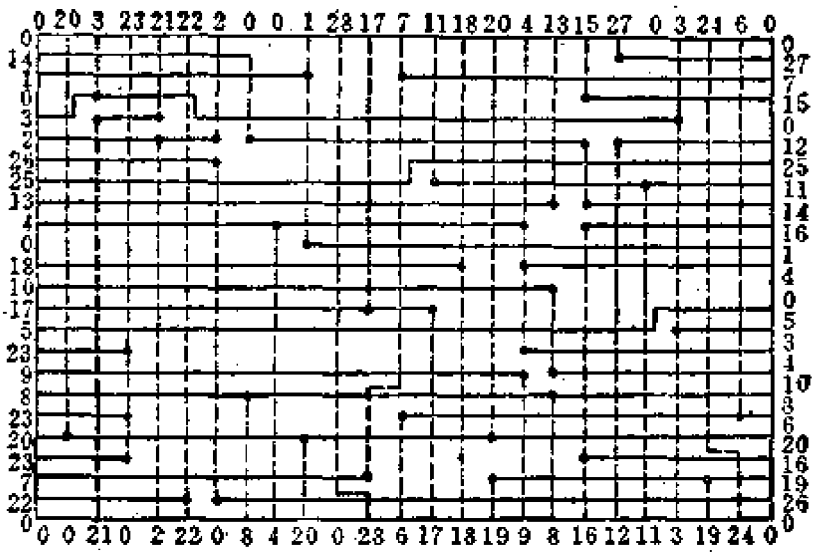


图7.40　复杂的switching box布线实例

第八章　数据库及其在设计自动化中的应用

8.1　数据库概述

一、什么是数据库

人类科学历史上，出现过三个最重要的基本概念：物质、能量和信息。数据就是一种重要的、人们在各种活动和交往中不可缺少的信息，特别在社会进入自动化之后，有关数据的收集、保存、描述、加工和传播等更是当前社会发展必不可少的环节。过去，由于计算工具比较简单，所能处理的数据多半是数值化的，即使后来出现了电子计算机，其应用也偏于科学计算方面，因此数值数据的贮存和处理相对比较简单。随着社会需要和计算机应用范围的不断扩展，数据已从纯粹的科学计算扩展到数据处理、信息检索、人工智能以及辅助设计等方面。这样一来，计算机所处理的信息就不只是数值化的，而且也有非数值化的了。不但如此，若从发展的趋势来看，后者将逐渐变为主要的。非数值化的数据在结构上比数值化的数据要复杂得多，其原因就在于除了数据本身之外，还有各部分数据之间的各种联系，如果对数据的组织方法仍采用传统的方式，那显然是不能适应的。另外，计算机软件的发展及网络系统的出现，往往需要在计算机系统中保存大量的各种用户所共享的数据，对此，也有必要进一步研究数据的组织形式、存贮形式和管理方式，因此，数据库理论和技术的研究就逐渐为人们所重视。

数据库(Data Base)[208]是一些相互有关的数据所组织成的集合，它的数据具有如下特征：

(1) 数据独立性

数据库中的数据应具有最大的独立性，比如，数据的存贮形式以及存于何类外存空间与应用程序是无关的。

(2) 小的冗余度

数据库中的数据应具有尽可能小的冗余度，随着要处理的数据量日益增多，应尽量避免不必要的重复数据的存贮，以便节省存贮空间。

(3) 数据共享性

数据库中的数据是由许多用户的数据积成的，一个用户只与数据库中的一小部分数据有关，不同用户的数据可能按照种种不同的方式互相重叠，其中某些独立的数据部分则可由许多用户共享。

(4) 交叉访问

数据库中各部分数据之间具有各式各样的联系，因此就存在一个交叉访问的问题，为此对这些联系的描述是一个十分重要的问题。

作为一个数据库，除了涉及具有上述特征的数据集合之外，还有它的管理系统。数据库管理系统 (Data Base Management System) 是指它的控制系统和设计数据库系统所需要的各种语言等。数据库控制系统是一组例行程序，它负责管理、控制和监督数据库以及应用程序对其所执行的各种操作，例如接收、插入、修改、删除等等。

数据库通常是一个具体应用领域中操作数据的集合，比如工厂的生产数据、银行的帐目数据、学校的人员和设备数据等。在 LSI/VLSI 计算机辅助设计系统中，一般由逻辑模拟、电路分析、测试码生成、布图设计、工艺模拟、掩膜制版等一些子系统组成。各子系统之间有着内在的联系，它们的数据也存在着某些共性和重叠。例如，电路的描述数据和一些其它数据，就是各子系统所共同需要的，若把各个子系统用到的所有数据以及由各子系统计算产生的数据加以组织，以一定的方式存贮在一起，就构成了一个 CAD

(Computer Aided Design)数据库。如果再配上对数据库进行统一控制的管理系统，便构成了一个 CAD 数据库系统。当然，对每个子系统来讲，也存在数据组织和管理的问题。比方说，一个布图设计系统也包含一些相对独立的部分(布局、布线、实体化等)，它可以单独建立库，也可以作为整个 CAD 数据库的一部分，但是不管怎样，CAD 数据库的建立必须在各子系统的数据组织和管理的基础上进行。图 8.1 就是一数据库系统简单视图。

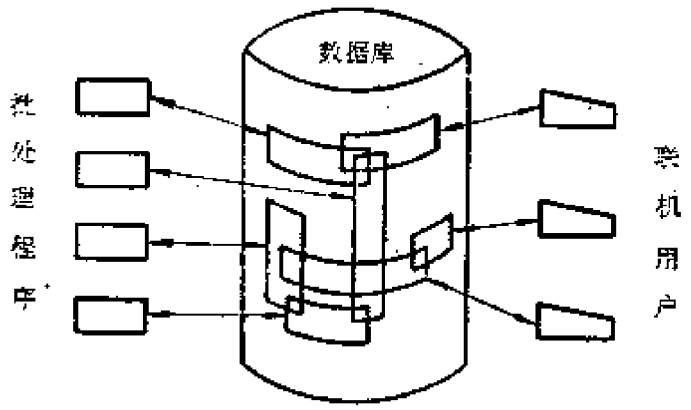


图 8.1 数据库系统示意图

二、CAD 系统与 CAD 数据库

正如上面所提到的，一个 CAD 系统由若干子系统按照一定的结构组成，但是这种结构随着人们研究的逐步深化也在不断变化和发展；反过来又将对其所涉及的数据结构提出新的要求。

1. 顺序结构系统

早期计算机辅助设计系统是顺序结构的。就是说，它的各子系统顺序地处理设计数据，这种处理是独立的。一个子系统要处理的数据来自它的前一个子系统，它处理后的数据则输出给后一个子系统，显然，这种数据管理的方式非常简单。系统中所涉及的数据基本上有两类：一类是可供用户共享的规则数据，这类数据一般在处理过程中是不变化的，如元件名、类型、几何尺寸、引线特性等；另一类是在设计过程中经常发生变化的设计数据，如逻辑描述、元件分配、联线表、测试码等。图 8.2 就是一个顺序结构的设

计自动化系统的一个简单例子。

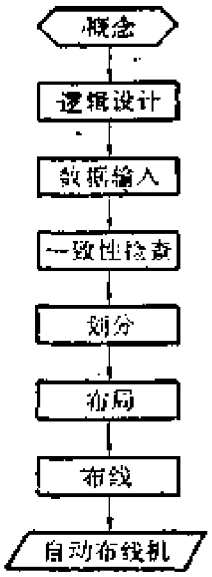


图 8.2 一个顺序结构的设计自动化系统

2. 并行结构系统

随着设计的数字系统愈来愈复杂，子系统也随之增加，数据规模相应地庞大起来，并且有些数据要供许多子系统共同使用。由于这种变化，必然引起数据管理上的复杂性，比如说数据如何组织；存取数据如何控制；数据的安全性、保密性又怎样加以保证等。基于上述种种原因，人们考虑从自动化设计的角度出发，着手建立公用的数据管理系统，试图通过它把 CAD 应用程序联系在一起，构成一个自动设计系统。这种系统的特点，就在于设计系统中的各部分不是以它们的设计次序相联系，而是以数据管理系统为中心联系各部分。这种管理系统要解决公用数据被多个程序访问的问题，以及数据的安全性、保密性、完整性和存取的可靠性。这种结构的系统又称并行结构系统。

3. 分级结构系统

后来由于 CAD 设计功能的扩展，系统相应的也复杂起来，因而产生了分级设计的思想。为了减少数据量的输入和存贮，系统逐渐采用了分层结构的形式。例如，对一个复杂的数字系统的设计而言，从逻辑设计的观点看，通常有以下几个步骤：

① 根据系统的概念和要求确定系统功能。

② 划分功能部件并确定相应的功能。

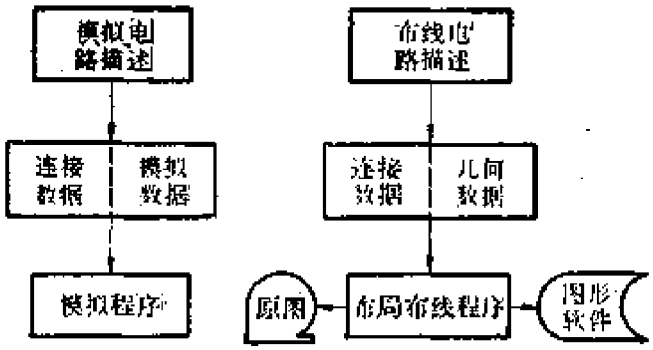
③ 根据需要再把功能部件划分成子功能部件并确定其相应子功能。以此类推，直到分成基本元件，最后形成详细逻辑。

若从工程设计的观点出发，其过程是相反的，即依据详细的逻辑，划分出插件并解决插件板布线，然后再划分底板并解决底板布线，最后由机架划分并产生机架接线表，从而形成系统。从自动化设计的角度看，这两者有密切联系，显然，设计中的每一步都有一

定的独立性，它所涉及到的数据也是相应一级的数据，而高一级的数据又可展成任意低级的数据。对应用程序来说，它要求哪一级的数据，系统就应该提供哪一级的数据。例如门级逻辑模拟程序，它需要对整个系统进行模拟，对此就应提供基本元件级的详细逻辑数据。

总之，为了组织好各种数据（其中包括设计数据、生产和维护所需的数据等），以便加以充分利用，必须按照设计要求把 CAD 各应用程序联在一起，构成一个自动设计系统。所以人们近年来逐渐认识到数据库在 CAD 系统中的作用，并着手建立 CAD 数据库。为了使读者更清楚起见，下面我们举一个例子来说明这个问题。

假定我们已经开发了图 8.3 所示的两个设计自动化应用系统：门级模拟系统和自动布局布线系统。



(a) 门级模拟系统　　(b) 自动布局布线系统

图 8.3　两个设计自动化应用系统结构

通过这两个系统的一般结构我们不难看出，每一个系统都有一套数据文件，而两个系统的数据文件中又有共同的部分，即“连接数据”。这里不仅反映出两个应用程序的数据文件有重复，而且还缺少统一的控制。事实上，这两个应用程序可以利用同样的连接数据。为此，可定义一个公用的数据文件和每个应用程序特定的数据文件，将这两个系统的数据统一控制起来（见图 8.4）。

图 8.4 所显示的系统结构，可以认为是一个简单的数据库，所有应用程序都在公共的数据文件上工作。显然，这种结构的缺点是

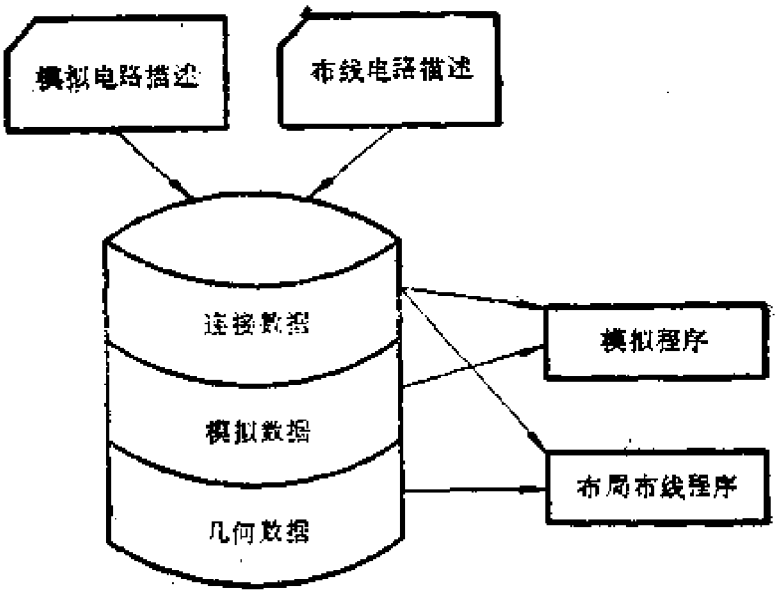


图 8.4　含有公用数据文件的两个应用系统结构

明显的，因为应用程序必须在知道数据库的整个数据结构的情况下，才能应用其中的数据。比如布局布线程序除了要知道连接数据之外，还要跳过模拟数据找到几何数据。另外，由于技术的更新和发展非常迅速，对数据库的修改和扩充是常常需要的，但是这种变动有可能影响到应用程序。所以最好的办法是把逻辑数据结构，物理数据结构与应用程序隔离开来。数据库管理系统就担负了这一任务。应用程序只向管理系统请求并获得所需要的数据，而数据库与应用程序的隔离则由管理系统通过“模式”和“子模式”来完成。图 8.5 所示出的就是一个带数据库管理系统的应用系统。

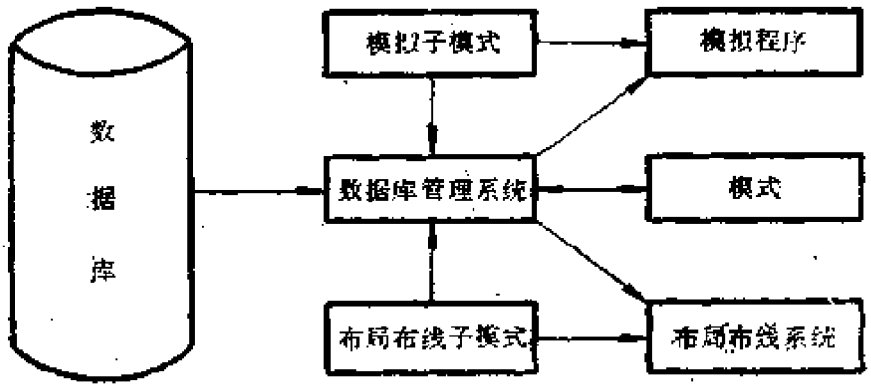


图 8.5　带有数据库管理系统的两个应用系统结构

在一个 LSI/VLSI 计算机辅助设计系统中，主要由以下子系统组成：逻辑模拟、逻辑图生成、电路分析、布图设计、测试码生成、工艺模拟、掩模制版等。由于 LSI/VLSI 的集成度高得惊人，不但该系统所涉及到的数据量极为庞大，而且数据之间还存在着复杂的内在联系，为了建立实用而有效的 CAD 系统，就必须对数据进行有效地组织和管理。目前，在各子系统不断地产生和完善的基础上，引入了数据库技术，建立以数据库为中心的计算机辅助设计系统。图 8.6 就是一个 CAD 系统的示意图[214]。

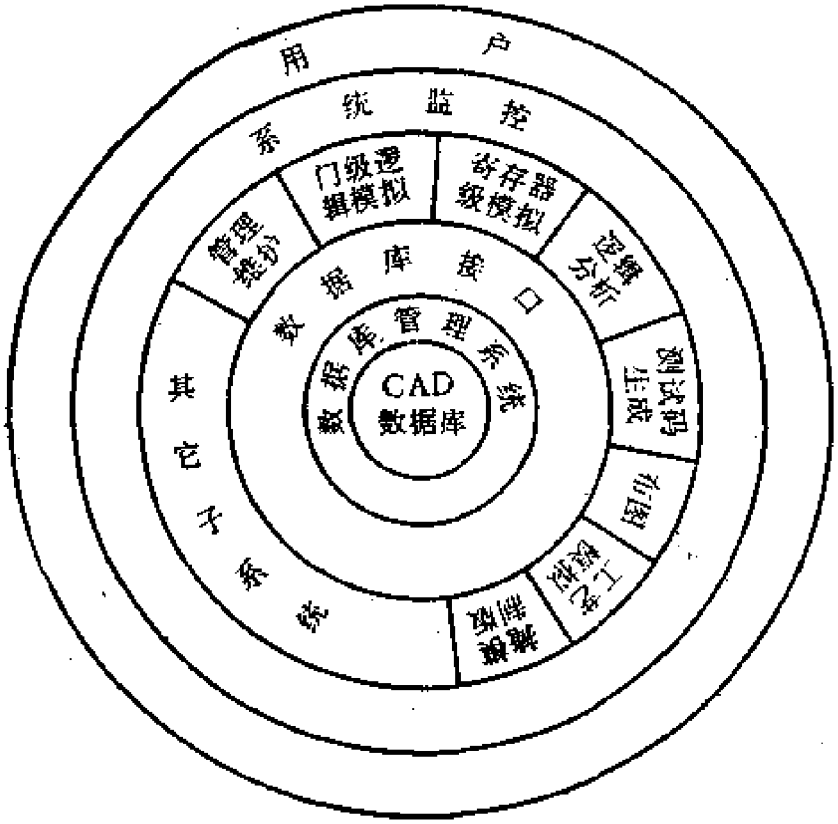


图 8.6 一个以数据库为中心的 CAD 系统

8.2 数据的组织和管理

一、数据的组织和管理

数据库系统是为满足日益发展的数据处理的需要，在文件系统的基础上发展起来的一种理想的数据管理技术[208]。初期的

文件系统具有以下几个特点：

① 用户数据文件主要存贮在磁带上，因而它的组织方式是顺序的。

② 文件在外存的物理结构与用户观点的逻辑结构完全一致。

③ 数据管理软件属于操作系统，只用来完成输出输入操作。

这种文件系统的缺点是：

① 这种组织形式只能应付批处理，而不能适应实时处理。

② 文件是用户建立的，用户既要编写应用程序，又要设计数据的物理安排。

③ 文件不易共享、数据冗余度很高。

④ 文件的物理结构发生变化时将影响应用程序。

后来由于使用磁鼓，尤其是使用磁盘作为联机的外存设备之后，使数据的组织和管理发生了很大变化：

① 对用户观点的数据进行严格细致的描述，使文件、记录、数据项等数据单位之间的联系清晰、结构简单。

② 允许用户以记录或数据项作单位进行访问，也允许多关键字检索和文件之间的交叉访问。

③ 数据的物理存贮可以很复杂，同样的物理数据可以导出多个不同的逻辑文件。用户可以简单的逻辑处理数据而无需考虑数据的存贮情况；改动数据的物理位置和存贮结构不必修改或重写应用程序。用户的逻辑数据与它们的物理存贮之间的转换由数据管理软件完成。

④ 目前的数据库系统，又在用户数据逻辑结构与物理存贮结构之间加入了数据的整体逻辑结构，使得数据的物理存贮结构的变化尽量不影响数据的整体逻辑结构和用户的应用程序；数据整体逻辑结构的改变也尽量不影响用户的应用程序。

二、数据库的设计要求

设计数据库[209][211]虽然要考虑用户的要求以及有关的经

济和技术条件，但是有些基本原则应该遵守：

1．保证数据独立性

数据库内的数据应保持相对的独立性，以便不同用户使用库内数据时能随时调用，以达到数据共享。

2．减少数据冗余

减少数据库中数据冗余，为的是节约存贮空间，消除潜在的不一致性危险。例如，本来认为两个文件中的某一记录是相同的，但是，当其中一文件的数据修改后，而另一文件没作相应的修改，因此，应该尽量减少数据冗余，提高共享程度。

3．用户与系统的接口要尽量简单

用户与系统接口简单，用户容易掌握。

4．确保数据库系统的可靠、安全和完整

一个系统的可靠性体现在它的软、硬件运行可靠，出了故障可以快速地恢复。所谓安全是指系统对数据的保护能力。完整性是保证数据库仅仅包含正确数据，消除那些由错误操作和错误推导所产生的不正确数据。

5．重新组织数据的能力

数据库中的数据访问频率不是固定不变的。通常把用户访问频率高的数据放在快速访问设备上（如磁盘），而把很少访问的数据放在慢速访问设备上（如磁带）。另外，数据库经过一段时间的运行之后，由于不断地进行了插入、删除操作，使原有的物理文件变得很乱，为了适应访问频率的变化，改善数据组织的零乱和时空性能差的状况，应该及时有效地改变文件的结构和物理布局。关于这一点，一般都是由系统周期性的处理来执行这一任务。

6．系统的可修改性和可扩充性

通常，一个数据库的建立都是逐渐的而不是一次性完成的，不但操作数据会不断增加和扩充，而且它的用户和应用情况也会发生变化，因此在数据库设计时就要预先考虑与未来应用程序的接口问题。使得既不因为后来的某些变化影响整个数据库的正常工

作(或改变设计方案)，又不因为某些修改或扩充影响原有用户的使用方式。所以在数据库的结构和组织技术上应该考虑到修改和扩充的容易性。

7．对数据间的内在联系有较强的描述手段

因为数据库中的数据是反映客观事物及其间的联系的，所以数据库系统必须有能力描述反映客观事物及其联系的复杂的逻辑结构。

三、数据的描述方法

数据库的主要研究对象是数据，因此首要的任务就是如何描述数据以及介绍相应的描述术语[212]。归纳起来，对数据有三种描述方法：抽象的、逻辑的和物理的。

1．数据的抽象描述

数据所表示的对象非常广泛，它可以代表现实世界中的任何事物。事物就是研究的对象，每种事物都有自己的特征。当然，事物又有“共同事物”和“特殊事物”之分。事物及其之间的联系就构成了“现实世界”(见图 8.7(*a*))。为了更好地认识和表现现实世界中事物及其间的联系，有必要把它们抽象成观念性的东西，图 8.7(*b*) 就显示出了与图 8.7(*a*)所对应的各种观念。

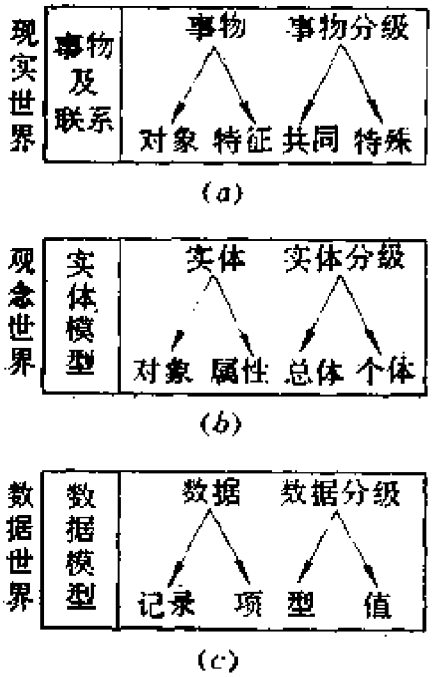


图 8.7　数据的三种描述关系

(1) 对象与属性

在观念世界中，是用“实体”来描述客观事物的。实体又分成“对象”和“属性”，它们与事物的“对象”和“特征”相对应。例如：“人”是研究的对象，那么“姓名”、“年令”、“性别”就表示了人的三个方面的特征。一个对象具有某些属性，而其中的某

一属性往往又可能是另一些属性描述的对象，比如对“工资”这个对象来说，它具有属性：应发工资、扣除金额、实发金额等，而扣除金额又是属性互助金、房租、水电费等所描述的对象。因此属性又可分成二种：不能再细分的属性叫原子属性，如性别、颜色等；可以细分的属性称为可分属性，比方说“出生”就可以分为“年”、“月”、“日”。当然这些观念均是相对的，它与描述的事物及观察研究问题的角度都有关系。例如描述职工的一般情况时，只要知道每个职工的工资多少即可，无需再加细分；而在描述财务部门发放工资的情况时就非得分细不可。

(2) 总体和个体

实体也分两级：“总体”和“个体”，分别对应着“共同事物”和“特殊事物。所谓“个体”是指单个的能互有区别的特定实体，如“张三”、“交大”；另一种是泛指由某一些个体组成的集合，这样的实体又称为“总体”，如“人”、“学校”。

显然，对象与属性的联系是对象内部的联系，而个体与总体的联系是对象的外部联系。当然，这种“内”“外”的概念与考虑问题的范围有关，如在教育界的范围内“大学”、“中学”、“小学”都可以看成是总体概念；若在工、农、兵、学、商这个大范围内考虑问题，“学校”就是一个总体概念，而“大学”、“中学”、“小学”就是个体概念了。另外，一个总体所包含的个体数目及包括哪些个体，完全取决于处理问题时的需要。

(3) 总体间的联系

数据库与一般数据组织的重要区别就在于：它不仅仅只是一个一般的数据集合，而且还要体现出数据间的关系。为此应该分析事物间各种各样的联系。若从总体之间联系的角度来看，两个总体 A、B（设两者均含有若干个体）之间的联系方式不外乎有以下三种：

① 一对一联系。若 A 中任一个体至多对应 B 中一个个体（反之亦然），则称 A 对 B 是一对一联系。如“班级”与“正班长”之间，见

图 8.8(a) 所示。

② 一对多联系。若A中至少有一个体对应B中一个以上个体；而B中任一个体至多对应A中一个个体，则称A对B为一对多联系。比如"班级"对"学生"就属这种联系，见图 8.8(b) 所示。

③ 多对多联系。如果A中至少有一个体对应B中一个以上个体(反之亦然)，则称A对B为多对多联系。如"学生"与"课程"之间，见图 8.8(c) 所示。

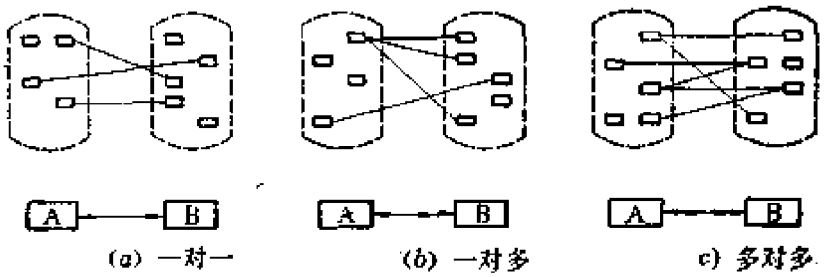


图 8.8 两个总体间的联系方式

原则上说，许多总体之间的复杂联系，都可以用若干组上述联系等价地表示。

(4) 实体模型

实体模型是确定数据库所含信息内容的关键，因此设计者必须与用户合作，开列用户问题表，继而对现实世界中所要考虑的客观事物及其联系进行模拟，以便建立一个正确反映客观事实的实体模型。在模型中，实体要逐一命名以资区别。实体间的联系要加以描述，比如以教学情况为例，通常要反映的事实有学生、教员、课程、学习、任课等几方面的情况。作为研究对象，它们又各自具有体现本身特点的属性：

学生：学号、姓名、性别

教员：工作证号、姓名、职称、专长

课程：课程号、课程名、教学时数

学习：学号、课程号、分数

任课：课程号、任课教员(可用工作证号)

其中“学习”实际上是学生与课程间的一种联系，“任课”是教员与课程之间的一种联系。在这里均被视作研究对象。学生对课程是多对多联系；而教员对课程是一对多联系，即一个教员可以讲授多门课程，但一门课程至多只能由一个教员任教(若同一门课程分别由几个不同教员开课时，就看作几门课程，课程名相同，但可以取不同的课程号)。图 8.9 就表示了教学情况的静态实体模型。这种模型只反映实体的当前状态，而不反映实体状态的变化过程。

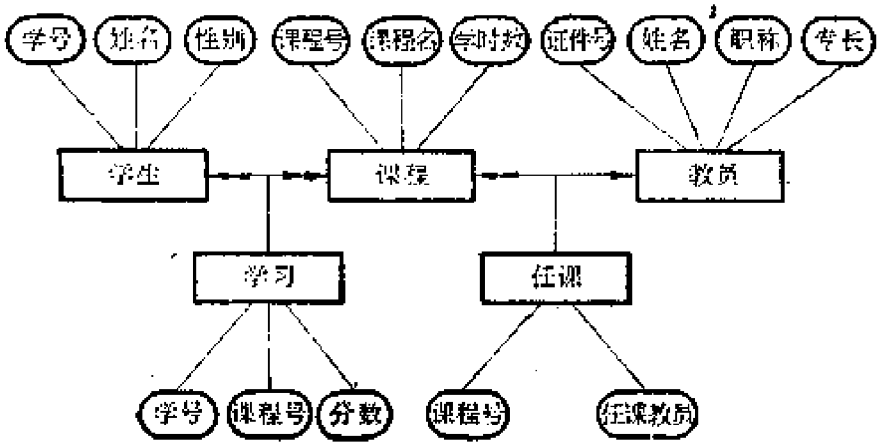


图 8.9　一个静态实体模型

2. 数据的逻辑描述

以上已把现实世界中的事物及其间的联系进行了抽象的描述，也就是说研究对象的一种抽象表示。但是从用户观点来看，要研究的应该是逻辑描述。与抽象描述的各种概念术语相应的逻辑描述的各种术语如图 8.7(c) 所示：

(1) 记录与项

实体有对象与属性之分，相应逻辑数据就是“记录”和“项”。一个对象具有若干属性，一个记录也有若干项。相应的原子属性和可分属性对应的就是基本项和组合项。基本项是最小逻辑数据单位；而组合项则由基本项与组合项构成。

(2) 型与值

实体分总体和个体两级，而逻辑数据也相应地分成“型”和“值”。“型”是个总体概念，相当于一个框架或一个表格形式。如教员记录的类型可显示成图 8.10 的样子；而“值”是个具体事物，一旦在这整体形式下填入具体内容（个体）时，如 “01941 张三 1930 1 20 男” 就说该记录类型有了值。

证件号	姓名	出生			姓别
		年	月	日	
01941	张三	1939	1	20	男

图 8.10 类型和值的关系示意图

当然，对数据项来说也有型和值之分。不仅如此，一个数据项的值又可以是另一个数据项的型，如数据项“学校”的值是“大学”，而“大学”又可以看作是数据项“大学”的型，它的值是“交通大学”。另外，有“值”的概念，就应该考虑值域的问题。如果两个记录的所有数据项的值均相同，才称两记录是相同的。

因为并非一个记录类型只有一个值，所以又把记录型与值的总和定义为“文件”。一个记录类型和它的一些当前记录组成“同质文件”；不同记录类型和它们的当前记录组成“异质文件”。特别地把其值能唯一标识记录的一个或多个数据项称为记录类型（或文件）的关键字（key），而用于组织文件的关键字称为主关键字（Primary key），主要通过它来区分和访问记录。图 8.11 显示了抽象描述和逻辑描述各术语间的对应关系。

(3) 数据模型

数据模型是与实体模型相对应的数据之间的一个整体逻辑结构。由于对象与属性、总体与个体区分的相对性，因此这种结构通常是相当复杂的。为使模型能清晰、准确地反映客观事物，在设计时应作到：

① 给数据模型命名；

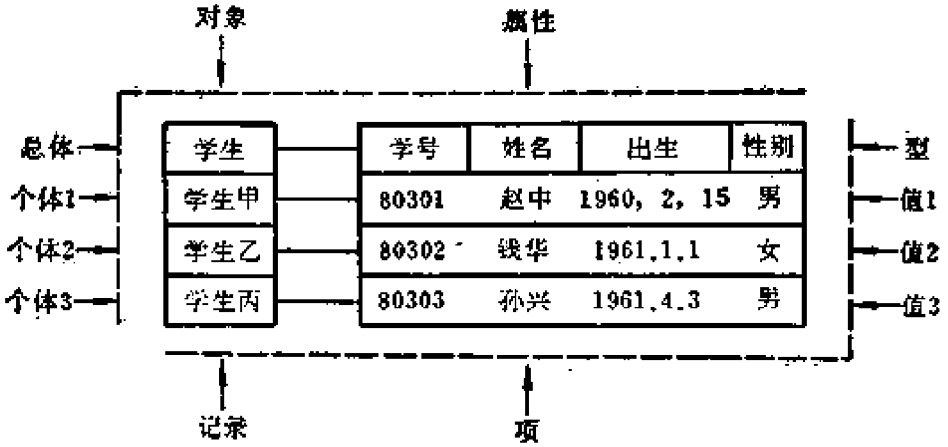


图 8.11　抽象描述和逻辑描述间术语的对应关系

② 给每个记录类型命名；

③ 给每个数据项命名，并确定作记录类型主关键字的项；

④ 说明各记录类型间的联系，必要时给这种联系加以命名；

⑤ 必要时指出各数据项的类型、长度、值域等。

整个数据模型构成一个大的框架，一旦填入具体的数据值就得到模型的一个实例。显然，数据模型的好坏将关系到数据库的性能，因而它的设计方法也将决定着数据库的设计方法。目前常用的模型有三种：层次模型(hierarchical model)、网络模型(network model)、关系模型(relational model)。详细说明见下一节。

3. 数据的物理描述

不管数据代表何类事物，作为研究对象最终都要存放在计算机存贮设备中。计算机的存贮设备一般分内存和外存，中间由通道隔开。外存较内存访问时间长、使用频率低，但外存贮容量大。数据库系统的数据主要在外存中。由此看来，还需要用适应计算机存贮器特征的术语对数据进行物理描述。

(1) 存贮介质

外存贮器是数据库用以保存数据的物质基础，它的一些特性和指标是进行数据库物理描述时必须考虑的关键因素。可以作为数据库外存介质的部件有磁带、磁鼓、磁盘等，它们都是由一个单独的读/写机构进行工作的独立的外存部件。它们的存贮空间是

外存分配的独立部分，一个独立的外存部件又称之为“卷宗”。磁盘是目前主要的外存设备，下面以它为例加以简单介绍。

计算机中用的磁盘分硬盘和软盘两种。硬盘容量大，通常在一百万字节以上；软盘容量小，一般在二万到一百万字节范围内。信息记录在盘面上。每个盘面由一组同心圆组成，每个圆称为一个磁道，信息就记录其上。盘面一般又被分成若干相等的扇形，这些扇形又将每条磁道分割成许多被称作“物理记录”的相等段。虽然同一扇形区中不同磁道被分割成的段的长度不等，但它们记录的信息量是相等的。若计算机存贮信息是以物理记录为单位，则段中还必须有相应的地址信息。图 8.12 是一个被分成 16 个扇形的分布图，用以建立数据库的磁盘通常指硬盘。这种磁盘往往不是一个盘片，而是一组盘片，而且每片均可两面记录信息。各盘面上磁道数目相同。盘面自上而下进行编号，称为面号（一个盘面两面均作记录的算作两面）。一个磁盘组所有盘面上同一磁道组成一个柱面，信息在同一柱面上连续记录，柱面由外向里编柱号。因此，要对磁盘进行信息读、写，就应该给出如下参数：柱面号、盘面号、段号、读写长度。

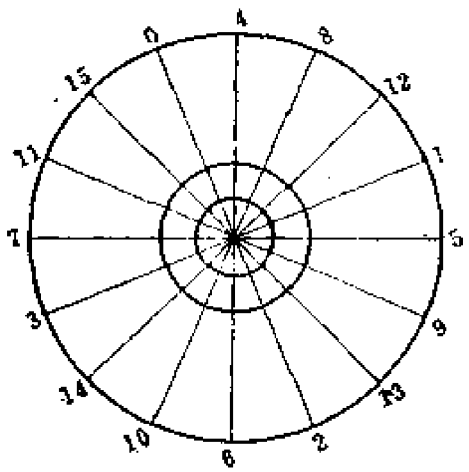


图 8.12　磁盘面的扇形分布图

（2）物理描述术语

计算机内存（或称主存贮器）存放数据的基本单位是字节或字；外存存放数据的基本单位一般是一个物理记录，它也是内外一次交换信息的单位。物理记录与逻辑记录可以等长，也可以不等长。例如，磁盘的物理记录是等长的，它可以存放一个逻辑记录，也可以存放几个逻辑记录，甚至还可以几个物理记录同时存放一个逻辑记录。

在虚拟存贮系统中，通常引进“页面”为物理单位的存放方式。此时内存要划分出位置固定而大小相等的的若干块，每一块称为一个页面，以便实现内外存的调度。

一个磁盘称为一个卷宗，可存放一个文件或几个文件。每个文件占用几条磁道或几个柱面，这要视文件的信息量而定。

数据的物理描述是计算机可以接受的，而用户则不易接受；数据的逻辑描述是用户可以接受的，而计算机则不能直接接受。因此，它们之间必须通过某种转换。

总之，了解了数据的上述三种描述，对数据库的设计大有好处。说到底，数据库无非就是“异质总体”所对应的记录的集合而已。表 8.1 是一种描述术语(或单位)间的对应关系。

表 8.1　三种描述术语的对应关系

抽象描述	逻辑描述	物　理　描　述
属　性	基 本 项	用字节或字描述基本项
可分属性	组 合 项	用字节或字描述组合项
个　体	逻辑记录	物理记录
同质总体	文　件	存文件于若干磁道、柱面或卷宗
异质总体	数 据 库	存数据库于若干柱面或卷宗

8.3　数据库的逻辑结构和设计方法

一、模式和子模式

设计数据库不仅为了存放数据，而且主要是为实现数据独立性、减少冗余度以及描述数据之间的自然关系。因此，从设计者的角度来看，总希望结合外存贮器的特点，把数据组织得除适应上述要求之外，还便于检索、定位、紧缩、修改、扩充等。但是这一切对用户来讲并不太感兴趣，他们感兴趣的是数据表示法的方便性、灵活性和简单性，并且不受物理存贮变化的影响。因此，在设计数据库管理软件时，要使数据在外存的物理组织与用户观点的逻辑组

织无直接关系；而它们之间的间接对应关系则由管理软件借助某种变换来完成。为此引进了“模式”(schema)这个概念，企图对用户观点的逻辑数据采用一种形式化方式。

1．模式

所谓模式就是对数据库中全部数据类型和记录类型的整体逻辑描述。事实上，模式的主体就是数据库的数据模型。数据库管理系统视模式为框架，并将用户各类数据的值按照该框架进行存贮和访问。填入该框架内的值可能不断变化，但模式本身不变。比如规定描述学生成绩的模式为：

“学号　姓名　性别　外语　语文　数学
　物理　化学　生物　历史　地理　政治”

而“00003　张明　男　90　88　95　90　75　76　83　91　87”则为填入该模式的一个具体值，即关于学生张明的一个记录值。这就是说，设计者规定了这种描述之后，用户就按照它的规定格式访问和操作数据。

2．子模式

随着系统的日益庞大，模式也将变得复杂起来。但是对于一个具体的用户，通常并不需要使用全局逻辑结构，而是只要其中的一部分(比如理工型的学校对诸如生物、历史、地理之类的数据项就不要求)，为此又引进了“子模式”(subschema)的概念。子模式是模式的一个逻辑子集，若干子模式有机的结合即为模式。子模式是面对用户的，它与模式的关系可由系统映射来完成。如“学号　姓名　性别　外语　语文　数学”，就是一个子模式。

二、数据库的逻辑结构

由前面的叙述可知，子模式是对用户的，而模式是整个数据库的数据模型。子模式与模式之间存在着独立性，即子模式的改变不影响模式。子模式到模式通过映射变换。另外，设计者还要设计数据在外存的物理结构，而模式到数据的物理结构也要通过映

射变换，它们之间也存在独立性。各级数据描述的结构关系可从图 8.13 看出。

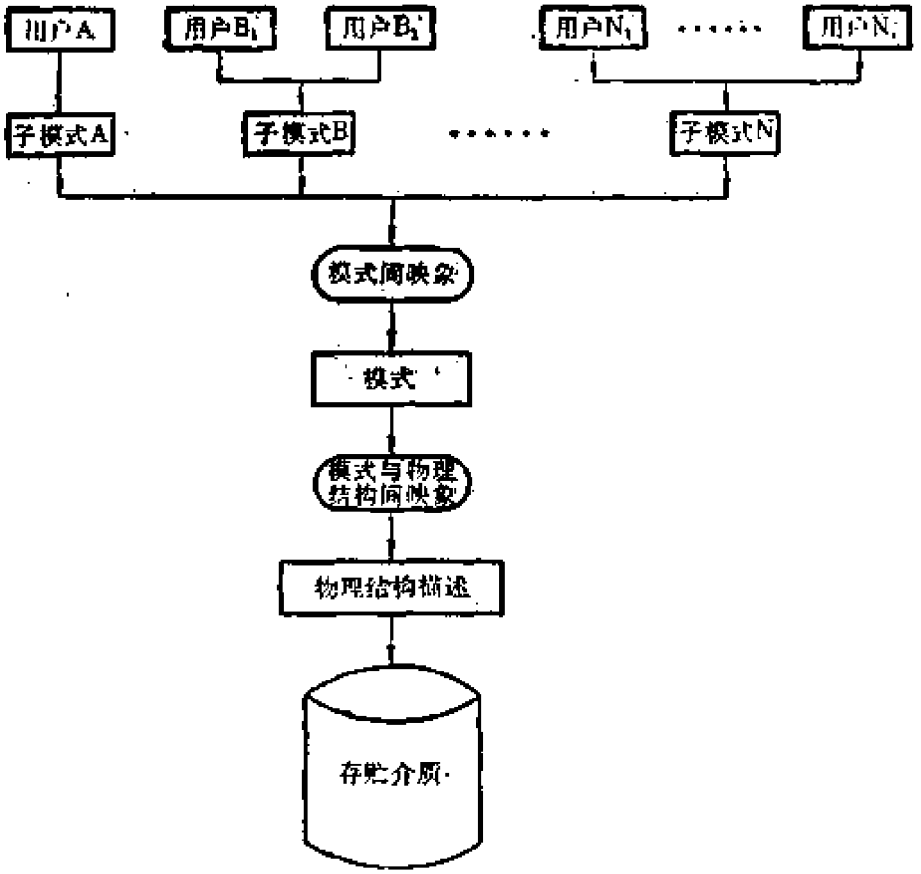


图 8.13 各级描述间的结构关系

从图中不难看出，数据库是个分级结构：

(1) 外模式

用户看得到并获准使用的那部分数据的逻辑结构，它根据系统给定的子模式，用询问语言或应用程序去操作库中的数据。

(2) 概念模式

它是对数据库的整体逻辑描述。它是数据库管理员看得到的数据库。它将把用户的数据结构有机地结合成一个逻辑整体。

(3) 内模式

又称存贮模式，它是物理级数据库。它包含着数据库的全部存贮数据。它是真正存在的数据库。

概念模式不过是内模式的一种抽象描述；而外模式则是用户与数据库的接口。三者间的转换由数据库管理系统来完成，它把用户对数据库的操作转化到物理级去执行。图 8.14 给出了数据库系统结构的简图，显示了数据库管理系统用以调度各部分之间的活动。用户的应用程序一般用高级语言编写，如FORTRAN,COBOL,BASIC 等。访问数据库要使用系统提供的数据操作语言，使用时要把它嵌套在用户使用的语言中。数据库管理系统按照操作语言所指的数据名字和类型等去查找相应的子模式和模式，再利用数据的物理结构描述去寻出数据在外存的物理位置，然后请求操作系统读出或写入相应的数据。若是读取数据，则将读出的数据先送

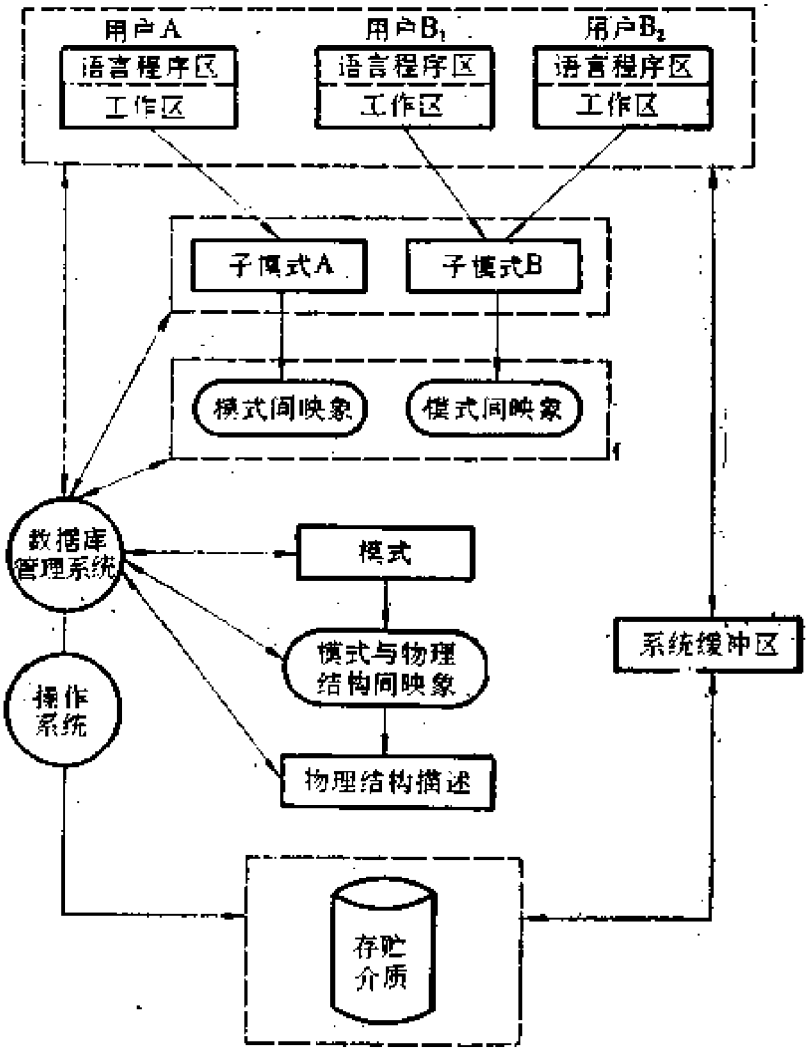


图 8.14　数据库系统结构

进缓冲区，再经过各模式之间的转换最终送到用户工作区。

三、数据库的设计方法

数据库的设计[211]可分两个阶段：逻辑组织设计和物理组织设计。前者是指确定整个数据库的逻辑结构；后者则是数据库的实际存取的实现方式，即是决定文件的结构和存取方法、确定数据的存放位置、选取索引用的关键字以及选定存取路径等。数据库是建立 CAD 系统的核心问题。而数据库的逻辑组织又是建立数据库的关键问题之一，因为它将影响到系统的每一部分。逻辑组织设计的中心问题则是数据模式和子模式的建立，它的类型决定着数据库的设计方法。目前数据库的数据模式有三种类型，即前面提到的层次模式、网络模式和关系模式，从而就导致了数据库的三种设计方法。现分别介绍如下。

1．层次设计方法

(1) 设计方法的特点

层次设计方法是用树形图描述数据逻辑结构关系的方法。就是说把数据模型归结为一棵有根定向树。树的结点表示数据类型；它的分级关系表示数据的层次关系；它的各条路径则描述记录之间的联系。对数据的访问借助路径来实现。树的主要特征是除根结点之外，任何结点只有一个父结(父记录)；一个父结对应多个子结，即父结表示的总体与子结表示的总体是一对多的联系，如图 8.15 所示。

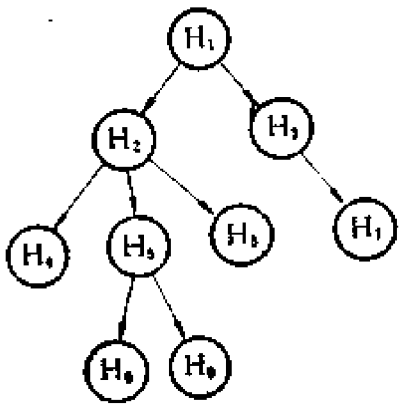


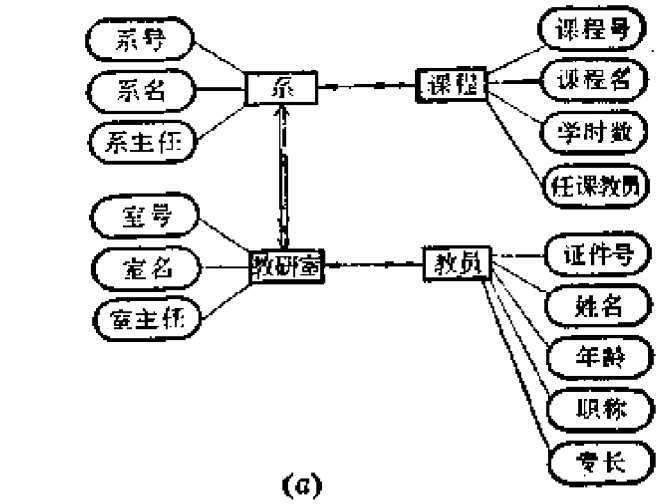
图 8.15 层次结构图

例如要建立反映“系、教研室、课程、教员”情况的数据库。它的实体模型见图 8.16(*a*)。其对象之间的联系是：每个系下设若干教研室，开设若干门课程；每个教研室又有若干名教员。根据实体模型可构造如图 8.16(*b*) 的层次数

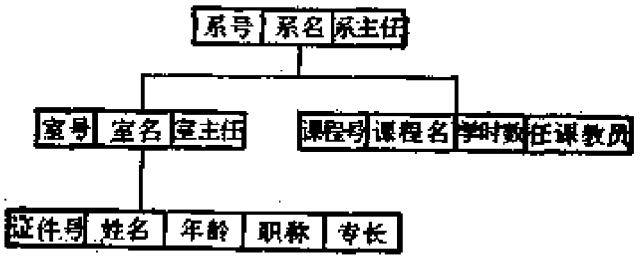
据模型。图 8.16(c) 是以计算机系为实例的值，同样的办法还可构造和填写其它系的情况。

(2) 查询方法

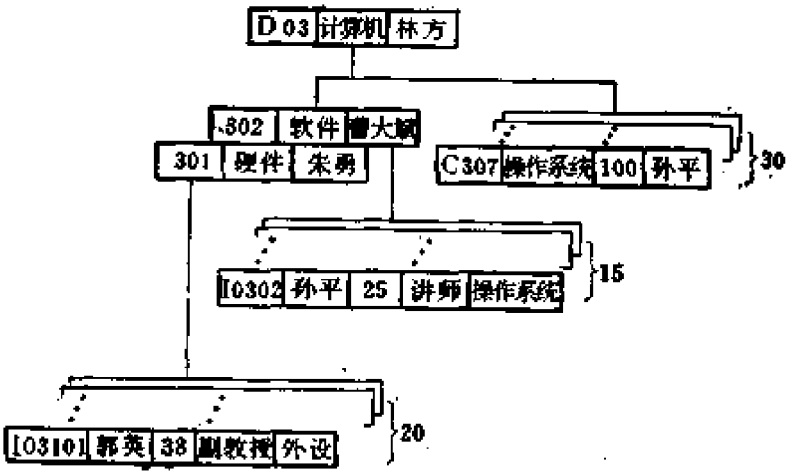
通过该模型可进行如下查询：



(a)



(b)



(c)

图 8.16 一个层次结构实例

① 某系的情况；

② 某系教研室和课程情况；

③ 某系教研室教员的情况等。

显然，这种模型必须按照从根开始的某条路径提出询问，否则就不能直接回答。例如 D03 系软件教研室教员孙平的年龄、职称等可直接得到回答；而 D03 系计算机原理课任课教员的年龄、职称就不能直接回答。因为在课程与教员之间没有路径相通。此时就需要分成两步来查询。由于这种模型中的路径一经规定就不能改变，所以在设计时路径的确定要特别慎重，否则将影响查询效率。

2. 网络设计方法

(1) 设计方法特点

层次设计方法反映了事物间“一对一”和“一对多”的关系。从树型图的观点看，树根只是一个结点而无双亲；其余结点有且仅有一个双亲。根据这一特点，在层次设计方法中记录之间的联系不必另行描述。然而，对于“多对多”的对应关系，层次设计方法处理起来就有困难。为此在层次设计方法的基础上发展了网络设计方法。它可以处理上述三种对应关系。其特点是网中可有一个以上的结点无双亲；并且至少存在一个结点有一个以上的双亲。这样在网中就多出了“枝”之间的交叉联系。对这个新的复杂关系，必须另外寻找新的描述手段加以描述。“系”(set) 这个概念就是为此而引出的。

除此之外，按层次设计方法，通常都是把属于同一双亲记录的若干子系记录均放在相应双亲记录之中。由于不同双亲所含子系记录的多少不一致，因而使双亲记录的长度变为不定长，这必然增加操作的复杂性。如果引用“系类型”就可以较好的解决这一难题。

(2) 系和系类型

“系”表示记录之间的联系，“系类型”是对两个以上记录类型

之间某种联系的描述。在此联系中，有一个记录类型处于主导地位，称为主记录类型，其余的就是处于从属地位的从记录类型。系类型也象其它记录类型一样，是对同样结构的一类记录的描述。它不仅应加以命名，而且有值，其值就是一组不同的具体的系记录。如果在一个系记录中无从记录与主记录相联系，则称该系记录为空的。系记录实际上是以记录为结点的一棵二级树，它的根是所在系类型中主记录类型的记录，其叶是所在系类型中从记录类型的记录。主记录类型中有多少个记录，它所在的系类型中也有多少个系记录。例如教员与课程之间构成的系类型如图 8.17 所示。

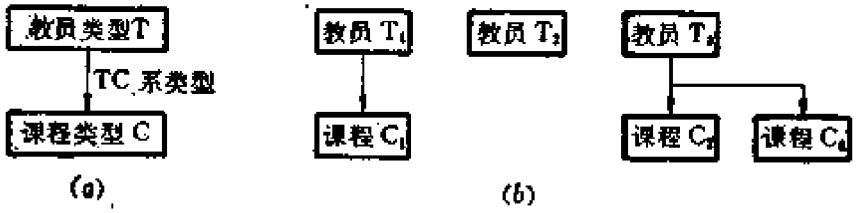


图 8.17　系类型示意图

因为在网络模型中，子结到父结的联系不唯一，因此就无法只依照父结来描述它们之间的联系，而是应该对每对联系都要加以定义。图 8.17(a) 中的 TC 就是对教师类型到课程类型之间联系的系类型的命名，图 8.17(b)中显示 TC 系类型的三个系记录，其中有一个为空。

在网络模型中，一个记录类型可以是一个系类型的主记录类型，同时又是另一个系类型的从记录类型(图 8.18(a) 中 B 类型)；一个记录类型可以是若干个系类型的从记录类型 (图 8.18(b) 中 C 类型)；环形结构也是允许的(图 8.18(c))；在同样记录类型之间定义两个系类型也是可以的(见图8.18(d))。例如在工人类型到设备类型之间可定义“使用”和“保养”两个系。有一点是必须指出的，一个记录仅仅可以联系到一个给定系类型的系上，而不能联系到它的几个系上，如设备 A 只能由某一工人使用，而不能同时被几个工人使用；但一个记录却可以联系到不同系类型的几个系上，如

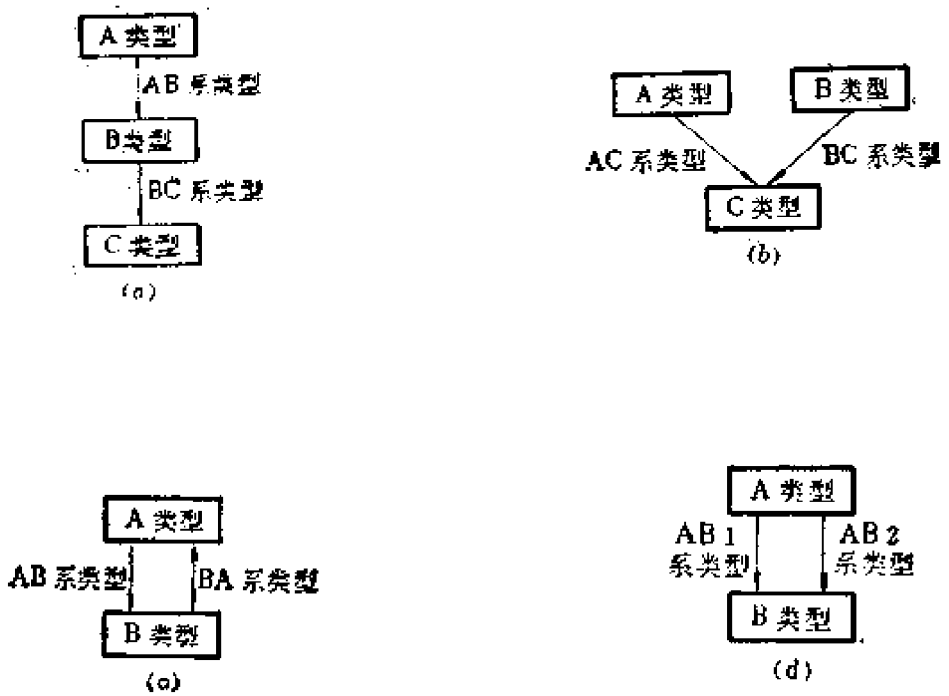


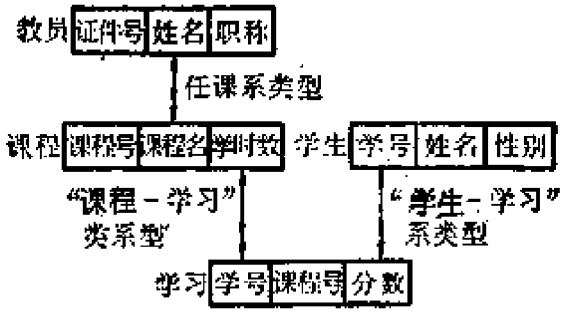
图 8.18　主、从记录关系图

设备A由工人甲使用而由工人丙保养。

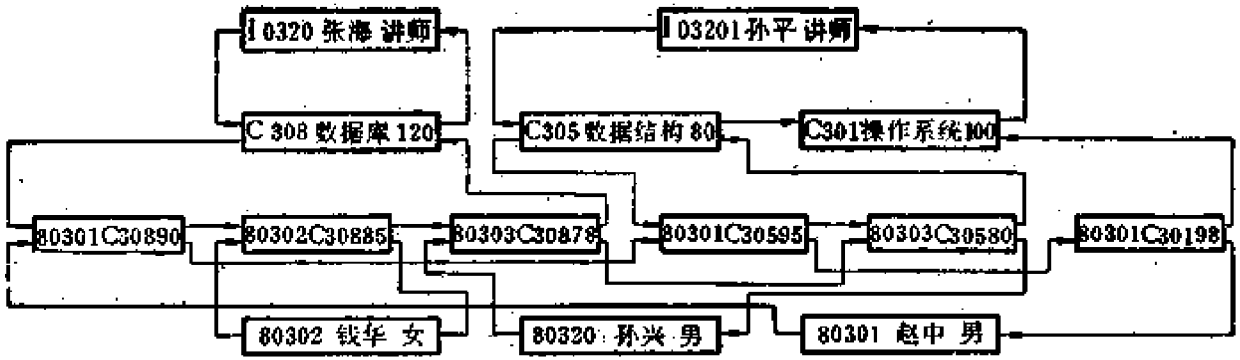
(3) 网络模型的建立

从以上的介绍不难看出，由于数据库可以包含任意个数的记录类型和系类型，因此就可以利用各种不同系的组合来表示任意复杂的网络模型。图 8.19 给出了一个网络模型及实例。

若两个记录类型之间是多对多的关系时，就用一个新的记录类型来联接它们。这个新记录类型的数据项由两个记录类型中标识记录的数据项串连组成。例如：设有学生若干人(假定A和B两人)和选修课若干门(假定W、X、Y、Z四门)，一个学生可选多门课，一门课也可被多个学生选修。那么这种关系网的存贮形式可表示成图 8.20。



(a)



(b)

图 8.19 一个网络模型结构实例

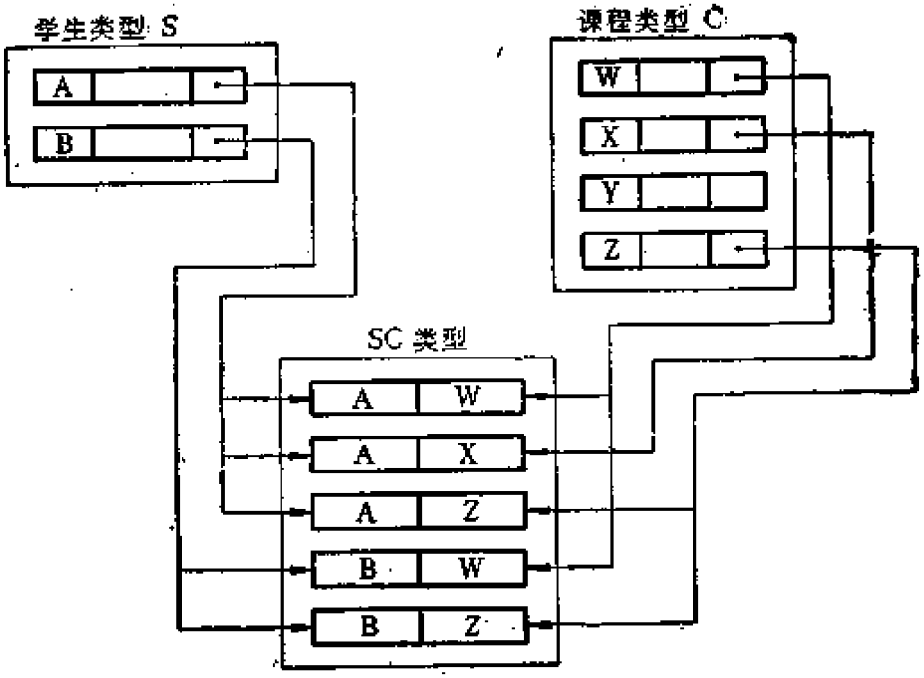
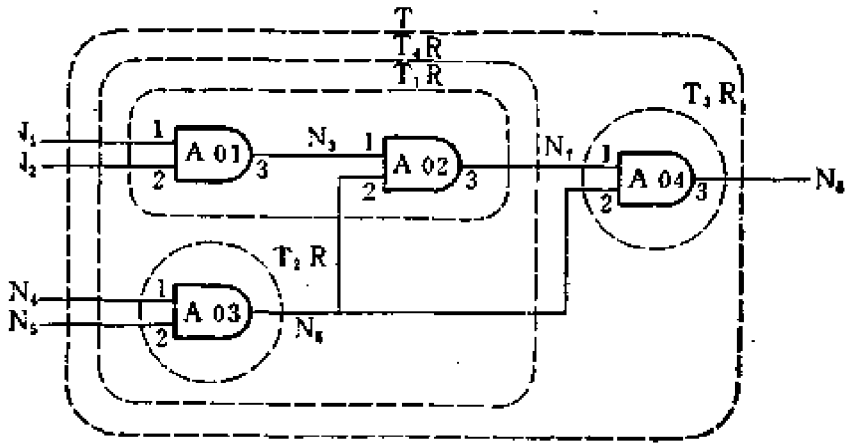


图 8.20　多对多关系的联接示意图

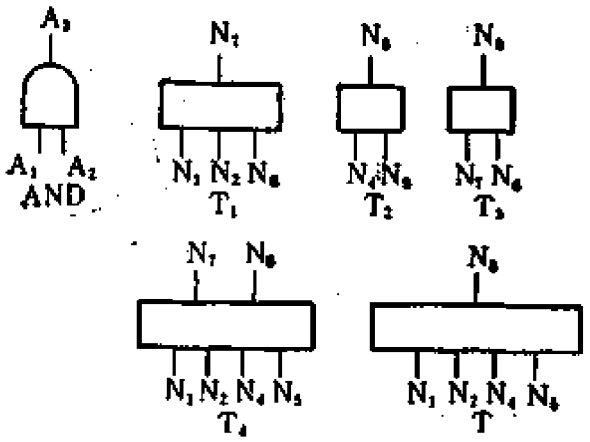
从图中不难看出，知道学生，可找到所选修的课程；同样知道某一课程，也可找到选修它的学生[214]。

数字系统的逻辑线路本身就是网状的，因此用网状设计方法是合适的。下面仅就电路的联接部分再举一例。图 8.21(*a*) 是给定的一个线路图，该线路中的基本部件是“与”门，它可以视为一类，其类型记录名取为 AND。这种类型的记录对应着三个引线记录 A_1, A_2, A_3。“I”表示初级输入引线，“O”表示输出引线。属于本类型的有四个具体的记录，他们是 A01, A02, A03 和 A04。另外，对线路图还可以进行适当地划分，划分成如下一些功能块(部件)：T_1R, T_2R, T_3R, T_4R。各功能块所对应的类型取名为 T_1, T_2, T_3, T_4，整个线路的类型取名 T_0。T_1 类型对应部件记录 T_1R，由基本部件 A01 和 A02 构成。T_2、T_3 类型分别对应部件记录 T_2R

和 T_3R，且分别由基本部件 A03 和 A04 构成。T_4 对应 T_4R，且由 T_1R 和 T_2R 构成。最后由 T_3R 和 T_4R 构成整个线路。部件之间是通过联线联接起来的。因此，对该线路图的数据结构描述不但要有引线表和引线表系，而且还应有部件表和部件表系以及联线表和联线表系。具体描述见图 8.22。例如，从图中不难看出，通过部件表系可知 T 是由 T_3R 和 T_4R 组成。通过类型——部件系又可查到 T_3R 的类型是 T_3，T_4R 的类型是 T_4。通过引线表系查出 T 的引线为 N_1、N_2、N_4、N_5 和 N_8，而且知道 N_8 是 T 的输出，



(a) 线路图



(b) 基本元件和功能块

图 8.21 一个线路图中功能块的划分

其它皆为输入。另外，通过联线表系可查得 T 的联线有 N_1、N_2、N_4、N_5、N_6、N_7 和 N_8，而且 T 的联线联接关系可通过引线值记录和相应的部件——引线系以及联线——引线系查找。

当然，使用另一种表示法也是可以的，即通过联线路径和部件路径把部件、联系和引线联接起来的办法。图 8.23(*a*) 表示了这种数据结构。如以图 8.21(*a*)的线路为例，可表示成如图 8.23(*b*)所示的基本数据结构。

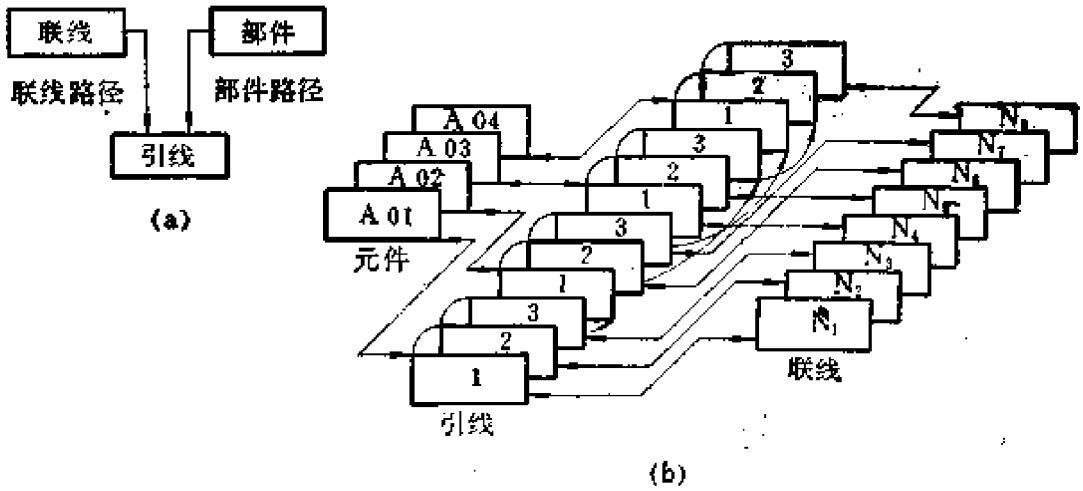


图 8.23　线路图 8.21 的网络结构描述(二)

3. 关系设计方法

(1) 设计方法的特点

以上介绍的两种方法，层次方法和网络方法，均是借助图形来描述数据的逻辑结构的，即把数据模式归结为一个有向图，结点表示逻辑记录，结点间的连线表示记录间的联系。由于数据库的应用不断扩展，由以上两种方法造成的复杂性越来越突出；另外更有用户强烈希望有更自由的查询等原因，IBM 公司的 E·F·Cood 便提出了规范化的思想，引进"关系"这个概念，使之能借助关系代数和关系演算的数学方法实现对数据库的各种操作。所谓关系设计方法[111]就是用满足一定条件的二维表来表示数据的逻辑结构。一个二维表相当于一个关系，一个数据模式由若干个关系组成。实

际上，关系就相当于一个同质文件。在关系方法中，记录间的联系也是用二维表来描述。例如描述学生与选修课之间的关系，这是一个多对多的关系。若借助二维表则可以把这种多对多关系分解为两个一对多的关系。设有三门课程 C_1, C_2, C_3 和四名学生 S_1, S_2, S_3, S_4，则学生与所选课程的关系可用表 8.2 表示。其中 "1" 表示其所在列的学生选修了所在行的课。该表显然既表达了学生与其所选课程的 1 对 3 的关系，又描述了课程与选它的学生的 1 对 4 的关系。

表 8.2　用二维表示的数据逻辑结构

课　程	学　生			
	S_1	S_2	S_3	S_4
C_1	1	1	1	
C_2	1	1		1
C_3	1		1	1

(2) 二维表的特点

使用二维表来表示数据的联系是用户易于接受的。所谓规范化过程实质上是将数据之间的联系逐步地换成二维表的过程。这种表格应具有以下特点：

① 表中每一项代表一个数据项，而且是不可分的；

② 各列的所有项属于同一类型；

③ 各列的名字互异；

④ 不允许有重复行；

⑤ 表格的行和列是无次序的，次序的变动不影响它们的内容和对它们的使用。但由于次序在数学运算中的重要性，一般仍认为次序不同的关系乃是不同的关系。

二维表的每一行称为一个元组 (tuple)；列命名为域 (domain)，关系也可说是元组的集合。

(3) 模型的建立和查询

下面以学习情况为例，从学生登记表和课程表可得到学生关系和课程关系，见表8.3和表8.4。为了反映学生的学习情况还应有第三张表，表8.5即是学习关系。根据这三种关系模型所设计的数据库可查询下列情况：

表8.3 学生关系

学　　号	姓　　名	性　　别
80301	赵　　中	男
80302	钱　　华	女
80303	孙　　兴	男
80304	李　　起	男
80305	邹　　好	男

表8.4 课程关系

课 程 号	课 程 名	学　　时
C301	操作系统	100
C302	编译方法	80
C303	算法语言	60
C304	离散数学	120

表8.5 学习关系

学　　号	课 程 号	成　　绩
80301	C301	90
80301	C302	90
80301	C303	85
80301	C304	87
80302	C301	75
80303	C301	70
80303	C302	95
80303	C304	80
80304	C301	91
80304	C304	88
80305	C301	95
80305	C302	80

① 各对象的情况，如学生钱华的性别；

② 对象间联系情况，如学生赵中各门课程的分数与相应学时；

③ 具有某种属性的对象，如女同学的人数及姓名，操作系统这门课成绩在 80 分以上的人数；

④ 某些对象的统计情况，如某课程的平均分数与最低分数等。

显然，关系与文件，元组与记录是对应的。关系中的数据项也就是记录中的属性。如果属性的值能唯一的标识关系中的元组，则称该属性为关系的关键字；若需要一个以上的属性的值组合起来才能唯一标识元组，那么组合起来的几个属性称合成关键字。一旦去掉关键字中的任一属性后的剩余属性不再组成关键字，则这种关键字是主关键字，它的值是作插入、删除、修改、检索元组时的操作变量。当然还有这种情况，关系 R_1 的某一属性 A_1 不是 R_1 的关键字，但它的值却是关系 R_2 的主关键字的值，此时则特别称 A_1 是 R_1 的一个外来关键字(foreign key)。如表 8.5 中属性“学号”和“课程号”都是外来关键字，它们正好表示了三种关系间的联系。总之，主关键字和外来关键字提供了两个关系中元组之间的联系办法。

由以上的分析不难看出，如果把关系视为元组(或数据项)的集合，那么有关查询、插入等操作实际上是集合中或集合间的操作，这正是数学里的集合运算的内容。比如“并”、“交”运算，就是对两个集合的操作，其运算结果在第三个集合上；“投影”运算就是依据一个集合产生具有不同特点的另一个集合；“联接”运算就是在两个集合上操作，并产生给定特点的第三个集合等。这实际上就是关系代数研究的内容。关于这一点，正好是关系模式优于层次和网络模式的地方：前者的存取路径通过关系代数的运算来实现；而后者则要求应用程序明确指定存取路径。虽然如此，当设计对象非常复杂时，关系将会变得非常大而且多，这不仅实现起来困

难，而且使用这样大而多的表格，其效率是不会高的。因此后来就有主张使用混合模型的设计方法。

(4) 三种模式的关系

综上所述，三种模式各有特点又各有联系；层次模式只是网络模式的特殊情况；而网络模式与关系模式之间又可以互相转化。例如把网络模型中的记录类型和系类型都变成关系框架，那么网络模型就转换成了关系模型。反之亦然。但两者的数据存贮和访问方法是不同的，用层次或网络方法设计的数据库系统，通过指针链查找数据；而使用关系方法设计的数据库系统则通过查表查找数据。正因为改进指针链的查找方法有一定困难，到目前为止进展不大，而加快查表速度的方法还大有可为，这也就是近来人们比较重视关系方法研究的原因。

8.4 数据描述语言和数据库管理系统

数据库系统是一个相当复杂的系统，它不仅需要硬件、软件和数据，而且还需要数据库管理员。

1. 硬件

数据库系统 首先要求有一个大容量的存贮器，以便存放操作系统、数据库管理系统、应用程序及其工作区、系统缓冲区、模式、子模式等。其次需要海量的直接存贮设备(外存)，用来存放各类数据、程序、模式和子模式的描述等。

2. 数据

数据是数据库不可缺少的成分，除用户寄存的应用数据外，还应有描述数据逻辑结构的描述数据。

3. 数据库管理员

数据库管理员(data base administrator)是管理和维护系统的人员，他们的主要任务是：

① 决定数据库的内容；

② 根据外存特征、通道能力和系统要求，选择数据的物理结构和确定访问策略；

③ 通过对用户操作数据的调查和研究，确定系统的模式和子模式，需要时还可修改它们；

④ 规定数据的合法性和有效性的核实与检验方法；

⑤ 制定系统出现故障时的恢复策略；

⑥ 监视系统的工作，响应系统的某些变化，改善系统的"时空"性能，提高系统的效率。

4．软件

这里的软件通常指数据库管理系统(DBMS)，它主要包括：数据库控制系统，模式数据和子模式数据描述语言，数据操作语言以及其他语言等。

一、数据描述语言

无论是应用程序员还是系统程序员都必须严格地描述他们的数据，以便数据库管理系统准确无误地完成对数据实施的各种操作。因此系统应具备数据描述语言 (DDL—Data Description Language)，其中主要指：

1．模式数据描述语言

模式数据描述语言(schema DDL)用来描述模式，它具备定义数据模型功能的能力，是一种与现行普通的程序设计语言不同的格式化的陈述性语言。系统程序员(或管理员)借助它描述数据库数据的全局逻辑结构，用该语言描述的源模式，通过模式编译程序，翻译成目标模式供计算机识别。

2．子模式数据描述语言

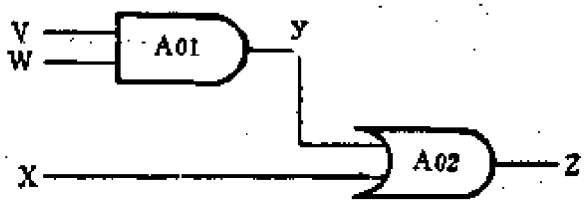
子模式数据描述语言 (Subschema DDL) 是一种按照用户观点描述数据的语言，它用于定义子模式。它往往是用户使用的程序语言（如 COBOL，PL/1，FORTRAN 等)的扩充。这不但有利于用户掌握，而且也易实现其翻译。

此外，对数据存贮结构的描述，对逻辑数据到物理数据映射的描述，对访问规则的描述等均可以单独设计语言或由模式描述语言来完成。

例1　使用模式数据描述语言对图 8.10 表示的记录类型可作如下描述：

```
RECORD NAME IS STUDENT;
  03  S-NO; PICTURE IS 9(5)
  03  NAME; TYPE IS CHARACTER 12
  03  BIRTH—DATA。
    04  YEAR; PICTURE IS 9(4)
    04  MONTH; PICTURE IS 99
    04  DAY; PICTURE IS 99
  03  SEX; PICTURE IS A
```

例2　使用为 CAD 数据库专门设计的模式描述语言——CAD 语言，描述图 8.24 所示的逻辑类型的形式是：



AND-OR

图 8.24　AND——OR 逻辑类型

```
TYPE = AND – OR
    INP JT – PINS = V, W, X
    OUTPUT – PIN = Z
PART = AO1      TYPE = AND
    INPUT – NETS = V, W
    OUTPUT – NET = Y
PART = AO2      TYPE = OR
```

INPUT－NETS＝Y，X

OUTPUT－NET＝Z

二、数据操作语言

数据操作语言(DML—Data Manipulation Language)是用户与数据库系统的接口之一，是供用户访问数据库使用的工具语言。它并不一定是一种完整、独立的语言。而是一些操作语句的集合，其原因就在于它不单独使用（单独使用的询问语言有时也有)，一般是嵌套在应用程序所使用的宿主语言中。因此在设计这种语言时，要考虑与宿主语言语法兼容性的问题。DML的基本功能是：

① 从数据库中检索数据；

② 向数据库中添加数据；

③ 删除数据库中无用的数据；

④ 修改某些数据项的值；

⑤ 用于多用户平行访问控制操作。

下面举几条 DML 语句：

OPEN：　打开有关物理文件，分配缓冲区，调入有关系统程序和表格，为用户访问数据库作好准备。

FIND：　寻找用户所需记录的位置并将它送入记录指示器，标明该记录是当前操作记录。

GET：　把当前记录读取到用户程序工作区。

MODIFY：　修改某记录的数据项。

DELETE：　删除指定记录类型的当前记录。

INSERT：　把应用程序工作区中的记录插入到一个或多个文件中。

REMOVE：　消除一个或多个文件中的某记录。

STORE：　把用户程序区中一个记录存入数据库。

KEEP：　通知 DBMS 使其保持对某个记录的继续访问

权。其它用户无权修改它。

FREE; 消除 KEEP 语句的作用，释放被用户保持的记录。

CLOSE; 关闭文件，消除有关信息，表示用户对数据库访问的结束；除非重新使用 OPEN 语句，否则不能再访问数据库。

除上述语言之外，有些系统还配有其它一些语言。如与DML相配合的数据查询语言，它是一种非过程化的语言，可以单独使用，进行简单的检索和修改等。还有数据库控制语言，相当于操作系统的作业控制语言，用以启动模式、子模式的编译和修改等。另外还有配备设备介质控制语言，它起到与宿主操作系统接口的作用，把数据库各部分映射到直接访问存贮设备上。

三、数据库控制系统

数据库控制系统(DBCS)是指数据库运行过程中所涉及到的各种例行程序。它们的功能是：计算或检索用户所访问数据在外存的物理位置；实现用户数据的保密、安全和完整性控制；对用户访问操作进行合法性与有效性的核实和检验；完成各模式间的数据转换；控制多用户可能出现的平行操作；紧缩数据库中的数据；控制数据通讯；监督数据库的各种活动和测试系统的性能；负责初始建立和出现故障时对系统的恢复等等。主要的例行程序列举如下：

① 初始建立(装入例行程序)；
② 系统恢复；
③ 性能测试；
④ 紧缩空间；
⑤ 数据库系统总控制；
⑥ 多用户平行访问控制；
⑦ 存取；
⑧ 有效性检验；

⑨ 合法性核实；

⑩ 安全、保密和完整性保护；

⑪ 通讯(用户与 DBMS 之间的通讯)。

另外，在数据库发展的早期，由于数据库系统比较简单、数据管理事务也不复杂，故一般都把 DBMS 与 OS(操作系统)合在一起，即扩大通用操作系统的功能，使其包含数据操作功能，以支持某种专门的应用。但是要扩充功能，就要修改系统，这必然造成系统性能的下降，随着数据管理功能的不断增加，后来的设计者就把 DBMS 从 OS 中独立出来了。当然 DBMS 仍然需要 OS 的支持并要求 OS 为其服务。

四、用户访问数据库的过程

为使读者加深对数据库的认识，现以用户通过应用程序读出一个记录为例，说明数据库系统的操作步骤如下：

① 用户在应用程序中，首先要给出所使用的子模式名称，而后在需要读取记录处嵌入一条用 DML书写的读记录语句。该语句给出要读记录的关键字或其它数据项的值。当应用程序执行到该语句时，即向DBMS 发出读记录命令或转入 DBMS 的特定程序。

② DBMS 按照子模式名，查找子模式表，确定模式名，并检验操作的合法性，核对用户的访问权等。若有错误，则向应用程序发出错误信息。

③ DBMS 按模式名查模式表，找对应的目标模式，从中确定该操作所涉及的记录类型，并通过模式到存贮的映射找到这些记录类型的存贮模式。

④ DBMS 查阅存贮模式，确定读取所要记录的物理文件、区域、设备、存贮地址以及应该调用的访问程序。

⑤ 访问程序找到有关的物理数据块的地址，并向操作系统发出读块命令。

⑥ 操作系统按照收到的命令从存放数据库信息的外存读出

数据块到系统缓冲区，并发出操作结束信息给 DBMS。

⑦ DBMS 根据操作系统发出的结束信息，按照模式、子模式的定义将缓冲区中的内容映射成应用程序所需要的逻辑记录，并送入应用程序工作区。

⑧ DBMS 向应用程序发出执行成功与否的信息。

⑨ 应用程序按照接收的回答信号决定是否利用工作区中的数据。

图 8.25 显示了访问数据的过程。

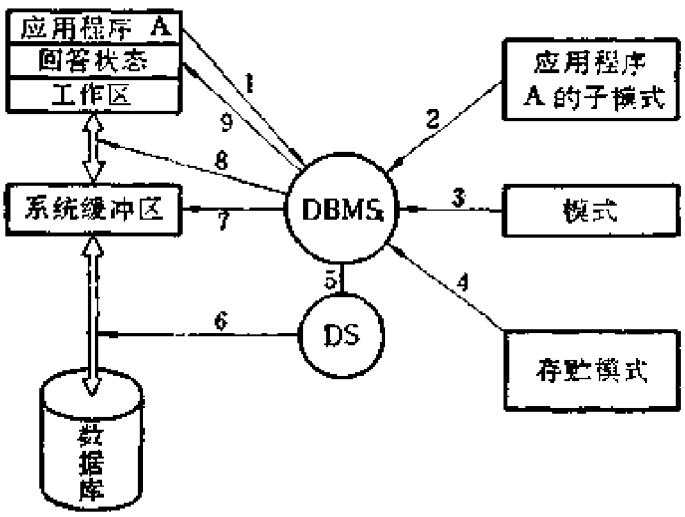


图 8.25 访问数据库示意图

如果用户更新一个记录内容，其操作步骤是类似的。首先把记录读到工作区，经过修改后向 DBMS 发出写回的命令，以与读出相反的步骤把修改过的记录写回原位置上去。

8.5 数据库的发展

虽然目前对“数据库”这个术语的来源及其确切定义均尚无定论，它的发展历史也比较短，但是，数据库技术已被公认是计算机应用方面的一个很活跃的领域，它是计算机软件的一个重要分支。它的发展速度是十分惊人的，目前估计世界上有近万个数据库系

统在运行。它的应用面也很广，如企业管理、交通运输、银行业务、产品分配、预测供求情况、搜集和检索情报资料、档案管理、航空服务、旅馆业务、普查和统计人口、制订国民经济计划等。不仅如此，它也是计算机科学及其它领域(如计算机辅助设计、人工智能等)不可缺少的工具。总之，数据库将逐步成为人类活动的公用资源。

早在1962年 C. W. Bachman 为了更有效地使用直接访问存贮设备而设计了集成数据存贮 (Integrated Data Store)系统，它的基本思想是连接不同类型记录的灵活的模式。可以把它看成是数据库的原形，对早期数据库的研究影响很大。不过早期的数据库仍属文件系统的范畴，并未解决数据库的实质问题，即数据的独立性、数据共享、交叉访问、扩充新的数据、冗余和结构等。真正使数据库从传统的文件系统中脱胎出来是六十年代后期，具有代表性的成果是：

① 美国数据系统语言协议会 (CODASYL) 所属的数据库任务组(DBTG)提出的报告，它包括了模式数据描述语言、子模式数据描述语言和数据操作语言。以文件的形式确定了数据库设计的 DBTG 方法，即网络方法。随着 CODASYL 工作的进展，陆续出现了许多商用数据库系统，如 Univac 公司的 DMSNOO 系统(1971年)、Cullinane 公司的 IDMS 系统(1972年)、Honewell 公司的 IDS/2 系统(1975年)。

② 1968年 IBM 公司研制了与计算机配套的 IBM 系统，该系统是层次方法的典型代表。

③ IBM 公司的 E. F. Codd 于1970年发表了“大型共享数据银行数据的关系模式”论文，从而开创了数据库的关系方法和规范化理论的研究。1975年以后，出现了商用的关系数据库，如 IBM 公司的 Query By Example 系统。该方法的出现，使数据库的工作由工程和实践阶段走上了理论研究阶段。

目前，对数据库的设计原则和方法虽然进行着不断地总结和探讨，使之通用化、标准化和理论化，但总的说来仍处在从工程实

践向理论过渡的阶段。它的概念、原理和方法还在继续发展和变化。以下几个方面乃是当前要研究的重要课题：

① 研制大容量的新存贮器件；

② 探讨数据库设计的方法论，诸如数据模型、数据的存贮与访问以及数据库的管理和保护等，使设计工程规范化；

③ 数据库规范理论的研究；

④ 数据库标准化，使系统更加统一和通用，使用户接口简单；

⑤ 研究分布式数据库系统，使数据库技术与计算机网络结合起来；

⑥ 解决 DBMS 软件运行时占用大量 CPU 时间的问题；

⑦ 研制我国实用的 CAD 数据库系统。

第九章　布图设计系统概述

布图设计方法的研究除了它们自身的学术意义外，主要的目的在于建立一个有效的布图设计自动化系统。彻底改变集成电路设计中，尤其是LSI/VLSI随机逻辑电路设计中的手工作业方式，使LSI/VLSI的设计进入计算机辅助设计——自动设计的新时代。从某种意义上来讲，这也是集成电路的进一步发展，尤其是LSI/VLSI的发展的必要先决条件之一。

一个布图设计自动化系统的设计目标一般可分下述四个方面来予以阐述：

(1) 应使设计过程尽可能地实现自动化

为了达到这个目标，将要求在各设计阶段中都具有完善的设计软件和必需的硬件支持，并尽可能地减少人的干预。

(2) 应使设计结果具有令人满意的设计质量

为了达到这个目标，除要求各部分的设计软件具有足够高的设计精度外，从某种意义上来讲，更重要的是使各部分设计和各阶段的设计具有更好的总体合理性。如各模块设计的总体合理性、布局设计、总体布线和最终布线之间的总体合理性。由于这些问题往往都是一些相当困难的问题，因此在目前的设计系统中，为了保证设计质量或解决自动设计遗留下来的问题，一般都通过人机交互并借助于设计者的智慧来解决。

(3) 应保证设计结果的正确性

对于集成电路的布图设计来讲，在成千上万个(在VLSI中将是几十万到几百万)图形、线条、接触孔中，哪怕只有一处严重的错误(如少一个关键的接触孔、二个图形间严重地违反了工艺规则所规定的最小间隔要求，以及连错一条连线等)，都将使设计得到的

电路掩膜用于实际生产时，成品率等于零。为了保证设计结果的正确性，一方面应使设计过程尽量自动化并尽可能地减少人工的干预。现有的系统在实际设计中得到的经验是：几乎100%的差错都是在人进行干预时引起的(如人工准备初始数据、人工读入、人机交互修改某些数据时)。由于不可能完全避免人的干预，而且有时人的干预甚至是必须的，因此保证设计正确性的另一方面的措施是发展、建立各种设计结果(包括中间结果)验证的手段。由于不少验证结果的软件在处理时往往需耗费相当多的计算机时间，因此研究、发展高效的验证软件也是布图设计中一个有意义的课题。

(4) 应使设计成本尽可能地降低。

这个目标的实现要求在系统的建立时，应结合具体的设计任务和设计要求，选择合理的硬件配置和软件设计方法。

本章将围绕上述目标对布图设计系统中所涉及的问题和处理方法作概要的介绍，并将把重点放在有关算法的讨论上。

9.1 布图设计系统的硬件配置

一、系统硬件配置概述

由于设计任务和设计模式的不同，各布图设计系统的硬件配置也不尽相同。图9.1为布图设计系统的一般硬件配置图。可以看到，系统一般具有较强或很强的人—机交互功能，当主机具有多用户分时处理功能时，可配置多台以图形显示器为中心的布图设计工作台。用户在布图设计工作台上可利用键盘或图形输入台及数字化笔把数据或图形输入计算机，可通过键盘和字符显示器(CRT)控制设计过程。字符显示器上将通告用户设计过程进展的情况及设计中间结果的评价信息。用户通过图形显示器可在设计的任一阶段显示当前设计的直观结果并利用键盘或数字化笔进行必要的修改，直至获得满意的设计结果。系统的另一个特点是

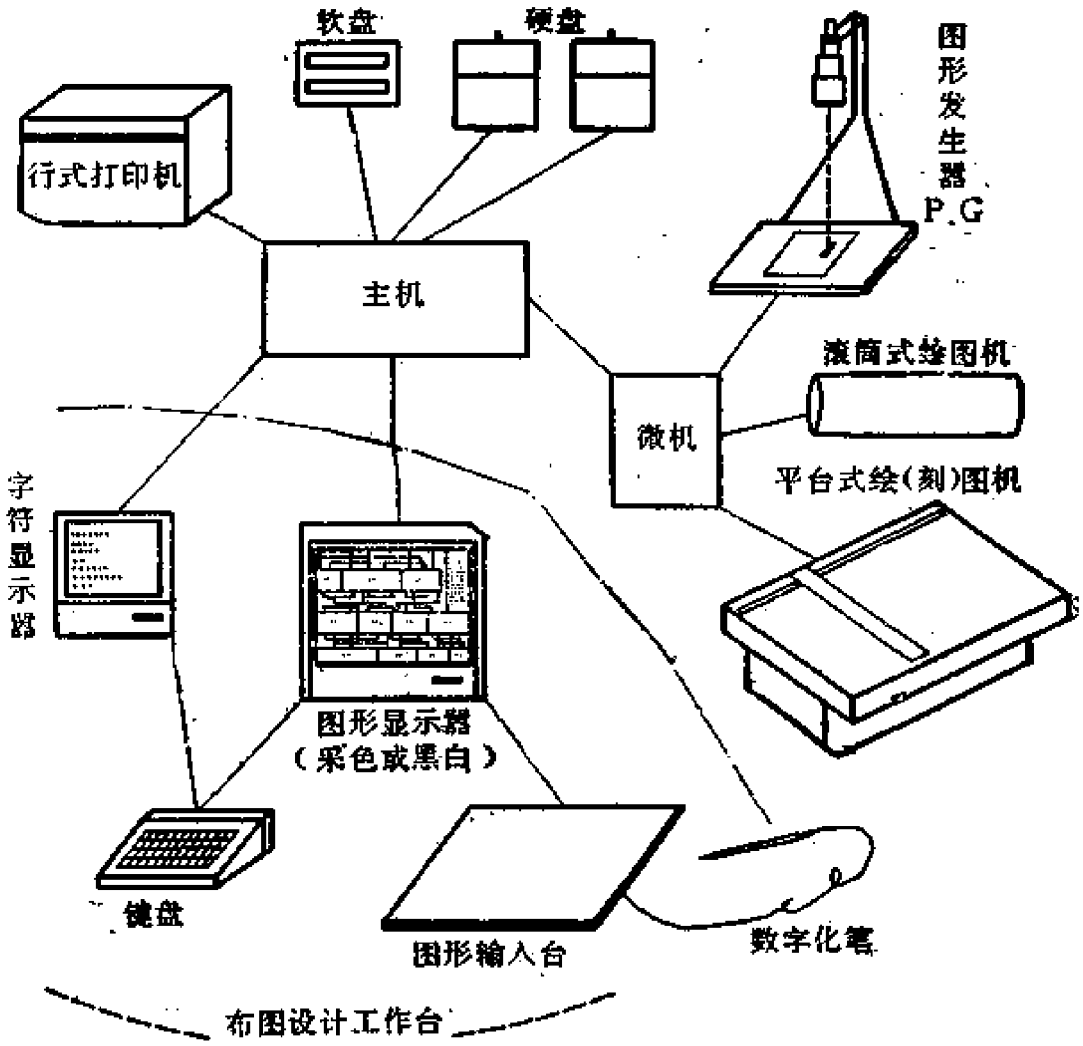


图 9.1　布图设计系统硬件配置图

备有一些专用的输出设备，微机控制的绘(刻)图机可将结果精确地描绘出来。在需要时，也可在平台式绘(刻)图机上刻制放大500倍的掩膜图形(在红膜上)，然后揭去不需要的部分的红膜，经过照相、缩小和分步重复后得到实际的电路掩膜。此外也可以在微机控制的图形发生器(光学的，或电子束的图形发生器)上直接制作放大5～10倍的中间掩膜版，然后经缩小、分步重复后得到实际的集成电路生产用掩膜。

图9.2示意地图示了上述掩膜制作过程。图9.2(a)为刻图机在红膜刻制的掩膜图（图中为一个双极晶体管的基区—扩散

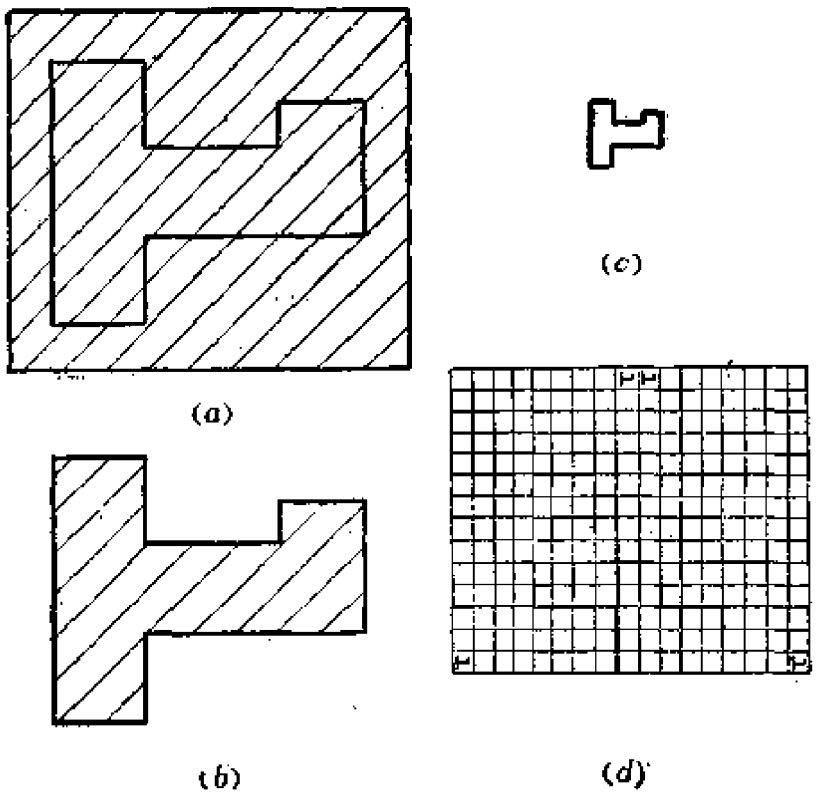


图 9.2 掩膜制作过程

区)，图 9.2(b) 为揭去不需要的红膜后的情况，此时图形的尺寸是实际尺寸的 500 倍(或 200 倍、400 倍、1000 倍等)。图 9.2(c) 为经照相初缩后的图形，此时图形的尺寸为实际尺寸的 5～10 倍。用图形发生器制版时，得到的结果即为 9.2(c)，图 9.2(d)为精缩、分步重复后的实际掩膜。

一套生产某一电路的掩膜一般约有 6～11 种用于各工艺步骤的掩膜。因此，高精度的掩膜制作设备也是布图系统的重要基础。

二、系统硬件配制实例

1. HP 公司的 HP98450 图形处理系统

该系统以 VAX11/750 为主机，包括的硬件配置有：

图形显示器(13 英寸，彩色) HP98450

图形输入台	HP9111A
页面绘图机	HP98720
滚筒式绘图机	HP7580A
针型打印机	HP2631B
软盘机	HP98859～HP9885M

2. Fujitsu 公司的计算机辅助设计系统

日本 Fujitsu 公司的一个计算机辅助设计系统的硬件配置使初始信息的输入更加自动化了 [162]。他们采用一台 FACOM M 系列的计算机作为主机（从 M—150F 到 M—200 型），人机交互通过 CAD 工作台实现。Fujitsu 的 CAD 工作台包括一台 20 英吋彩色图形显示器(F9434, 1024×1024)、一个图形输入台、一个字符显示器及一个键盘。一台 FACOM M150F 可联结四个这样的 CAD 工作台。Fujitsu 公司计算机辅助设计系统中用一台微机（PANAFACOM—U--400）控制一台筒形扫描器和一台打印—绘图仪。用筒形扫描器可直接把 80×80 cm 的手画的电路图或逻辑图读入计算机。从而使初始信息的输入改变了以往的人工数字化读入的方式，提高了自动化的程度和减少了出错的机会。

随着布图设计系统的发展，采用模式识别等先进技术，实现图形信息的自动输入和识别，将成为人们关注的问题之一。此外，更方便，更正确地表达设计者意图的人机交互手段的发展也是人们所关心的。而在适当的硬件支持下，软系统将具有更广大的发展前途，也必将使整个系统的水平得到进一步的提高。

9.2 初始信息的准备及正确性验证

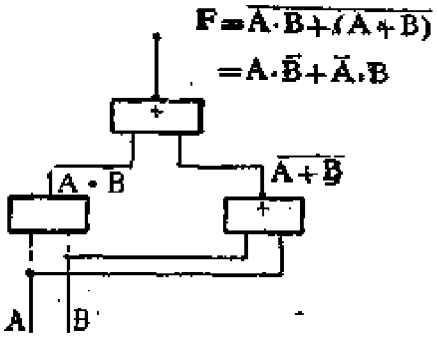
初始信息的准备即设计对象的描述是用户和计算机联系的重要界面。描述及输入方式的简便及有效性往往是用户最关心的问题之一，也是衡量整个系统的有效性的重要指标之一。其主要内容已在第三章作了一定的介绍，本节将对一些专门问题进一步加以

讨论。

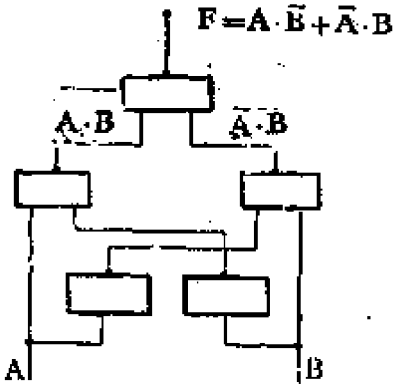
初始信息的准备与系统所采用的设计模式和设计方式有着密切的关系。

一、门阵列设计模式中电路的等价转换

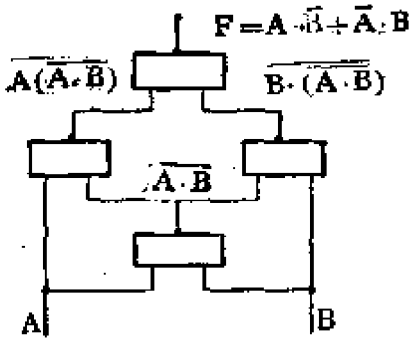
在一般的门阵列设计模式中，由于母片上单元的类型、功能是预先设计确定的，因此对于用户需设计的具体电路往往首先要求进行必要的逻辑转换。如设计采用的母片为双极二输入端与非门阵列，则图 9.3 的用户电路将必须预先转换为以二端与非门为基础的等价逻辑图，才能用描述语言输入计算机。对于复杂的电路，这个工作通常由相应的软件自动地实现。从图例中的简单例子（一个二输入异或门）可以看到，在转换时存在着如何优化的问题，这个问题更多地属于逻辑综合的问题，在本书中将不作进一步的



(a) 用户电路



(b) 转换成5个门的等价电路



(c) 转换成 4 个门的等价电路

图 9.3 电路转换成等价逻辑图示例

讨论，但提出这个问题对用户和设计者来讲，都是必要的和有意义的。

除了研究、发展优化的转换软件外，改变母片上单元的设计方法，使其功能（如输入端数）具有一定的可调整性，也是人们关心的一个问题。

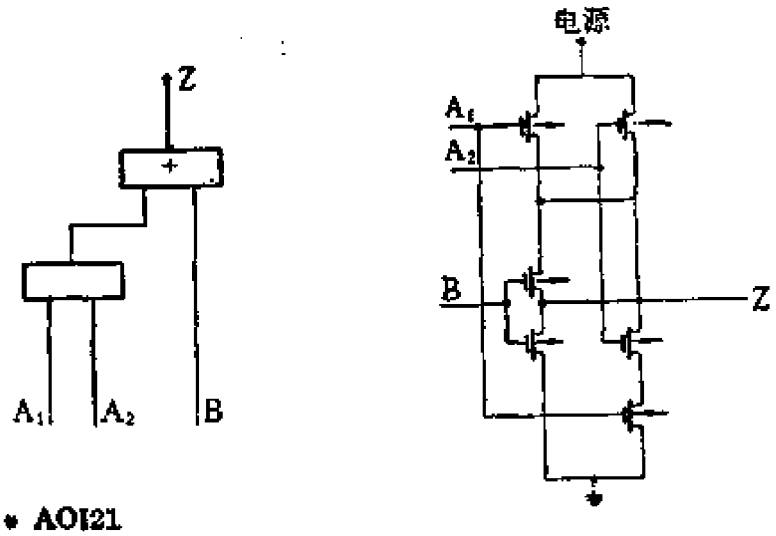
二、多元胞模式中标准单元的设计和描述

在多元胞设计模式中，标准单元的设计是初始准备的主要内容之一。有些系统采用人工设计方式来进行标准单元的设计，并采用图形描述语言将人工设计的布图结果输入计算机。经过设计者的精心并反复多次的设计，可使标准单元具有极好的合理性，并为提高设计的电路质量奠定了基础。但是这种方式设计的标准单元当工艺水平或用户要求发生变化时，修改将变得非常困难，以致于不得不重新设计这些标准单元。尤其是近年来，与微细加工技术相关的集成电路制造工艺水平得到了相当快的发展，平均每二年，集成度增加一倍，这就客观上使标准单元的修改变得更经常化，因此人们越来越倾向于用标准的方法（如符号法等布图设计方法）来设计标准单元。

1．单元设计的描述

图 9.4 为一个二与一或的标准与或非门单元的符号法设计描述。请读者注意，在设计描述中并不涉及版图设计的工艺规则，如多晶硅条宽度、孔的尺寸等。图中 AOI21 为标准单元名，INPUT 为输入端名，OUTPUT 为输出端名，N—FET 与 P—FET 分别为 N 沟和 P 沟 MOS 管位置。NET 为连线描述；LEQ 为逻辑等价接点描述；在本例中 LEQ L1，2，表示第 1 个输入端（A_1）和第 2 个输入端（A_2）是逻辑等价接点。

在采用上述描述方式时，将把各标准单元电路制造有关的工艺参数集中地统一地加以描述（如扩散条宽、间距；窗口尺寸；多晶硅条宽，间距；金属线宽，间距；……等等）。当由于工艺的进步或



符号描述 * AOI21

```
NAME AOI21
INPUT A1, A2, B
OUPUT Z
N-FET VSS A1 I1 A2 Z—Z B VSS Z
P-FET VDD A1 I2 A2 VDD—I2 B Z Z
NET Z:5, 2:2, 2:7, 3:10, 3;
NET Z: 9, 4:9, 3;
NET 1, 2:3, 4:7, 4;
LEQ L1, 2;
```

图 9.4 符号法设计描述实例

用户设计要求变化时，只需修改相应的工艺参数就可以方便地得到全部新的标准单元。相信在工艺技术飞速发展的今天，这种设计方式将成为主要的单元设计方式。

2. 单元描述文件

为了检查和进一步设计的需要，对于给定的单元布图设计描述，应可用系统软件产生四种单元描述文件。

(1) 布图设计时的单元模型文件

这种文件将忽略单元内部的具体结构，而描述单元的外部特征。如单元的长和宽；各接点的位置和层次；各接点间的等价关系(电学等价关系可自动地由软件建立，并由软件自动定义过渡通道)等。

(2) 用于电路分析的单元模型文件

电路分析软件如 SPICE[165] 可用来模拟分析单元电路的特性是否与用户的原设计要求相符。它可以进行电路的非线性直流分析、非线性瞬态分析和线性交流分析。还可用于直流小信号灵敏度、噪声、失真特性及输出波形的富里叶分析。可模拟分析电路在各种温度下的特性变化情况。为了进行电路分析，必须向电路分析软件提供电路中所含各种元件(电阻、电容，二极管、晶体管)，包括寄生元件的电参数及这些元件的联结关系。系统软件可根据单元电路的布图描述原文件和给定的工艺参数自动地生成上述分析所需的数据和参数[191]，形成电路分析时的单元模型文件。

集成电路版图上电路元件及其参数的提取一般通过对各层掩膜图形的逻辑运算和拓扑运算来实现[173]。

① 图形的逻辑运算包括对二个图形集的与(AND)、或(OR)、非(NOT)、差(SUB)、异或(XOR)、与非(NAND)及或非(NOR)运算等，如图 9.5 所示。

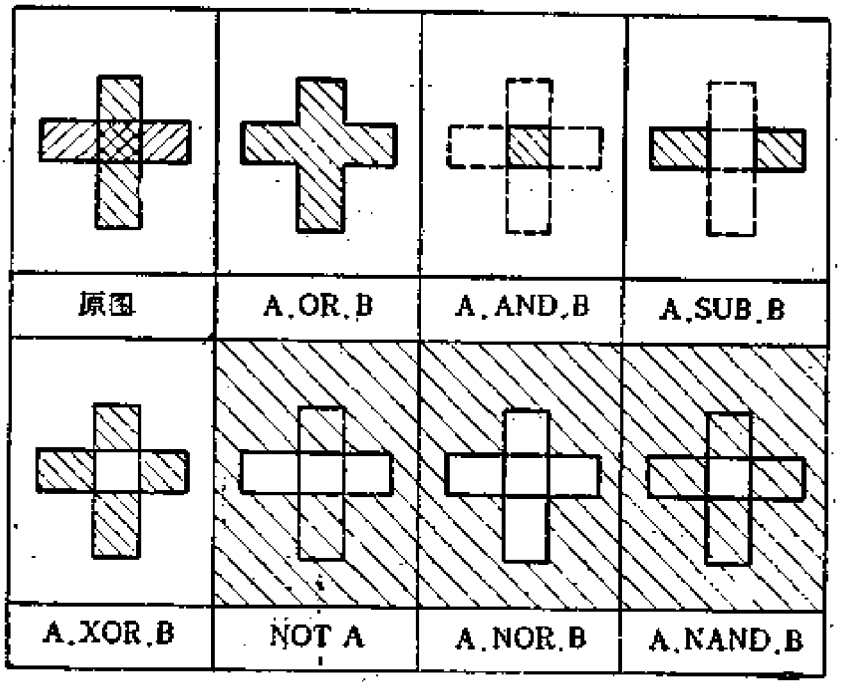


图 9.5 图形的逻辑运算示意图

② 图形的拓扑运算包括含有(CONTAIN)、重迭(MEET)、分离(DISJOINT)、接触(TOUCH)、相交(INTERSECT)等判断处理，如图 9.6 所示。

根据电路的性质和工艺类型，利用上述运算或运算的组合，就可方便地识别版图上的电路元件并提取出相应的几何尺寸参数(如 MOS 电路中的沟道的宽长比，双极电路中的发射区面积等)加上已确定的工艺参数(如结深、栅氧化层厚度，场氧化层厚度，发射区渗杂浓度，基区杂质浓度等等)。就可以计算得到所需的电学参数。

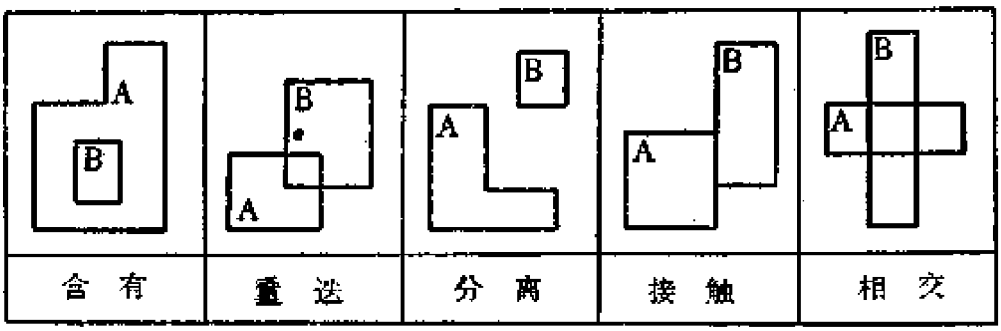


图 9.6 图形的拓扑运算示意图

以 N 沟 E/D MOS 电路为例，典型的制造工艺将有扩散、耗尽型注入、多晶硅、接触孔、引线孔、铝线等掩膜版图。根据电路特点和制造工艺的性质可知，多晶硅条和扩散区相交处将形成一个 MOS 管。当我们将多晶硅层掩膜的图形与扩散区层掩膜的图形集进行“与”(AND)运算时，其结果可用来确定各 MOS 管的位置，并得到这些管子的一些有用的几何尺寸。若令结果图形集为 T_{AL}。当把 T_{AL} 与耗尽型注入层图形集进一步作“与”运算后，得到的结果指明了所有耗尽型晶体管的位置和几何尺寸。令该结果集为 T_E，则 $T_{AL}\cdot SUB$、T_E 的结果指明了所有增强型晶体管的位置和几何尺寸。同样我们可方便地提取多晶硅连线和铝线的寄生电容等所需的电参数以及它们所体现的连接关系，最后形成电路分析

用的单元模型文件，在用户认为必要的时候，可调用这个文件去进行单元电路的分析模拟。

③ 用于延时分析或时间关系分析（Timing）的单元模型文件。电路分析软件虽然分析精度高，但处理效率较低。对于设计的整个电路的分析模拟则需要另一种精度较低但处理效率较高的软件，以满足处理大型电路的需要。延时分析软件的典型代表是MOTIS软件[169]在获得了电路分析用的单元模型文件后，生成延时分析用的单元模型文件是十分简捷的。

④ 用电路图方式描述的单元文件。这种文件可以通过绘图机以电路图的形式输出，或者使用户可直接在图形显示器上看到单元布图设计所对应的电路图。用户可以从电路图来检查电路的电特性，逻辑特性是否满足设计要求。当利用图形的逻辑运算和拓扑运算以版图中提取出电路的元件及它们的连接关系后，生成这个文件也是十分简捷的。需要注意的一点是应该使其输出形式尽量符合人们的习惯，即符合绘制电路图的一般规范。

三、分级描述方式

讨论了单元有关的描述后，下面我们将进一步对整个电路的描述作一点讨论。

在目前的系统中，在应用设计对象描述语言或其它手段进行问题的描述时，对于规模较大的电路，一般采用分级描述的方式。在这种方式下电路将划分为若干模块，这里存在着一个所谓“划分”[201]的问题，由于目标的多样性（模块内部的合理性和模块间的总体合理性等）这个问题在理论上也将是一个非常困难的问题。但在实际设计中，划分问题并不像想象的那么困难。对于主要模块的划分，人们常常根据经验和电路功能的自然划分可方便地得到相当满意的结果，需要注意的是，此时设计者的经验和技术素养将对结果有较大的影响。

由于各模块电学、逻辑特性的不同，它们的布图结构也有相应

的不同特点。在近年来发展起来的设计系统中，人们愈来愈注意到采用不同的设计模式来设计不同特点的功能模块的必要性。如用人工设计方式(版图编辑)来设计 RAM，用 PLA 方式设计计算机的控制逻辑，用多元胞或栅阵列方式设计 ALU 和缓冲存贮器等等。这就要求系统在问题(对象)描述时，能兼容各种描述方式，并使各种以不同方式描述——设计的模块在外界面（即模块间连接有关的信息)上能够统一化和规范化。因此，一个良好的设计数据库是必要的。

四、描述正确性验证

描述的正确性是保证设计正确性的先决条件。同时尽管用户设计的电路已经过人的严格检查或已通过计算机得到了验证，但在描述和输入计算机时，往往不能避免人工的操作过程(当具备把经模拟验证的逻辑图直接输入计算机的系统功能或直接将模拟的结果信息作为布图设计的初始信息时，上述问题可以避免或减轻)，因而相应的初始信息正确性检查是必须的。

一般，在应用设计对象描述语言给定问题后，描述语言编译系统将在原文件输入时进行语法(描述规则等)检查，除此之外，系统常常还具备两种检查手段可供用户调用。

1．高效率的常规检查软件

这种软件可以较高的处理效率检查出大部分操作人员在描述和输入时引起的差错。

这种软件中含有相当多的检查规则，但几乎每一条检查规则都是相当直观和简捷的。以连接关系描述为例，设 NET 301; CELL 101, 7; CELL 9, 4; CELL 47, 3; 描述了一条电路中的连线。则常规检查的一些规则可以是：

① 分别检查是否存在 CELL 101, 9, 47 的类型描述。

② 分别检查 CELL 101 的第 7 个接点是否存在；是否是具有实际意义的接点；是否是已用于其它线网的接点；与其他接点的连

接是否正常(如同一线网中是否存在二个输出端;是否把信号接点与地线或电源线短路了;等等)。

③ 检查是否有过多的单元的信号接点浮空的情况出现(即不与任何线网相连的情况)。

④ 检查单元输出端的扇出是否正常、即与该输出端相连的输入端总数是否超过电路性质所允许的值,等等。

值得注意的是,与描述语言的语法检查类似的,在常规检查中违反检查规则的情况并不一定都是真正的错误,而可能是一个"警告",错误的确定和修改将由用户或操作人员来实现。

人们常常有一个误解,认为常规检查既然在理论上不能保证

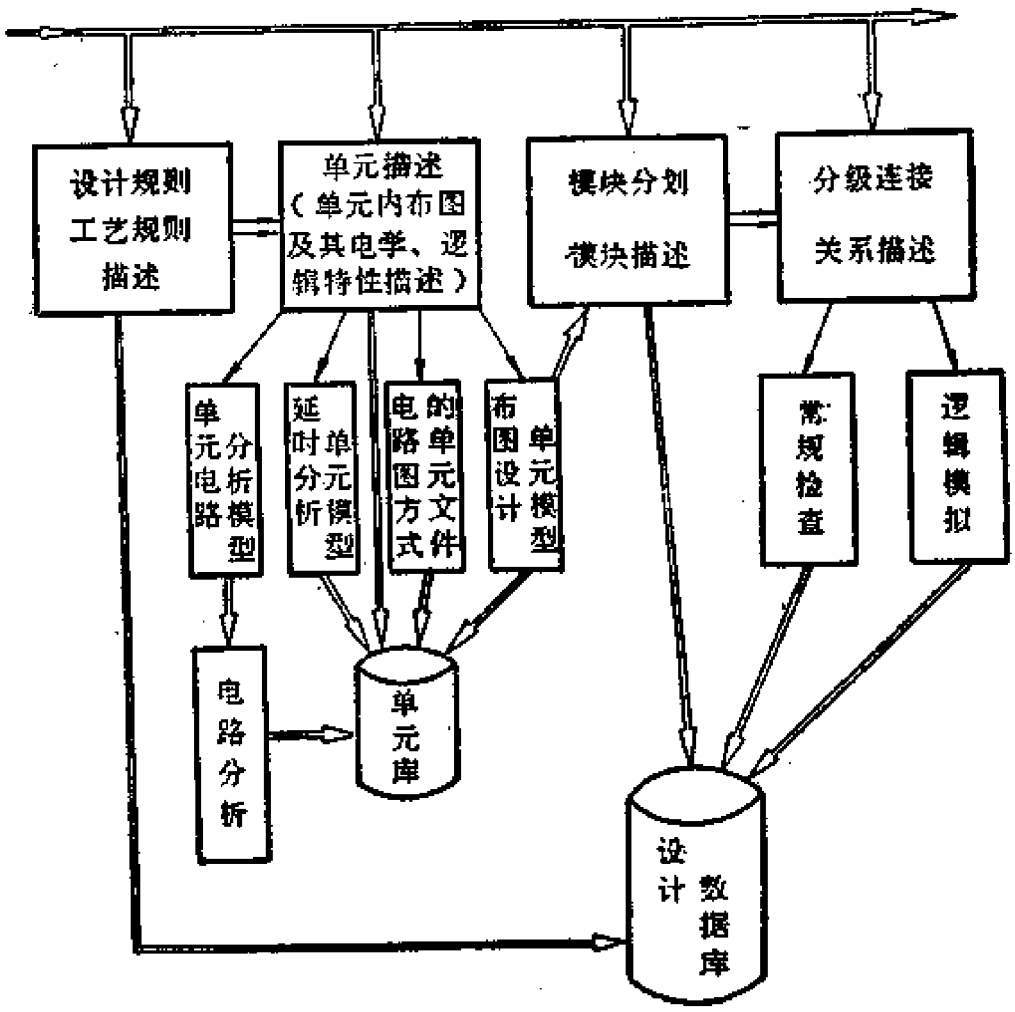


图 9.7 初始信息准备及正确性验证

检查的严格性，因此没有值得重视的实际意义。事实恰恰相反，由于常规检查处理效率相当高，一个设计得较好的常规检查软件不仅容易适应规模庞大的设计问题的检查，并在实际应用中具有相当高的有效性，以致只有一小部分实际问题才真正需要用严格的但费时很多的检查手段，在设计开始前对设计问题的描述进一步进行严格的检查。

2．利用逻辑模拟进行描述正确性检查

系统中采用的更严格的检查手段实际上就是电路的逻辑模拟，此时是利用输入到计算机内的问题描述信息，以设计要求的电路功能为目标进行逻辑模拟，用来检查单元描述和连接关系描述的正确性。对于规模很大的设计问题，检查将耗费相当多的机时。图 9.7 给出了初始信息准备和正确性验证的过程示意图（多元胞模式）。

9.3 人机交互子系统及其功能

对目前的设计系统来讲，在具体设计过程中，由于布图设计问题的复杂性和具体设计要求的多样性。为了获得满意的设计结果，借助设计者的智慧和判断力往往是必需的。此外由于初始信息修改、设计过程的控制、对部分自动设计遗留下来的问题的处理（如门阵列模式设计时的剩线处理），由于特殊设计要求对设计结果的修改等等也需要有一个功能健全的人机交互子系统 [164] [187]，使设计者与计算机之间有一个交换信息的界面。交互子系统的功能是系统有效性、实用性的重要决定因素之一，已经并将继续得到人们的重视。本节将对其基本功能和作用作简要的介绍。

一、设计跟踪及设计过程控制

在 LSI/VLSI 布图设计中，整个设计过程的各个设计阶段实质上是相互联系、密切相关的，一个不合理的布局将很难或根本不

可能得到一个好的布线结果。

1. 设计跟踪

所谓设计跟踪就是在每一个设计阶段完成后，系统通过字符显示器或图形显示器把该阶段设计结果的有关信息通告用户，并将一些需注意的问题对用户进行提示，以便用户进行设计过程的控制。

从布局设计过程为例，当自动布局软件对给定的问题得到一个布局结果后，系统将可把下述信息通告用户：

① 该阶段设计所耗费的机时；

② 在多元胞模式中将给出预计完成全部设计后芯片的总面积和面积利用率(也可以各子块列高度的下限值分布来度量)；

③ 在门阵列模式中给出可布性评价；

④ 连线长度有特定要求的线网在布线完成后满足设计要求可能性的预计；

⑤ 其它有关的信息。

在设计跟踪中，必要时还可调用以往设计中或前一个设计阶段的设计跟踪信息以帮助用户进行比较和判断。

设计跟踪要求系统发展与之相关的设计结果(中间结果)的评价软件，好的有效的评价软件如可布性分析软件将对提高设计效率、降低设计成本具有重要的意义。

2. 设计过程控制的作用

设计的过程控制使用户具有调度设计过程的可能性。用户可在任一设计阶段进行"重入"。即选择任一设计阶段的中间结果开始重新设计，并用较满意的阶段设计结果去代替原设计结果。

设计的过程控制也可使用户在需要的时候中断设计进程，并在合适的时候重新继续该设计进程，从而使设计过程具有更大的灵活性。

设计过程控制更重要的功能是可以在设计过程中方便地借助于人的设计经验和智慧，为了获得满意的设计结果，这一点常常是

很重要的。我们知道，初始布局的构形往往对于迭代改善后的结果有相当大的影响。有时在初始布局进行前，用户通过选择适当的算法参量(如规定多元胞单元行的长度等)；或对某些单元预先指定其布局位置(在有些算法中称之为确定一些"核")；或对某些引出接点的位置预先作专门的分配等等来控制初始布局的结果。在布局的迭代改善中，由于算法的限制，软件对某些单元的位置组合将是不灵敏的（即软件无法识别的某些实际上是好的迭代对象的单元组合)，此时，借助于设计者的智慧，通过修改某些算法参量或直接指定迭代对象将使设计过程可能继续处理而获得一个更好的结果。

在有些系统中，在若干设计阶段存在着多个可选用的设计软件，如成对交换法的迭代改善布局软件和单元插入法的迭代改善软件等等，设计者可通过设计过程控制根据需要调用不同的软件来处理该阶段的设计任务，可以比较各软件的设计结果择优取之；也可以交替使用不同的迭代改善软件去寻求更满意的结果。

自然，在设计过程控制中对部分较小的设计问题也可采用全自动设计的方式进行。

可以看到一个好的设计跟踪和设计过程控制子系统将使整个设计系统具有更大的灵活性和有效性，将可更好地在设计过程中借助于设计者的经验和素养进一步提高设计效率和设计结果的合理性。在分级设计方式中，设计过程控制将具有更加重要的意义。

设计过程控制要求系统的各设计软件应实现模块化，并要求对设计数据进行集中的统一的管理。这样也将使多用户可以共享系统的软件及数据资源。

二、数据编辑及图形编辑

1. 数据编辑

数据编辑的功能主要是设计参数和设计数据的输入与修改。

这时可以方便地借鉴或借用一般用于其它目的的系统数据编辑软件。

2. 图形编辑

具有特色的是布图设计系统的图形编辑功能，它的主要用途是实现图形的输入、显示和各种修改。

(1) 图形输入

一般的交互子系统中都具有功能齐全的图形描述语言，可方便地处理各种图形，如线段、折线、矩形、各种多边形、圆、弧等，并可以描述图形或图形组合的各种变换(如平移、x 轴或 y 轴镜象变换、旋转变换，等距放大缩小、部分延伸或收缩等)。通过键盘或图形输入台用图形描述语言可方便地把图形或图形的集合输入计算机。图 9.8 为一些基本图形的描述方法。图 9.9 为一些基本变换的含意图示[194]、[195]。

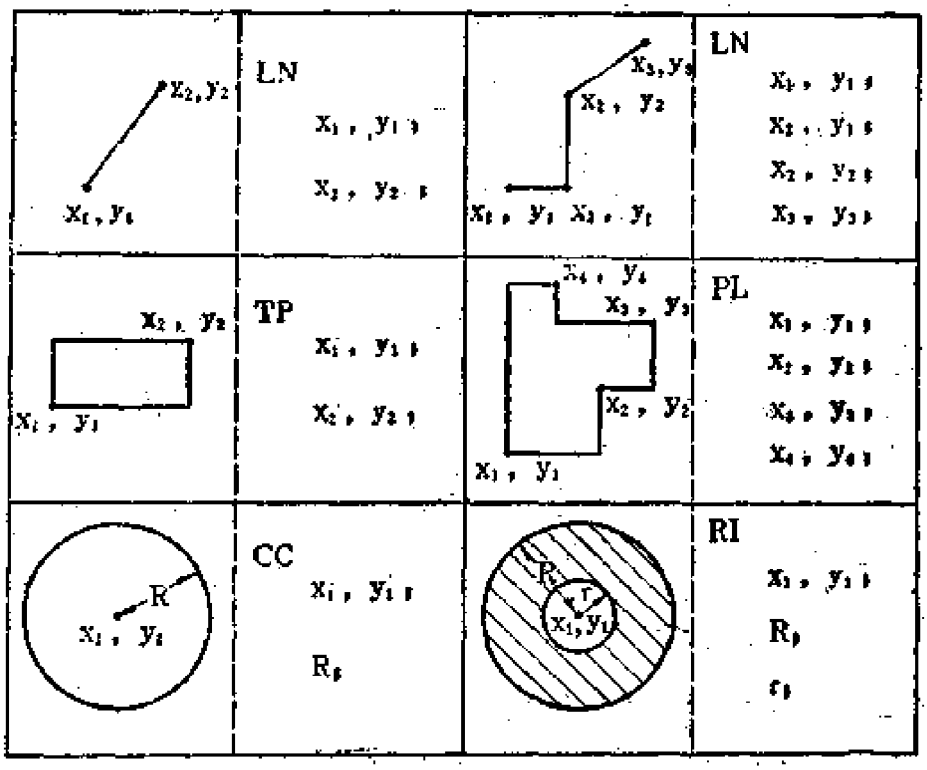


图 9.8 基本图形的描述方法

在图形输入时，利用图形输入台和数字化笔将可提高输入效率并减少差错。这时，可把需输入的图案放在图形输入台上，在定义了基准点的座标后，对组成图案的每一个基本图形，可首先通过数字化笔在图形输入台的输入图形种类标识菜单中定义该图形的类别(只需用数字化笔在菜单相应位置点一下)，然后按图形描述语言规定的格式，用数字化笔指点表示其特征的点的位置。如对一个矩形图形，首先用数字化笔指点菜单中"TP"项或图示的矩形项，表示将要输入的图形种类，然后用笔在矩形的左下角和左上角分别点一下，该图形就输入了计算机。由于避免了人工对图形特征点座标的读取和具体数字在键盘上的输入操作，从而可显著地提高处理效率并减少差错。

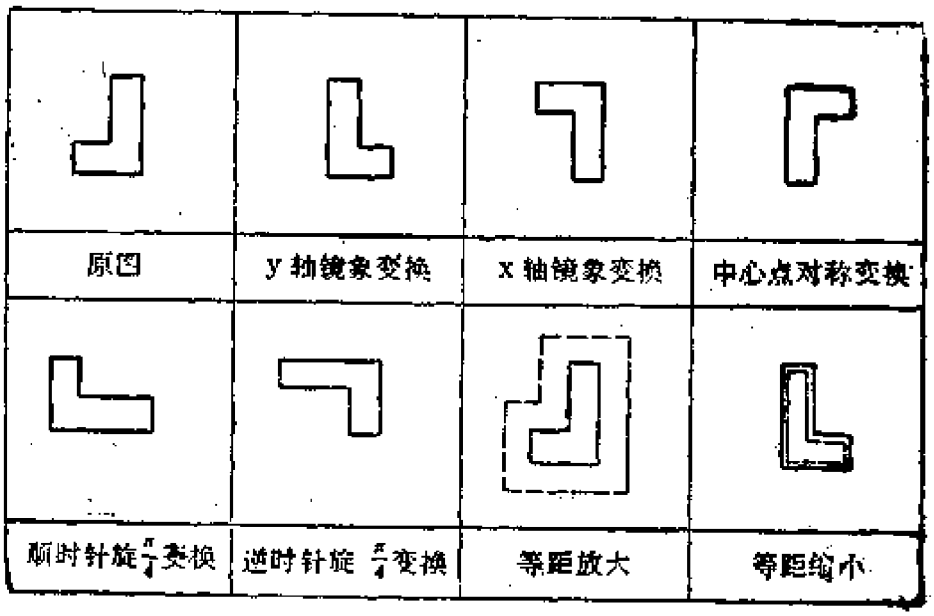


图 9.9　若干图形变换图示

利用图形变换功能，可使图形尤其是图形组的描述变得更加简练，这样不仅可进一步提高处理效率，更重要的是简练的描述可进一步减少发生差错的可能。

(2) 图形显示

为了进行必要的修改，图形的显示一定位是必须的，系统将允许用户显示每一级设计过程的结果。如显示输入的或调用的单元

图形、显示布局设计后的拓扑结果、显示布线的结果等等。

我们知道，一个相当规模的芯片上将含有几万至几十万个晶体管和其它元件，每层版图上的图形数以万计。在19吋或更大一些的显示屏上显示整个版图的情况，即使是拓扑的情况往往也是难以辨别设计的细节的，因此在显示时常常使用的方法是所谓"开窗"或局部的放大。所谓"开窗"就是确定用户感兴趣的版图上的显示区域，软件将自动地将该部分图案放大并显示在屏幕上。一般规定的显示区域即窗口为矩形区域，显示时可根据需要仅显示一层版图上的该部分图案，也可以显示若干层的图案以观察它们相互之间的位置关系。图形显示器可通过各种色彩或线形（如点划线、细线、粗线等）及亮度变化，使用户容易识别图形的层次。系统将提供多种开窗命令供用户选择。

窗口显示方式使用户可以方便地了解设计结果的细节和定位修改，但不利于使用户了解窗口内图形与窗口外图形的关系，因此系统将提供另一些显示命令来克服上述缺点。窗口的移动和滚动就是其中一些很有用处的命令，窗口移动命令可使显示区域从一处移到另外一处，上下左右的移动窗口可使用户了解原窗口内图形与窗口外图形的关系和连接。窗口滚动命令是使窗口以一定速度向一定方向很快地移动，使用户可扫描地观察整个设计结果而了解设计结果的全貌。当用户需要对整个设计结果作全面的观察和检查时，这种方式将是一种较适合的方式。

为了确定一些用户感兴趣的图形的实际尺寸和位置，可在图形显示器显示出相对的或绝对的坐标标尺。在有些系统中，与图形显示器配合，附加有"光笔"装置，这种笔状的装置可直接对其所指点的显示屏上位置数字坐标化，显示出来并输入计算机。

(3) 图形的修改

图形编辑的另一重要作用就是对图形进行修改，任何修改都包括以下二个过程：确定修改对象的过程；对被修改对象的修改过程。自然，当具有光笔装置时，由光笔指点的方法可方便地确定

修改的对象。有时也可利用屏幕上一个可操纵移动的十字型叉丝来确定修改对象，此时十字叉中心指点的图形就是被修改的对象。最麻烦的办法是通过显示读出被修改图形的有关位置坐标值，然后用键盘通过输入的数字来确定被修改的对象。

图形修改的种类有：删除某一图形或一组图形；移动一个图形或一组图形；对图形或图形集进行镜象变换或旋转变换；使一些图形放大或缩小，使图形的一部分延伸或收缩(图 9.10)。

在有些系统中并可具有某些自动或半自动修改功能，如当修改某一层版图的图形后，可自动地根据设计规则和工艺规则修改其它各层的相关的图形。例如当移动了某一双极晶体管的基区后，系统将自动地修改基区孔的位置及相关金属连线的位置。又如在对连线连行修改时，当指定某一线网的一个端点后，在相应的删除命令作用下可自动地删除整条连线，有时还可按一定规则进行试探性的重新连线等等。由于图形修改时将不可避免地进行较多的人工干预，因此尽量简化修改过程和实现某种意义的自动化是很有必要的。这也是在人机交互中引起人们的重视的一个问题，尤其在门阵列设计模式的剩线重布阶段，如何使计算机起到更多、更有效的辅助作用，直至实现自动的剩线处理更是人们关心的问题之一[123]。

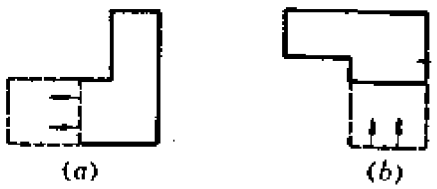


图 9.10 图形的部分延伸和收缩
(a) 图形的部分延伸； (b) 图形的部分收缩

大量的实际设计经验表明，最后结果出错误的地方往往就是人工干预的地方。因此在人工修改后，对修改相关的部分进行设计规则、工艺规则的检查(对连线来讲、进行连结性检查)都是很必要的。当把检查的范围局限于与修改相关的部分时，检查处理可具有相当高的处理效率，并可及时地对刚进行的修改给出有效的评价。

(4) 图形编辑对数据结构的要求

一般情况下 LSI/VLSI 设计得到的版图往往含有几万到几十万甚至几百万个图形。在图形的显示及修改时将反复进行数据的检索和修改，因此为了提高交互处理的效率，如何设计一个易于检索的数据结构及其管理方法是系统设计者需注意的问题。

在实际应用时，常常采取分级的层次式数据结构。芯片上最基本的单位是单元和布线通道区，单元可以是一个基本的门、移位寄存器、触发器或全加器等功能部件。系统可以按版图层次对其实际布图进行描述，由单元可以组成"块"，或模块(BLock)，若干相关的模块可组成更高一级的宏模块……，采用上述数据结构将有利于各模块的检索及修改。为了提高显示和修改的处理效率，经常放置一个动态的数据缓冲栈，存放当前参与显示、修改的数据。这样不仅可提高显示和修改的效率，而且便于实现信息的管理，只有当修改结果被认可后，才由数据缓冲栈的信息对实际版图进行真正的修改，否则恢复原设计结果也是极其方便的。

(5) 图形编辑所需的硬件

目前系统中用于图形显示的设备主要有刷新式和存贮式二种。刷新式图形显示设备可迅速显示和动态地修改显示结果中的每个图形，是一种立即响应方式的显示、修改设备。为了使显示的图像不致于闪烁，这种设备一般需保持每秒 30 次左右的刷新率，即不断地由数据动态缓冲栈向显示设备发送显示数据。当显示的图形数相当多、图案相当复杂时，则要求显示设备具有更高的性能，因此一般造价较贵。存贮式显示设备(如 4014、4015 图形显示器)利用存贮显示屏可显示高质量的图象而无闪烁，造价较便宜，但在删除、修改时不能立即响应，必须重新显示全图，因此将在一定程度上影响修改，显示的效率，不适宜于频繁的、高度的人机交互情况。

3. 图形编辑中的有关算法

一般来讲，显示、修改涉及的算法都是比较简单的，但处理效率和某些优化处理方式常常是必须考虑的。

当在显示屏上显示动态数据缓冲栈内的结果时，一个复杂的图案在缓冲栈内往往对应着一个矩形描述块的集合(有时，集合中也可含有其它基本图形的描述数据块)。如果直接显示这些数据块所描述的图形时，将在屏幕上出现许多多余的线条或重迭的部分，尤其在同时显示多层的图形时，画面将变得难以分辨（如图9.11所示)。由于一般用户都希望在每层版图上仅显示各个图案的轮廓线，因此，系统显示软件中应具有“求相关矩形集(或含更复杂的其它基本图形的情况)的包络线”的软件包[197]，在有些系统中，也称之为“去多余边”软件包和“去重迭部分”的软件包。

版图上绝大部分的基本图形为矩形，实际问题中大量的问题是图 9.11 所示情况，因此我们对这二种情况下求包络线的算法作概要的介绍。当相关图形集中含其它图形(斜矩形、圆等)时，处理原则将稍复杂一些，处理效率也将较低。

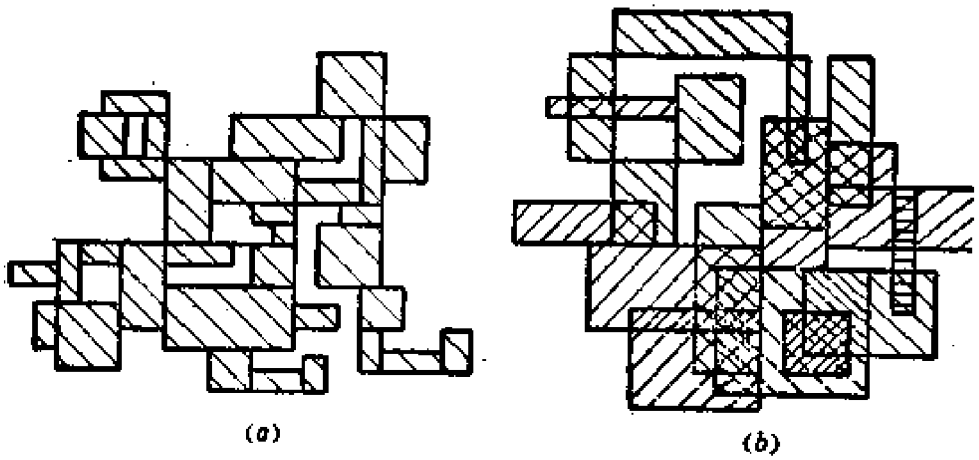


图 9.11　边相关和任意相关矩形集示意图
(a) 边相关矩形集　(b) 任意相关矩形集

(1) 求边相关矩形集的包络线算法

定义：求边相关矩形集中，各矩形水平边端点的集合记作 B 集，B 集在 $y=y_i$ 线上的子集记作 $B_{y=y_i}$ 集。

定义：所求包络线各水平边端点集合记作 A 集，A 集在 $y=y_i$ 线上的子集记作 $A_{y=y_i}$ 集。

A 集的性质：

① $A \subset B$；$A_{y=y_i} \subset B_{y=y_i}$

② 如果点(x_a, y_i)和点(x_b, y_i)满足下列条件：

$$x_a = \min x_i \quad x_i \in \{A_{y=y_i}\}$$

$$x_b = \min x_k \quad x_k \in \{A_{y=y_i} \wedge x_k \neq x_a\}$$

则以点(x_a, y_i)，(x_b, y_i) 为端点的水平线段一定是包络线的一条水平边。

根据A集的性质①，我们只要从 B 集中筛去所有不属于 A 集的元素，余下的子集就是 A 集，根据 A 集的性质②，求出 A 集也就可以求得包络线的所有水平边。

定义：$B_{y=y_i}$ 集中元素(x_s, y_i)的等值元素总数 N 为点(x_s, y_i)的相关系数。

在边相关矩形集中，所有端点的相关系数 N 是小于、等于 4 的正整数。

$B_{y=y_i}$ 集可分成 $C_{y=y_i}$ 集和 $D_{y=y_i}$ 集二个子集。其中 $C_{y=y_i}$ 是 $B_{y=y_i}$ 集中 N 为偶数的元素集合，$D_{y=y_i}$ 是 $B_{y=y_i}$ 集中 N 为奇数的元素集合。显然，

$$B_{y=y_i} = C_{y=y_i} \cup D_{y=y_i}; C_{y=y_i} \cap D_{y=y_i} = \phi。$$

在 $C_{y=y_i}$ 集中，若元素(x_A, y_i)的相关系数等于 2，则点 (x_A, y_i)的位置有下列二种情况：

a) 点(x_A, y_i)所在边和相关点 (x'_A, y_i) 所在边至少有一部分重迭，在边相关矩形集中，重迭边一定不在包络线上，因此点(x_A, y_i)也不在包络线上，即$(x_A, y_i) \notin A_{y=y_i}$。

b) 点(x_A, y_i)所在边和相关点 (x'_A, y_i) 所在边除该点外不重迭。在边相关矩形集中，这二条边实际上连成了一条边。这时点(x_A, y_i)已不是边的端点，因此$(x_A, y_i) \notin A_{y=y_i}$。

当元素相关系数等于 4 时，元素位置可以看成是第一种情况的二次出现，因此结论也是适用的。根据上述分析，$C_{y=y_i} \cap A_{y=y_i} = \phi$。根据 A 集的性质① $A_{y=y_i} \subset B_{y=y_i}$，因此 $A_{y=y_i} \subset D_{y=y_i}$。

在 $D_{y=y_i}$ 集中，若元素相关系数等于 1，则元素(x_B, y_i) 的位置有下列二种情况：

a) 点(x_B, y_i)不在其他矩形的边上。显然点(x_B, y_i)是 $A_{y=y_i}$ 集的一个元素即$(x_B, y_i) \in A_{y=y_i}$。

b) 点(x_B, y_i) 在其他一个矩形的边上（除端点外）。这时点(x_B, y_i) 一定是非重迭边的端点。在边相关矩形集中非重迭边一定是包络线的一条边。因此点$(x_B, y_i) \in A_{y=y_i}$。

当元素相关系数等于 3 时，元素位置可以看作是 $N=2$ 和 $N=1$ 二种情况的组合。筛去与该元素相关的二个点，余下的点$(x_B, y_i) \in A_{y=y_i}$ 的结论同样适用。根据上述分析可知 $D_{y=y_i} \subset A_{y=y_i}$。

前面已经证明：$A_{y=y_i} \subset D_{y=y_i}$，因此包络线端点集 $A_{y=y_i} = D_{y=y_i}$，$A = D$。

综上所述，从 B 集中筛去下列二部分元素，第一种情况中所有相关系数为偶数的元素。第二种情况中所有 $N=3$ 的元素，只留一点而筛去其他二点。余下的子集就是包络线的端点集 A，且元素相关系数皆为 1。

根据 A 集的性质②，可由 A 集求得包络线的水平边。同理，也可由 A 集求得包络线的所有的垂直边。在很多场合中，这样分段求出的包络线已能很好地满足实际工作的需要。但在有些场合需要连续地给出包络线的各部分。为此我们在下面进一步讨论包络线水平边和垂直边的连接问题。

包络线端点集 A 集 $x=x_i$ 轴上存在一个子集 $A_{x=x_i}$。根据 A 集的性质②可知，与水平边端点(x_i, y_i)相连的点(x_i, y_k) 可由下列关系式确定。

$$y_k = \max y_g,\ y_g \in \{A_{x=x_i} \wedge (y_g < y_i)\},$$

或

$$y_k = \max y_g,\ y_g \in \{A_{x=x_i} \wedge (y_g > y_i)\}。$$

决定 y_k 取上式的条件是$\{A_{x=x_i} \wedge (y_g < y_i)\}$的元素为奇数，$y_k$ 取下式的条件是$\{A_{x=x_i} \wedge (y_g > y_i)\}$的元素数为奇数。

由于 $A_{x=x_i}$ 集的元素集必定是偶数，因此除去点 (x_i, y_i) 的

$\{A_{x=x_i}\wedge(y_p>y_i)\}$和$\{A_{x=x_i}\wedge(y_p>y_i)\}$二个子集必定有一个也只有一个子集的元素数为奇数，即与(x_i, y_i)相连的点(x_i, y_k)是唯一的和可确定的。上述关系式就称为包络线端点连接原则。

这样，只要选定A集中某一元素(x_1, y_1)为起点，根据A集的性质②和端点连接原则，就可由$A_{y=y_1}$子集中求出以点(x_1, y_1)为端点的水平边和它的另一个端点(x_2, y_1)。根据端点连接原则可由$A_{x=x_2}$子集中求出以点(x_2, y_1)为端点的垂直边及另一端点(x_2, y_2)。重复上述过程，直至求出的连接点为起点(x_1, y_1)则得到了一条包络线，如果A集中还存在未连接的元素，就继续确定新的起点，找出下一条包络线，直到A集的元素全部连接完毕。

(2) 求任意相关矩形集的包络线算法

为了叙述的方便，首先让我们引入关于线段运算的概念。以点(x_a, y_i)和(x_b, y_i)为端点，且$x_a<x_b$的线段表示为$\overline{ab}$。若相关线段$\overline{ab}$和$\overline{CD}$的端点x坐标由小至大的排列为a', b', C', D'，则它们间的运算结果定义如下：

求和运算：$\overline{ab}+\overline{CD}=\overline{a'D'}$。

求交运算：$\overline{ab}\cap\overline{CD}=\overline{b'C'}$。

去交运算：$\overline{ab}-\overline{CD}=\overline{a'b'}+\overline{C'D'}$。

以(x_a, y_j)和(x_D, y_{j+1})为对角线顶点，且$x_a<x_D$的矩形表示为矩形CD。其与线段$\overline{ab}$相关的参考点(x_E, y_i)和(x_F, y_i)定义如下：

① 当$\overline{ab}$和矩形$x=x_a$的垂直边相交时，$x_E=x_a$，否则$x_E=x_b$。

② 当$\overline{ab}$和矩形$x=x_D$的垂直边相交时，$x_F=x_D$，否则$x_F=x_b$。

线段$\overline{ab}$和矩形CD的面相关(除边外)，它们间的去交运算结果定义为：$\overline{ab}-\overline{CD}=\overline{aE}+\overline{Fb}$。

以上各定义中，若线段二端点坐标相同，表示的是实际不存在的0线段，该项应从表达式中除去。

任意相关矩形集中各矩形元素的边具有下述性质：

① 不相关矩形边的端点一定是A集的元素。

② 若二个矩形的同向边相关，相关边线段和的端点才可能是A集的元素。

③ 若二个矩形的异向边相关，相关边去交后得到的线段端点才可能是A集的元素。

④ 若矩形边与其他矩形面相关，线段与矩形去交后得到的线段端点才可能是A集的元素。

根据上述性质，任意相关矩形集A集的求法可归结为：

① 所有相关的同向边分别求和。从B集中筛去不是A集元素的点。

② 所有相关的异向边分别去交。从B集中筛去不是A集元素的点。

③ 所有与矩形面相关的边分别去交。从B集中筛去不是A集元素的点，並加入新增的相关参考点。

经过上述处理，余下的线段都是不相关的(除端点外)。因此其端点集就是A集。

由A集求包络线的方法原则上与边相关矩形集的情况相同。

上述二个算法程序实现简单，且处理效率较高，其功能亦可基本满足实际需要。

(3) 图形或线段的裁剪算法

显示中另一个比较重要的算法是图形或线段的裁剪算法。在显示时，由于"窗口"内或屏幕上往往只是显示整个版图上的一个部分，有不少图形或图形的一部分将落在屏幕的"外面"，在修改、变换时，有可能变换的结果图形也会有一部分变换到屏幕"外面"去。如果不对这些落在屏幕外的图形进行处理，屏幕上实际显示的图形就可能变得杂乱无章。一种较好的方法是对送往屏幕进行显示的图形进行裁剪，使送去的图形都可在屏幕范围内显示出来而自动地舍去屏外的信息。

虽然一个封闭图形作为图形和把它看作是其边界线段组的集

合是有所不同的。如图 9.12 中，多边形 $ABCD$ 被一个矩形窗口裁剪时，作为图形裁剪的结果是 $GHIJKL$ 多边形(图中阴影区域)；而作为边界线段集合时其裁剪结果为 $\overline{GH}$, $\overline{IJ}$, $\overline{KL}$ 三条线段。但在显示时，关心的仅是图形的轮廓线，因此显示裁剪下来的线段集和图形区域效果是相同的，而处理速度可较高，程序实现也较简单。在实用系统中，广泛采用线段(图形边界)裁剪的方法。

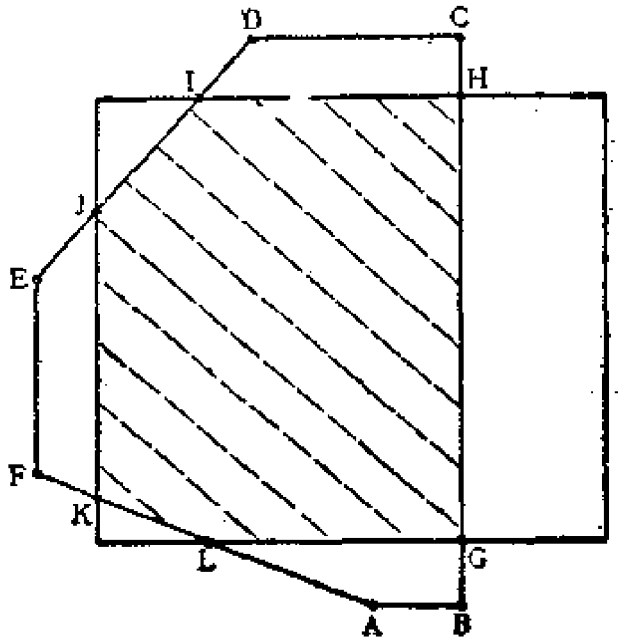


图 9.12　一个多边形被一个矩形窗口裁剪的情况

裁剪算法逻辑上是很简单的，关键在于处理效率。下面介绍二种具体的方法。

① 目前国内、外普遍使用的方法是 Ivan Suther Iand 提出的算法[187]。对每一条线段该算法分二步进行处理，第一步判断该线是否全部在窗口内或全部在窗口外而决定对该线段的取舍，第二步对有一部分落在窗口内的线段进行处理。此时将窗口边界线延长，从而把整个窗口所在平面划分为 9 个区域。对每一个区域以四位代码编号。把线段的两个端点以其所在区域代码赋以编码值。判断其与窗口边界的关系，将线段二分之，并舍去全部落在窗外的子线段。然而对余下的线段再作类似的判断，直至余下

的线段全部落在窗内为止。此方法程序实现简单，但不能避免作多余的与边界(及其延伸线)的交点计算，使裁剪一条线段最多需8次乘除法，16次加减法和40多次逻辑判断。

② 文献[207]提出了另一种多级分区判别舍弃，直接线段裁剪的算法。对给定的窗口，可尽量减少不必要的交点计算，使得裁剪一条线段最多只需4次乘除法，8次加减法和13次逻辑判断。只要经过不到6次的逻辑判断，就可以开始舍弃不落在窗口内的线段。从而提高了处理效率。

9.4 实体化处理及正确性验证

一、实体化处理的主要内容

把布图设计的拓扑描述或符号描述的结果转变为实际版图图形描述并进一步处理成版图制作的执行设备(各种图形发生器、刻图机等)控制信息的过程称作为布图设计的实体化处理过程。

实际电路布图设计的实体化处理包括基本单元或基本模块的实体化处理和连线(包括与连线相关的通孔，引出端点等)的实体化处理。在不同的设计模式中，实体化处理的内容也有所不同。在标准的门阵列模式中将主要是连线的实体化处理。

二、掩膜的制作设备

版图掩膜的制作执行设备一般采用刻图机和图形发生器。

1. 刻图机

刻图机由一个放置红膜的平台和平台上沿x轴与y轴移动的刻刀组成。通过控制刻刀的位置，刻刀的升降(抬刀和落刀)以及刻刀移动的速度可在红膜上刻出所需的图案。揭去不需要的红膜后便可得到放大几百倍的版图实际结果。刻图机控制信息描述的是实际图形的边界即线段信息。当采用刻图机制作规模很大的电路版图时，由于受平台尺寸及刻图机精度的限制需采用分区制作的

方法。利用刻图机作执行设备最大的缺点是制作工序多(刻图——揭膜——照相及一次性缩小——精缩及分步重复),其中揭膜通常需人手工操作,容易发生差错,而且一旦出现差错又很难及时检查和修正。

2. 图形发生器

广泛采用的另一种掩膜制作执行设备是图形发生器。它通常由一个产生光束或电子束的装置与一个放置干版(照像底版)并可沿 x 轴、y 轴移动的平台组成。通过控制平台的移动、定位及控制光束(电子束)的产生与否和光束(电子束)的尺寸可直接在干版上成像为掩膜所需的图形。图形发生器控制信息描述的通常是大小不同的矩形信息。表 9.1 是 Gerald 对美国 38 家公司采用台面速度为 63.5mm/秒的 EM-2000 可变光阑(光束大小可调节)图形发生器制版的统计结果。为了进一步提高图形发生器的制版效率,国内已研制成功双台面图形发生器[203]。这种图形发生器的上台面上可放置若干预先制作好的掩膜子图案。这样不仅可通过光束在干版上成像或扫描得到矩形图形,还可以通过这些上台面图形直接在干版上成像为一个复杂的子图案(一般可含 10 至几十个矩形图形)。若这些上台面图形(或称基本图形单元)选择得较合理,使它们在版图上有足够高的出现率,将可较大幅度地提高制作效率。一般可提高效率几倍,对一些重复性较好的版图,则可提高得更多[198]。

表 9.1

组别	版图图案分解成长方形的数目	版数	占百分数	工作时间(分)	占百分数	平均时间(分)
1	1,000 以下	520	42%	2.038	0.5%	4
2	1,000—20,000	675	55%	21.825	59.5%	32
3	20,000—50,000	26	2%	5643	14.0%	217
4	50,000—500,000	12	1%	10321	26.0%	860

在实体化处理时，数据的转换显然将与所选择的执行设备及其工作方式有关。同时常常需考虑如何提高执行速度等优化问题。

三、单元或基本模块的实体化处理

在布图设计开始时，单元或基本模块已采用图形描述语言、符号（符号法布图及栅阵列方式布图等）或编码（可编程逻辑阵列及唯读存贮器的布图描述等）加以定义。因此版图上单元或基本模块的实体化处理原则上是很简单的。可根据布局的结果，直接或通过简单的转换将每个单元或基本模块用它们的实际的布图来代替。

对于人工设计的基本模块进行严格的检查常常是必须的。除利用9.2节阐述过的方法，从版图上提取元件参数进行相应的电特性和逻辑特性检查外，实用时考虑到处理效率广泛使用几何设计规则检查手段。在每一个特定集成电路生产厂家和每一特定工艺中都有其相应的掩膜几何尺寸的要求或容差的要求。这些规则的集合就称之为几何设计规则。如图形的最小宽度，图形间的最小间距，各层版图间的套刻精度等，对 MOS 电路来讲设计规则还包括最小接触孔尺寸；最小沟道长度；接触孔周围 P 扩散区、薄氧化层和铝膜之间的最小容限；P 扩散区与薄氧化层之间的最小交迭；P 扩散区和薄氧化层及铝膜之间的最小交迭等等。

进行这些检查的程序实现方法往往并不复杂，但处理效率常常是人们关心的问题。在实际处理时常常采用分区处理的方法以提高处理效率。在实际应用中，几何设计规则的检查对于减少人工设计的布图差错常常是很有效的。

四、连线的实体化处理

各级模块或单元间的连线在布线时，常常用无宽度的线来表示。这种表示方法便于处理，便于修改。当需要对结果作某些优

化处理时（如通孔最小化处理等）也都很方便。在实体化处理时，需把连线由线段拓宽成实际图形，同时按工艺规则处理连线上的接触孔和引出接点(压焊点图形等)，这些处理的方法实现时比较简单。

对于人工干预、修改的有关连线，检查其连接性是否正确，如有无断路、短路、冗余器件等往往是必要的，同时进行设计规则检查也是必要的。

在版图设计完成前，设计者往往无法精确地预测寄生元件对电性能的影响。这些影响中实际连线的长度及连线上寄生电容的影响是主要的。版图设计完成后，实际连线对电路特性的影响将可以得到较精确的测定。可以利用延时关系分析软件(Timing)对电路特性进行模拟分析，有时为了提高处理效率，也可仅仅对影响电路特性的关键信号传输链或选定的其它信号传输链进行严格的分析检查，并由设计者决定最后的修改方案。值得注意的是即使全部自动设计，虽然理论上设计规则和连结性检查都将不会出现错误，但电特性检查仍可能检查出错误或"警告"。因此，对于布图设计来讲，电特性检查的方法是值得人们注意的。

五、图形数据的转换及其算法

1. 图形数据转换的必要性

当版图设计结果的实体化处理完成后，为适应制版执行设备的要求，往往需要对版图的图形数据作进一步的处理。

我们知道，对刻图机来讲，实际处理的是图形的边界(线段)信息。对于相交矩形集，刻图机刻画的是它们的包络线，因此显示中采用的求相关矩形集的包络线的算法对刻图机所需的图形数据转换也是很重要的。

对图形发生器来讲，实际处理的对象是一定大小的矩形。对于版图图形描述中存在的其他图形(多边形、圆、环等)，需把它们转换成等价的相关矩形集。大量的实际问题是把图 9.13 所示的

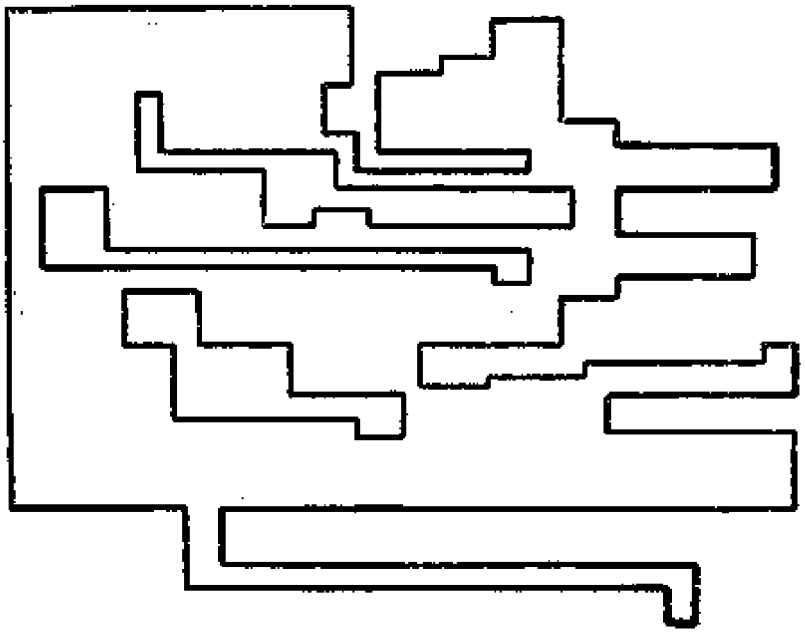


图 9.13　某一MOS 4096 RAM 版图上的实际图形

直角多边形转换成一个矩形集（该图为某一 MOS 4096 RAM 版图上的实际图形）。使该矩形集所有元素表示的域的总和等于直角多边形表示的域。考虑到制版质量和效率，应使矩形集中各元素除边界外互不重叠且元素数尽可能少[197]。

2．图形转换的图模型

一个直角多边形可以转换成不同的矩形集。对其中任一矩形集，如果把每一个矩形都用一个点来表示，用连接二点的边来表示

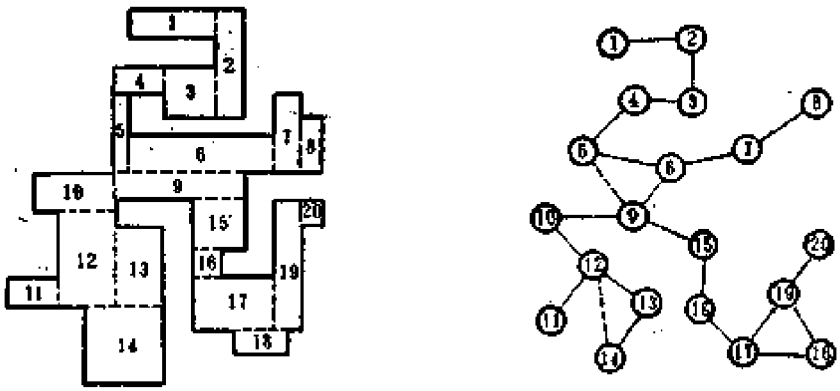


图 9.14　矩形集连通图及其映象的树

二个矩形边界上有重叠，就可得到一个表示矩形集的连通图 G。连通图 G 的不包含任何回路的最大子图就称作矩形集映象的树，如图 9.14 所示。一个矩形集映象的树不一定是唯一的，而是一个树的集合。如果树 T 是该集合的一个元素，则称树 T 为矩形集的可能树。

3．定义和性质

定义：树上一个顶点，如果它的线度等于 1，这样的顶点定义为树的端点。

端点具有下述性质：

① 一个树上至少存在一个端点。

② 把端点以及连接它的边从树上折下，余下的子图仍是一个树。

③ 连通图 G 的顶点 V，如果满足下述条件之一：

a）它的线度等于 1；

b）连接顶点 V 的所有边都是回路的组成边，且通过顶点 V 的所有回路相互间至少具有一条公共边，则顶点 V 一定是图 G 可能树上的一个端点。性质③的证明如下：

i．顶点 V 满足条件 a，结论显然是正确的。

ii．顶点 V 满足条件 b 时，由于一个回路中去掉任何一个点以及连接它的边，回路中余下的各顶点一定仍是连通的。因此如果去掉顶点 V 以及连接它的边后，各回路余下的点仍是连通的。其次，由于各回路间至少有一条公共边，即至少有二个公共顶点，因此去掉顶点 V 后，各回路余下的点相互间至少还存在一个公共顶点，即各回路所有余下的点也是连通的。这样只要恢复一条边连接 V，整个图就是一个连通图，该图中顶点 V 的线度等于 1，显然顶点 V 必然是连通图 G 可能树的一个端点。 证毕。

4．算法描述

根据端点的性质，直角多边形转换成矩形集的方法可归结为下述递归过程：

① 对直角多边形求一个矩形 R，使矩形 R 映象的点是结果矩形集可能树上的端点。

② 从直角多边形中取出矩形 R，即从树 T 上折下表示矩形 R 的端点以及连接它的边。

③ 将余下部分修改成一个新的直角多边形。

重复上述过程，直至余下部分为零，直角多边形就转换成了矩形集。

上述方法的具体实现如下：

(1) 直角多边形的描述

① 定义直角多边形包络线的逆时针方向为正向。

② 直角多边形水平边起点的坐标有序集合。用来描述直角多边形的水平边起点定义为基点。如图 9.15 所示，直角多边形可描述为：(x_0, y_0)；(x_1, y_1)；(x_2, y_2)；……；(x_{14}, y_{14})。或 (x_2, y_2)；(x_3, y_3)；(x_4, y_4)；……；(x_{14}, y_{14})；(x_0, y_0)；(x_1, y_1)。显然基点数等于多边形的边数的二分之一。

(2) 求矩形 R 的方法

① 以二个相邻基点 (x_i, y_i)、(x_{i+1}, y_{i+1}) 为对角线顶点作一个矩形，如果该矩形包络线与直角多边形包络线重迭部分方向相同，则该矩形内一定包含着直角多边形可能树上的顶点(但不一定是唯一的顶点)。如图 9.16 所示，即相邻基点即满足下述关系：

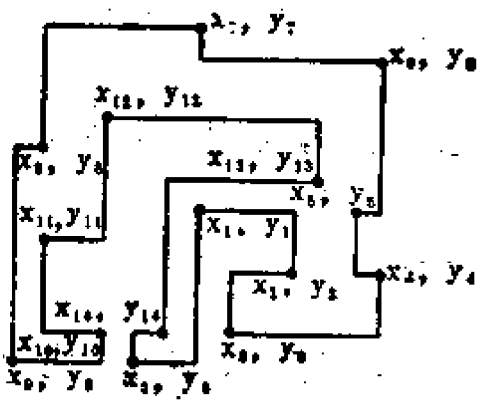


图 9.15 直角多边形的描述

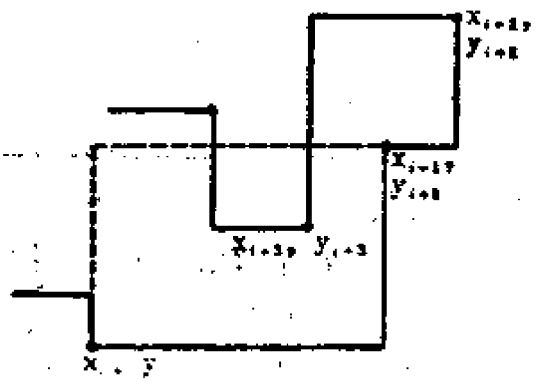


图 9.16 矩形内含多个顶点情况

$$\begin{cases} x_i > x_{i+1} \\ y_i > y_{i+1} \end{cases} \text{或} \begin{cases} x_i < x_{i+1} \\ y_i < y_{i+1} \end{cases}$$

② 点$(x_g, y_g) \in$ {基点}。如果满足 $(x_g, y_g) \notin \{(x_o, y_o) | (x_{i+1} < x_o < x_i) \wedge (y_{i+1} < y_o < y_i)\}$ 或$(x_g, y_g) \notin \{(x_o, y_o) | (x_i < x_o < x_{i+1}) \wedge (y_i < y_o < y_{i+1})\}$，则该矩形可以只是直角多边形可能树上的一个顶点。

③ 根据端点的性质3，如果该矩形同时满足：

$$(x_g, y_g) \notin \{(x_o, y_o) | [(x_{i+1} < x_o \leqslant x_i) \wedge y_o = y_{i+1}] \vee [x_o = x_i \wedge (y_{i+1} \leqslant y_o = y_i)]\}$$

或

$$(x_g, y_g) \notin \{(x_o, y_o) | [(x_i \leqslant x_o < x_{i+1}) \wedge y_o = y_{i+1}] \vee [x_o = x_i \wedge (y_i < y_o \leqslant y_{i+1})]\},$$

则该矩形映象的顶点是可能树的端点。此时以 (x_i, y_i)、(x_{i+1}, y_{i+1})为对角线顶点的矩形就是所求的矩形 R。

(3) 将余下部分修改成一个新直角多边形

去掉的矩形R与余下部分相关的边只可能存在于边 (x_i, y_{i+1})、(x_{i+1}, y_{i+1}) 和边 (x_i, y_i)、(x_i, y_{i+1})上且它们的重迭部分方向相反，因此矩形边(x_{i+1}, y_{i+1})、(x_i, y_{i+1})的终点(x_i, y_{i+1})就是新直角多边形增加的包络线水平边(唯一的)的起点。所以只要在原基点列中去掉(x_i, y_i)和(x_{i+1}, y_{i+1})二个点，而将(x_i, y_{i+1})作为新的基点置入原(x_i, y_i)点的位置就得到了新直角多边形的描述。这时原列元素数减少了一个。由上可见，修改的方法是相当方便的，而算法是收敛的。重复进行上述过程，直到基点列的元素数等于1，直角多边形就转换成了矩形集。

需要说明的是在经过修改的直角多边形中，相邻基点可能出现下述情况：

$$\begin{cases} x_i = x_{i+1} \\ y_i \neq y_{i+1} \end{cases} \text{或} \begin{cases} x_i \neq x_{i+1} \\ y_i = y_{i+1}, \end{cases}$$

这时我们只要认为它们也符合上述求矩形R的条件，取出的矩形宽度或长度等于零而不送入结果区，但同样进行修改即可。

上述算法思想简捷有效、程序实现简单，处理效率高。

可以证明[199]当直角多边形的任意二个基点都不在一条平行于数轴的直线上时，无论如何划分，结果矩形集的元素数M的最小值为：

$$M=N+Q-2$$

其中N是基点数，Q为直角多边形不相连的包络线条数。

很明显，对任何直角多边形，上述算法求得的矩形集元素数$M\leqslant N+Q-2$。因此当直角多边形任二个基点都不在一条平行于数轴的直线上时，该算法求得的矩形集是元素数最少的矩形集。

5．广义直角多边形的转换

在实际问题中常常还存在如图9.17所示的多条不相连包络线(设为Q条)所组成的广义的直角多边形。这时，我们只要适当添加$Q-1$条辅助线，使这些包络线分别地联结起来，如图9.17示。如果我们想象在这些辅助线处存在着宽度为Δ(Δ趋于0)的缺口（如图9.18）则这种添加辅助线后的多条包络线直角多边形就完全可以用上面介绍的算法统

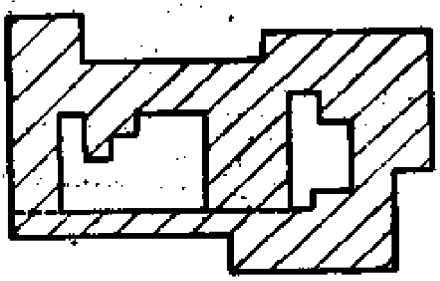


图9.17 含多条不相连包络线的直角多边形

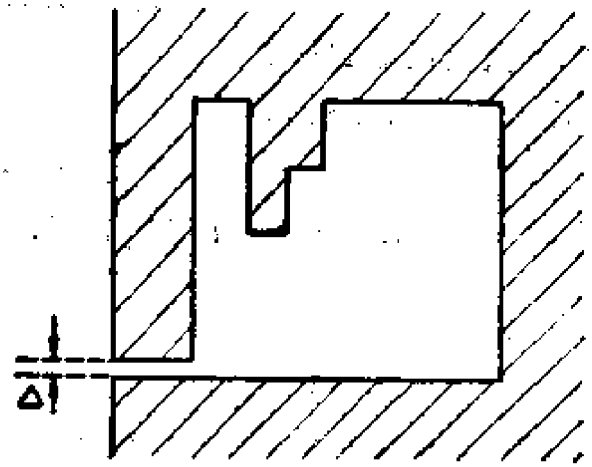


图9.18 在辅助线处存在缺口的情况

一地进行处理。

添加辅助线的处理可由软件自动实现 [197]，这时需注意的问题是：

① 广义直角多边形的外边界包络线上基点序列与各内边界上的基点序列方向应恰好相反。

② 应使添加的辅助线能连接且仅连结其中二条原不相连的包络线。同时辅助线应全部处在广义直角多边形覆盖的实域内。

③ 应使添加辅助线后增加的基点数最少。

国内近年来还发表了一些讨论如何求得直角多边形和斜多边形(多边形边界线段中含有斜线)的元素数最少或接近最少的基本图形集(矩形和斜矩形)的算法[196]，并取得了较好的结果。

在实际系统中，当算法精度影响执行设备的处理效率为主要时，应对提高算法精度所付出的降低计算机处理效率的代价与提高算法精度后提高的执行设备处理效率进行统一的考虑，目标应是使整个设计成本尽可能低。换言之，应兼顾计算机处理效率和执行设备的处理效率。

六、输出优化处理

在LSI/VLSI版图上往往存在着几万至几十万甚至上百万个图形，在利用图形发生器或刻图机等执行设备制作掩膜时，执行效率往往是必须考虑的重要问题之一。

1．图形发生器的输出优化处理

图形发生器的输出优化处理通常包括四个方面：下台面路径优化，光阑路径优化；上台面图形基本单元的优选；上台面路径优化。

(1) 优化处理的内容

上台面路径优化是指在完成掩膜制作时如何使下台面移动的总路程最短化。图形发生器在制作大小不同的图形时，可控制光阑即光束通过的狭缝的尺寸使光束在干版上成象为不同大小的矩

形以提高扫描效率。光阑路径优化就是指在完成掩膜制作时，如何使光阑尺寸改变的次数或狭缝移动的总路程最短化。在双台面图形发生器中，利用上台面上若干预先制作好的图形基本单元，可在于版上直接成象为一个复杂的图案，从而大大提高制版效率。显然，这里存在着一个要解决的问题是，如何选择一个一定元素数的图形基本单元集，使制版效率提高得较多(上台面上允许存在的图形基本单元数及图形基本单元的尺寸由具体的图形发生器设备决定)，以及如何在完成掩膜制作时，使上台面由于选择不同的图形基本单元所移动的总路程最短，也就是上台面路径优化问题。

(2) 下台面路径优化算法

在路径优化算法中，较重要的是下台面路径优化算法。

如图 9.19 所示，要制作同样宽度的 8 个矩形时，可不改变狭缝的大小，只要使下台面移动到指定位置曝光或在移动中连续曝光即可。显然矩形的不同制作顺序即下台面运动的不同路径，将使下台面移动的总路程也各不相同。在实际问题中这种差别可相当大。在图 9.19 中，对于 A、B、C 三个矩形的制作，一种方案是光束制作完 A 矩形后移动（实际台是下台面移动）到 B 矩形的左端，连续曝光完成 B 矩形的制作，然后由 B 矩形右端移动到 C 矩形右端再连续曝光完成 C 矩形。下台面非曝光移动的距离为图9.19中 1、2 线段长度之和。同时还存在着多种其他执行方案，如可沿图中 3、2 线段所示意的方案执行，但下台面移动距离将比前一方案增加。对于一个实际问题，求得下台面移动总路程最短的执行方案是一个难解的问题，上述问题当把各矩形抽象为一个点时，实际上就是典型的巡回售货员问题。

巡回售货员问题的一般描述是：给定几个城镇并已知每一对城镇之间的距离，一个售货员从 A 城镇出发，他希望访问其他每一个城市一次且仅仅一次，最后回到 A 城，使整个旅行的总路程最短。这个问题是运筹学中著名的一个难题。若穷举所有可能的路径，将有 $n!/2$ 种可能路径，显然当 n 足够大(如 $n=20$)穷举是不

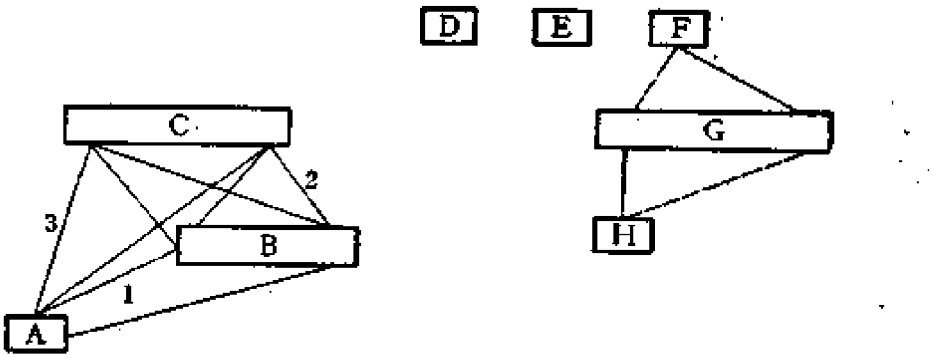


图 9.19　矩形制作示意图

可行的。采用动态规划或分支限界法可使计算复杂性有所降低，但处理效率对一般实际问题仍相当低。

很容易证明巡回售货员问题与找出一组点的最短链（不是闭回路）的问题是等价的。

由于实际版图上每个矩形具有一定的大小，并不能真正看作是一个点，而将使可能存在的路径数大大增加，进一步增加了问题求解的复杂性。此外，在实际应用中，必须对计算机处理效率和执行设备的执行效率进行统一的考虑，选择总成本最低的方案。图9.20给出了下台面路径优化问题的精度与计算机处理成本及执行设备的处理成本之间的定性关系。从图中可以看出选择 P 点附近的方案，可使总的设计成本较低或最低，因此，在实际应用时一般采用近似算法求解下台面路径优化问题。

（3）光阑路径优化和上台面路径优化

光阑路径优化和上台面路径优化比较简单。光阑路径优化问题只要使光阑从小至大或从大至小每次扫描完同宽度的矩形，即可使光阑的变化次数最少且移动的总路程最短。上台面路径优化的原则是类似的，可依一定顺序选择每一个上台面图形基本单元，每次将版图上同一图形基本单元的所有图案全部处理完毕，从而可保证上台面所走过的总路程最短。然而，上述原则往往与下台面

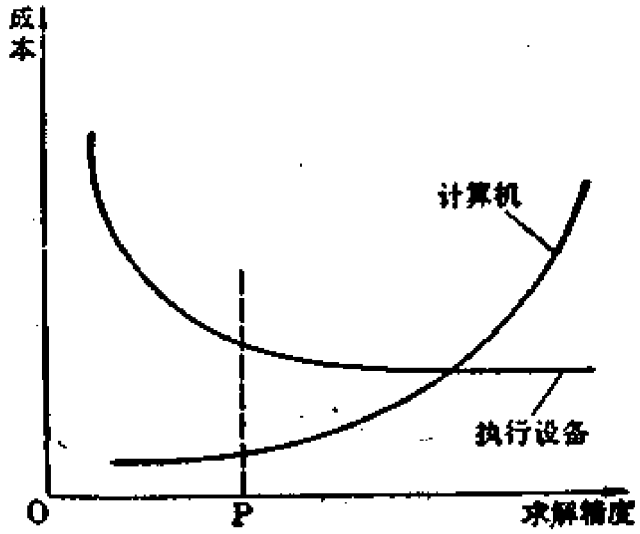


图 9.20　路径优化问题的精度与计算机和执行设备处理成本间的关系

路径优化处理原则相矛盾，以下台面路径最优化原则选择的下一个扫描图形并不是当前光阑或上台面图形基本单元所适宜做的图形。因此，在实际问题中如果光阑和上台面的运动可与下台面移动同时进行且到位速度高于下台面到位速度，则光阑路径优化和上台面路径优化问题可基本上不予考虑，否则将对它们予以适当的统一考虑。

(4) 一种实际应用的路径优化方法

① 对版图图形的分区、分类处理。考虑到光阑和上台面路径优化，可对版图图形进行预先的分类。这种预先的分类可分为三大类：

a) 由图形基本单元构成的版图图形集；

b) 由水平矩形，即矩形 y 方向宽度等于光束宽度的矩形，构成的版图图形集；

c) 由垂直矩形，即 x 方向宽度等于光束宽度的矩形，构成的版图图形集。

各集中还可以依图形基本单元种类和光束宽度分为一些相应的子集。

整个版图对不同的图形集分别进行分区处理。如对 a、b 类图

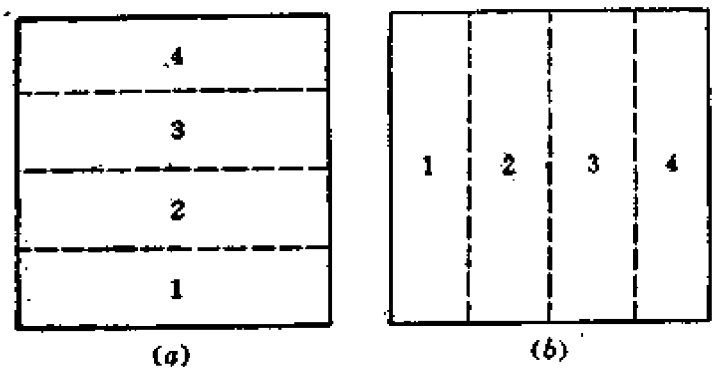


图 9.21　对版图的分划处理

形集可采用图 9.21(*a*) 所示的水平带区的分划方式，而对 c 类图形集可采用图 9.21 的垂直带区分划方式。由于LSI/VLSI版图上图形数目繁多，为提高处理效率，分区是必要的，但过细地分区将影响结果的优化，因此一般分区要考虑实际版图的尺寸，总图形数等因素由人工或自动地分别予以处理。

② 在每一区中，对每一图形子集分别用下台面路径优化的近似算法求各图形扫描的合理顺序。

一种实际采用的近似算法是从当前光束的位置出发，在所有未制作的同一图形子集中求其端点(对水平矩形来讲，考虑的是它的左、右二个端点，对垂直矩形来讲，考虑的是它的上、下二个端点）离当前光束位置最近的图形作为光束下一个扫描、曝光的图形。若在第 1 区中以光束宽度由小到大的顺序处理各图形子集，则在下一区中以光束宽度由大到小的顺序处理各图形子集。这种方法兼顾了上、下台面的路径优化处理，虽然在实际上并不总能得最短路径，但算法处理效率高并常常能产生相当满意的结果，能较好地满足实际的需要。

(5) 上台面图形基本单元的优选

在图形发生器输出优化处理中另一个问题是双台面图形发生器图形基本单元集的自动选择问题[198]。

① 基本单元集的优选目标函数。在双台面图形发生器工作

过程中，由于上台面一个基本单元在下台面上的一次曝光成象，就可等效于可变光阑或基本矩形的多次曝光成象，因而提高了制版效率。我们容易确定当采用一个基本单元后，在制版全过程中可减少的曝光次数 f_o 为：

$$f_o = n \cdot (q-1) - q$$

其中 n 为该基本单元在版图上出现的总次数，q 为基本单元图案分解成长方块的最小数目。

因此，一个上台面基本单元集在制版过程中可减少的总曝光次数 F_o 为：

$$F_o = \sum_{i=1}^{k} [n_i \cdot (q_i - 1) - q_i]$$

其中 k 为上台面基本单元数。

显然，F_o 是所选基本单元集的一个函数，不同的基本单元集将分别对应着一个 F_o 值。因此，我们可以将 F_o 作为基本单元集优选的目标函数，即对于一个指定的版图，选取的基本单元集所对应的 F_o 值越大，则认为该基本单元集越合理。

如果我们着重考虑由于采用了上台面基本单元集而在整个制版过程中所减少的下台面行程，则相应的优选目标函数 s 为：

$$s = \sum_{i=1}^{k} [q_i \cdot (n_i - 1) - n_i] \cdot \bar{d}_i$$

其中 $\bar{d}_i$ 是第 i 个基本单元的图案在版图上的平均间距。

在实际工作中，采用 F_o 作为上台面基本单元集优选的目标函数已能很好地满足需要，因此在下面的讨论中，我们将以 F_o 作为所讨论的目标函数。

② 版图单元及其分划。一般说来，无论 LSI 或 VLSI 其基础总是有限种类的基本电路，与之相应，在 LSI 或 VLSI 版图上，也可认为，存在着相应的版图单元。

定义：在版图上，矩形、斜矩形、圆、环称为图元。其形状参数和基点的定义见图 9.22

设版图上存在一个图元集 $B_1 = \{b_{11}, b_{12}, \cdots, b_{1i}\}$。其中 b_{11}，

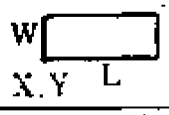
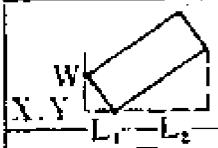
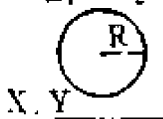
图元	形状参数	基点
	L、W	X.Y
	L1.L2.W	X.Y
	R	X.Y
	R.r	X.r

图 9.22　图元的形状参数和基点的描述

$b_{12}\cdots$, b_{1t} 分别为图元集中的一个图元，其相应的基点为x_{11}, y_{11}; x_{12}, y_{12}; $\cdots$, x_{1t}, y_{1t}。则图元集 B_1 的基点 x_1, y_1 定义为：

$$\begin{cases} x_1 = \min x_{1t},\ x_{1t} \in \{x_{11}, x_{12}, \cdots, x_{1t}\} \\ y_1 = \min y_{1t},\ y_{1t} \in \{y_{11}, y_{12}, \cdots, y_{1t}\} \end{cases}$$

图元集 B_1 存在着两个相应的参数集 D_{11} 和 D_{12}：

$D_{11} = \{F_{11}, F_{12}, \cdots, F_{1t}\}$，其中 F_{1t} 为 b_{1t} 所对应的形状参数。

$D_{12} = \{S_{11}, S_{12}, \cdots, S_{1t}\}$，其中 S_{1t} 为图元 b_{1t} 基点与图元集 B_1 基点 x_1, y_1 的相对坐标 x'_{1t}, y'_{1t}。

$$\begin{cases} x'_{1t} = x_{1t} - x_1 \\ y'_{1t} = y_{1t} - y_1 \end{cases}$$

对于版图上非图元的图形，如直角多边形，可通过图形转换[9]，转换成一个矩形图元集。

根据上述定义，若版图上存在图元集 B_1, B_2, $\cdots$, B_n，如满足下述条件：

a)　$B_i - B_k = \phi$, B_i, $B_k \in \{B_1, B_2, \cdots, B_n\}$,

b)　$\begin{cases} D_{i1} = D_{11} \\ D_{i2} = D_{12} \end{cases}$　$i = 2, 3, \cdots, n$。

则称 B_i 为该版图的一个版图单元，且在版图上出现了 n 次。

覆盖版图单元所含全部图元的最小矩形域称为版图单元的域，其矩形边界称为版图单元的边界，版图单元边界内（含边界）的任一矩形域称作版图单元的子域。

版图单元是选择上台面基本单元的对象，但上台面每个基本单元的线度受台面的限制，为一定值 L_0，因此选作基本单元的图集边界也必须限制在线度 L_0 之内。所以，当版图单元边界线度大于 L_0 时，基本单元只能是版图单元的一个子域。为此，在很多情况下，必须对版图单元进行分划。我们认为，存在着两种可行的分划模式，即密集子域分划和最小整体不均匀分划。

③ 密集子域分划。

定义：子域内所含版图单元的完整图元称为有效图元。

密集子域分划就是在版图单元边界内求子域 b_{max}，使 b_{max} 满足：

a) b_{max} 子域的边界线度 $\leqslant L_0$，

b) b_{max} 子域内的有效图元数 $q_{max} \geqslant q_i$。

$q_i \in \forall$ 版图单元线度为 L_0 的子域的有效图元数。

则 b_{max} 称为版图单元的密集子域。

显然，这种分划模式可以严格地求得版图单元有效图元的密集子域。但是这种分划模式存在两个问题：

i. 在分划过程中，子域边界对某些图元将进行切割。此时，有的图元如斜矩形、圆、环由于不允许被切割，因此不能将其在子域内的部分选入基本单元，但由于这些图元在版图单元中占的数量相当少，而且在软件中采取了一些相应措施，因此对结果的影响并不重要。而大量出现的矩形图元若被切割，将严重影响这种分划模式的结果。一个矩形图元若被子域边界切割，将被分为 2～5 个图元，子域内仅能包含其中的一个图元。此时，若将子域内包含的部分选入基本单元，由于所余部分图元数肯定 $\geqslant 1$，因此，选取这个切割的部分对减少制版过程的总曝光数或是没有贡献，或是

反而造成损失。所以，只有子域边界内未被切割的图元才对减少总曝光数是有效的。换言之，只有当子域边界切割的图元数比较少，子域内能包含足够多的有效图元时，这种分划模式才是比较适宜的模式。

ii．密集子域分划模式对一个版图单元进行一次分划只能求得一个密集子域。它不能保证以最少数量的子域覆盖版图单元的全部图元。因此，当一个版图单元由于平均图元密度比较高而可能全部选入基本单元集时，这种分划模式将需要比较多的基本单元数而降低了基本单元的平均图元数。

因此，这种分划模式适用于版图上图元平均密度较低而图元密度不均匀，且有效图元密集的版图单元的分划。

算法上可采用变步长搜索法以提高处理效率，具体实现方法从略。

④ 最小整体不均匀分划。求最少数量的子域(各子域线度$\leqslant L_0$) 覆盖版图单元的所有图元称作最小整体分划。为了提供分划方式控制的信息，在具体实现时，目标修改为求尽量少的线度$\leqslant L_0$的子域，使之覆盖版图单元的全部图元，且尽量使各子域所含图元数比较悬殊。这种分划模式称为最小整体不均匀分划。

显然，当版图单元由于平均图元密度比较高而可能全部选入基本单元集时，这种分划模式所需的基本单元数可比较少。此外，这时版图单元中所有被子域边界切割的矩形都是有效的。从而提高了基本单元的平均图元数而有利于获得较高的优选目标函数F。值。

因此，这种分划模式适用于版图上图元平均密度比较高，版图单元中图元密度比较均匀的版图单元的分划。算法上可采用双割线法。

由上述可知，存在着两种分划模式，分别适用于不同特点的版图单元。由于所涉及的“图元平均密度比较高”，“版图单元中图元密度比较均匀”等概念都是一些相对的、模糊的概念，取决于每

一个具体版图的特点和版图单元构成间的相对情况。因此，一般说来，要严格地求得 F_a 的极大值所对应的基本单元集并不是一个容易解决的问题。而关键在于如何对各个具体的版图单元决定具体的分划模式。

⑤ 最小整体不均匀分划阈值 U。我们已经了解分划方式的确定取决于版图单元的图元平均密度，以及版图单元内图元密度的均匀程度。自然，版图单元在版图上的出现次数也是影响它能否被选中的重要参数。

我们认为，用版图单元的平均优选目标函数 f_a 值可较好的描述上述诸因素的影响。

定义：决定版图单元采用最小整体不均匀分划的平均优选目标函数的下限值称为最小整体不均匀分划阈值 U。

设版图上第 j 个版图单元进行最小整体不均匀分划，求得子域数为 m_j，其中各子域所对应的优选目标函数值为 f'_{ai}，则该版图单元的平均优选目标函数值为：

$$\bar{f}_{aj} = \left(\sum_{i=1}^{m_j} f'_{ai}\right) \Big/ m_j$$

定义：P_j 为该版图单元被选中的下限权重。

$$P_j = \min\{f_{ai}\},\ f_{ai} \in \{f'_{a1}, f'_{a2}, \cdots f'_{am_j}\}$$

对一具体版图，设共有 j 个版图单元，各版图单元平均优选目标函数值的集合为：

$$A = \{\bar{f}_{a1}, \bar{f}_{a2}, \cdots, \bar{f}_{aj}\}$$

设 $U_1 = \max\{\bar{f}_{aj}\},\ \bar{f}_{aj} \in \{A\}$，

$U_2 = \max\{\bar{f}_{aj}\},\ \bar{f}_{aj} \in \{A\}$ 且 $\bar{f}_{aj} \neq U_1$，

⋮ ⋮

$U_k = \max\{\bar{f}_{aj}\},\ \bar{f}_{aj} \in \{A\}$，且 $\bar{f}_{aj} \neq U_1, \bar{f}_{aj} \neq U_2, \cdots, \bar{f}_{aj} \neq U_{k-1}$

则最小整体不均匀分划阈值 $U = U_{k+1}$，U 满足下述各条件：

$$\begin{cases} U_{k+1}=\max\{\bar{f}_{oj}\},\ \bar{f}_{oj}\in\{A\},\ \text{且}\ \bar{f}_{oj}\neq U_1,\bar{f}_{oj}\neq U_2,\cdots \\ \qquad \bar{f}_{oj}\neq U_k \\ \sum_{i=1}^{k} m_i < W-\delta \\ \left(\sum_{i=1}^{k} m_i\right)+m_{k+1}\geqslant W-\delta \end{cases}$$

其中W为上台面基本单元总数，δ为阈值参量，根据版图特点可取0到W的任何正整数，一般可定义$\delta=0$。

利用阈值U可有效地决定各版图单元应采用的分划模式。即只有满足下述各条件的版图单元，可采用最小整体不均匀分划选优。

a）该版图单元对应的$\bar{f}_{oj}$值是当前所有未分划选优的各版图单元所对应的$\bar{f}_o$值中的最大值；

b）$\bar{f}_{oj}\geqslant U_3$；

c）$P_j\geqslant U_3$；

d）$m_j\leqslant w'$，w'为当前待选基本单元数。

不满足上述条件的版图单元采用密集子域分划选优。

实验结果表明，这种分划模式的控制是相当有效的。求得的F_a值是相当理想的。

⑥ 图集的归并。版图单元所覆盖的图元集常常存在着几类不同的构成情况。设版图单元B_1，B_2覆盖的图元集为R_1和R_2。R_1由子图元集r_1，r_2，$\cdots r_i$的合集构成，R_2由子图集r'_1，r'_2，$\cdots r'_i$的合集构成。则图元集R_1和R_2可能具有下述三类构成情况：

a）构成R_1的各子图集都相同；

b）构成R_1的各子图集中有一些是相同的，同时也含有不相同的子图集；

c）R_1和R_2的各子图集中有一些是相同的，即$r'_1=r_1$，$r'_2=r_2$，$\cdots$，但也含有各不相同的子图集。

显然，在基本单元的选择时，应该对此予以足够的考虑，原则上，当相同的子图集足够大时，应进行归并处理并作为一个独立的

版图单元。算法上可采用印模算法，但这将花费大量的机时。我们在处理中，采用分类处理的方法。在第一种构成时，当版图单元边界线度大于 L_0，且所含子图集内的图元数足够多时，将子图集进行归并处理，作为新的版图单元，并同时修改其在版图上的出现次数。对于第二、三类构成，则在分划选优的过程中分别对分划得到的子域所覆盖的图元集与已选中的基本单元进行比较，判别和归并处理。这样，就提高了处理效率，提高了软件资源的利用率，其结果也是令人满意的。

⑦ 算法的基本逻辑过程。基本单元自动选择的逻辑过程如图 9.23 所示，整个选优过程在最小整体分划阈值的控制下可分为两种模式的四次选优、归并过程。在求得该版图对应的最小整体分划阈值后，系统进入最小整体不均匀分划选优的实过程。即对满足最小整体分划条件的各版图单元进行整体最小分划，并选出最佳的单元作为基本单元，同时对结果进行归并处理。当该实过程结束时，系统降低最小整体分划阈值。如令 $U=1$，系统进入最小整体分划选优的虚过程，即并不优选出新的基本单元而仅对分划结果进行可能的归并处理。在最小整体分划选优的虚过程结束后，系统进入密集子域分划选优的实过程。在这种分划模式下，选出可能最佳子域作为基本单元，同时对结果进行归并处理，直至选出整个所需的基本单元集。此后，系统进入密集子域分划选优的虚过程。再一次对密集子域分划结果进行归并处理。这样就在选优过程中较好地解决了图集的归并问题。这种处理方法虽然不能严格地实现所有的图集归并问题，但可以相当高的效率和软资源利用率得到令人满意的结果。

在上述算法中影响处理效率的主要部分是版图单元在密集子域分划模式下的虚、实过程。其计算复杂性为 $O(q_i^2)$，q_i 为 i 号版图单元的图元数。由于版图单元所含图元数不会很大，因此处理效率比较高。以 MOS 16 K RAM 电路为例，对整套版图进行基本单元自动选择处理，在 DJS-130 机上所费机时为 15 分钟左右，

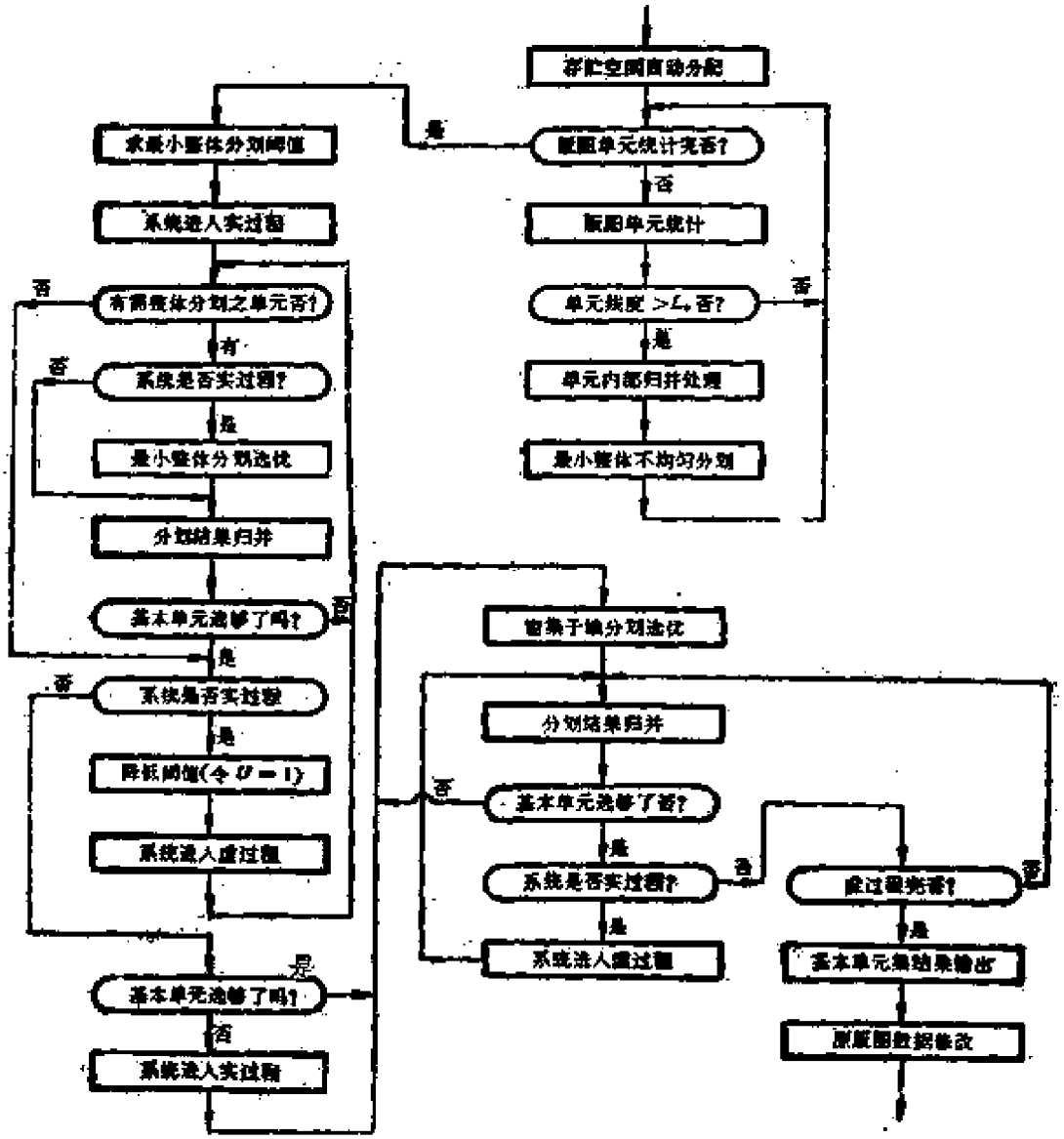


图 9.23　基本单元自动选择的逻辑过程

其选择结果可与人工选择的结果相当。

2. 刻图机的输出优化处理

刻图机的输出优化处理比较简单，它的输出优化问题可描述为如何在刻制版图上所有图形（也可理解为刻制所有图形的边界线)时，使刻刀移动的总路程最短。从理论上来讲这也是一个类似"巡回售货员问题"的问题，也必须在实际处理时采取有效的近似算法。此时首先利用 9.3 节中阐述的求相关矩形集包络线的算法

去除图形中的“多余边”及“多余部分”。并直接利用上述算法的结果，把各图形的边界线段自然地分成水平线段集和垂直线段集，在每个线段集中以其 y 坐标或 x 坐标分成若干子集（相同 y 坐标的水平线段子集及相同 x 坐标的垂直线段子集）。

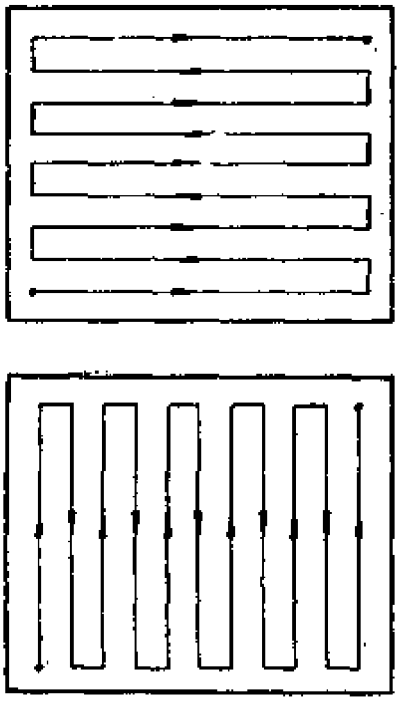


图 9.24 刻刀所走路径图

刻图操作时，先刻水平线集，然后转换刻刀方向刻垂直线集。

在刻水平线集(垂直线集)时，先刻 $y(x)$ 坐标小的线段子集。

在刻画每一个水平线段（垂直线段）子集时，每一次依各线段左端点（下端点）x 坐标（y 坐标）由小到大的顺序进行刻画，下一次则以各线段右(上)端点 $x(y)$ 坐标由大到小的顺序刻画。刻刀所走的路径如图 9.24 所示的 S 型。

9.5 本章小结

本章概要地介绍了组成布图设计系统的硬件配置以及系统中一些有关的处理过程，并着重地介绍了其中的一些算法。

从设计问题的提出到电路生产用整套掩膜的完成，整个设计过程涉及了如此之多的问题及解决这些问题的算法。对于其中的几乎每一个问题都可以进行独立的深入的而有意义的研究，但对一个实际系统来讲，它们是相互联系的统一的整体。从系统的角度出发更应该注意它们彼此的联系和影响，而在算法作用、精度和效率上作统一的考虑。

由于 LSI/VLSI 布图设计问题的复杂性和对象的多样性，在实际设计中将集成电路设计者的智慧和以算法体现的已物化了的

软件设计者的智慧结合起来往往是必要的。不断地把集成电路设计者的经验和智慧算法化，并和不断地发展更方便、更灵巧的人机交互系统，都是布图设计自动化向前发展的重要课题。

对于一个实际系统来讲另一个极重要的问题是设计正确性验证问题。这里如何严格地验证设计结果和如何提高验证的效率都是人们关心的问题。

参考文献

[1] C.A.Mead and L.A.Conway, "Introduction to VLSI Systems", Addison-Wesley, 1980。

[2] M. R. Garey and D.S.Johnson, "Computers and Intractability A Guide to the Theory of Np-Completeness", W.H.Freeman and Company, 1979。

[3] J. Soukup, "Circuit Layout", Proc. of the IEEE, Vol. 69, No.10, October, 1981。

[4] T.Kozawa, et. al, "Advanced LILAC—An Automated Layout Generation System for MOS/LSI" proc. 11th D. A. Conf.,pp. 26—46, 1974。

[5] G.Persky et. al, "LTX-A Minicomputer-Based System for Automated LSI Layout" J. of Design Automation and Fault-Tolerant Computing, Vol. 1, No. 3, pp. 217, May, 1977。

[6] K.W.Koller and U. Lavther, "The Siemens-AVESTA-System for Computer-Aided Design of Mos-standard Cell Circuits" proc. 14th D.A.Conf., pp. 153—157, 1977。

[7] H.Beke and W. Sensen, "Calmos, A Portable Softward System for the Automatic and Interactive Layout of MOS/LSI" proc. 16th D.A. conf., pp. 102—108, 1979。

[8] K. Sato, et. al, "MILD-A Cell-Based Layout System for MOS-LSI" proc. 18th D. A. conf., pp. 828—836, 1981。

[9] G. Persky et. al, "The HUGHES Automated Layout

System—Automated LSI/VLSI Layout Based on Channel Routing" proc. D. A. conf., (18th) pp. 22-29, 1981。

[10] C. Tanaka et. al, "An Integrated Computer Aided Design System for Gate Array Masterslices" part 2 the Layout Design System MARS-M3 18th D. A. conf., pp. 812。

[11] R.H.Khokhani and A.M.Patel, "The Chip Layout Problem, A Placement Procedure for LSI" 1977 14th D. A. conf., pp. 291-297 IBM。

[12] K.A.Chen et. al, "The Chip Layout Problem, An Automatic Wiring Procedure" 1977, 14th D.A.conf., pp. 298—302, IBM。

[13] R.F. Woodward, "Operational Aspects of Design for the IBM 3081", proc. 19th D. A. conf., pp. 91—95, 1982。

[14] Y. Horiba et. al, "A Bipolar 2500-Gates Subnano-Second Masterslice LSI" IEEE International Solid-state Circuits conf., 1981 pp. 228-229。

[15] T. Chiba, et. al, "SHARPS, A Hierarchical Layotu System for VLSI" 18th D.A.conf., pp.820 1981。

[16] Chi-Song Horng, et.al, "AN Automatic/Interactive Layout Planning System for Arbitrarily-Sized Rectangular Building Bloks" 18th D.A.conf.,1981,pp. 293。

[17] R. Malladi et.al, "Automatic Placement of Rectangular Blocks With the Interconnection Channels" 18th D. A. conf., 1981, pp. 419。

[18] H. Fleisher, et al, "Introduction to Array Iogic"IBM J. Res. Develop. Vol. 19, 1975, pp. 98。

[19] J.F.Paillotin, "Optimization of the PLA AREA" 18th D. A. conf., pp. 407, 1981.

[20] I. Suwa et. al, "A Computer-Aided-Design System for Segmented-folded PLA Macro-Cells"" proc. D. A. conf.(18th) pp. 398, 1981.

[21] Williams. J., "STICKS-A Graphical Compiler for High Level LSI Design" proc.of the 1978 Ncc May 1978, pp. 289-295.

[22] Hsveh, M-Y.: "Symbolic Layout and Compaction of Integrated Circuits," 1979 Memorandum No. UCB/ ERL, M79/80 University of California Berkeley Dec.

[23] Neil West, "Virtual Grid Symbolic Layout" 18th D. A. conf., 1981, pp. 225.

[24] Lopez,A.D, et. al, "A Dense Gate Matrix Layout Style for MOS LSI" IEEE Journal of Solid State Circuits, Vol. SC—15, No.4, Aug. 1980, pp.736-740.

[25] Johannsen, D., "Bristle-Block "Proceedings of the 16th D.A.conf., San Diego, 1979, pp. 310-313.

[26] B. T. Preos, et.al, "Methods for Hierarchical Automatic Layout of Custom LSI Circuit Masks" proc 15th D.A. conf.,1978, pp. 206-212.

[27] W.M. Van Cleemput, "An Hierarchical Language for Structural Description of Digital Systems" proc. 14th D. A. conf., 1977, pp.377-385.

[28] M. A. Breuer (Ed.), "Design Automation of Digital Systems" Vol. 1, Theory and Techniques. Chapters 4, 5 and 6 Printice-Hall, 1972.

[29] H.R. Charneg and D. L. Plato, " Efficient Partitioning of Components" proc. 5th Annual Design Autom-

ation Workshop, July, 1968。

[30] M. Hanan: "On Steiner's Problem with Rectilinear Distance"SIAM J. Appl. Math Vol. 14 1966, pp. 255—265。

[31] D. G. Schweikert et. al; "A Proper Model for Partitioning of Electrical Circuits" proc. 9th Design Automation Work shop. Dallas 1972, pp. 57-62。

[32] M. Hanan and J. M. Kurtzberg; "Placement Techniques"Chap. 5 us Design Automation of Digital Systems Vol. 1, 1972, pp.213-382。

[33] J. M. Kurtzberg: "Backboard Wiring Algorithms for the Placement and Connection Order Problems" Burroughs Report TR 60-40, June 28 1960。

[34] J.M.Kurtzberg: "Algorithms for Backplace Formation" in Microelectronico in Large Systems" Spartan Books, 1965 pp. 51-76。

[35] 近藤: "実装設計にわせる設計自動化の検讨"1970, ‹研究実用化报告›pp.23-27。

[36] S. Goto: "A Two-Dimensional Placement Algorithm for the Master-Slice LSI Layout Problem" 1979 16th D. A. conf., pp.11-17。

[37] D.M. Schvler and E.G. Ulrich; "Clustering and Linear Placement" proc. 9th D. A. workshop, pp. 50-56, 1972。

[38] K. W. Koller: "The Siemens-AVESTA-Systems for Computer-Aided Design of MOS Standard Cell Circuits" 1977, 14th D. A. conf.,pp.153-157。

[39] Lemke, C. E. et. al: "Direct Search Algorithms for Zero-One and Mixed-Integer Programming" operations Res. 15, 1967, pp. 892-914。

[40] H.W. Kohn; "The Aungarian Method for the Assignment Problem" Nav. Res. Log. Quart., Vol. 11, 1955, pp.83-97。

[41] J.Munkres; "Algorithms for the Assignment and Transportation Problems" J. SIAM, Vol. 5 1957, pp. 32-38。

[42] S. GoTo and E. S. Kuh; "An Approach to the Two-Dimensional Placement Problem in Circuit Layout" 1978, IEEE trans. on Circuit and Systems, Vol. CAS-25, No. 4, Apr。

[43] B. W. Kernighan et. al; "An Efficient Heuristic Procedure for Partitioning Graphs"Bell Syst. Tech. J. O, Vol.49, pp. 291-307, 1970。

[44] N. Christofides et. al; "The Optimal Partitioning of Graphs" SIAM J. Appl. Nath., Vol. 30, No. 1, Jan, 1976。

[45] R. M. Karp, "Reducibility Among Combinational Problems" in Complexity of Computer Computations R. E. Miller and J. W. Thatcher Eds, New York Plenum Press, 1972。

[46] K. J. Loosemore; "Automated Layout of Intergrated Circuits" in proc. 1979 IEEE Int. Symp. Circuits and Systems, pp.665-668。

[47] 程可行, 庄文君; "LSI/VLSI "双边单元的自动布局算法" «半导体学报», 1984 年, 第四期。

[48] M. Hanan. et.al; "Some Experimental Results on Placemeut Techniques" in proc. 13th D.A. conf., pp. 214-244, 1976。

[49] 上海计算技术研究所; "电子计算机算法手册"。

[50] J.M. Kurtzberg: "An Approximation Method for the Assignemt Problem" J.ACM. Vol. 9.No. 4, 1962, pp. 419-439。

[51] F. S. Hillier. et.al: "Quadratic Assignment Problem Algorithmas and the Location of Indivisible Facilities" Management Science, Vol. 13, No. 1, 1966, pp. 42-57。

[52] J.D.C. Little et. al; "An Algorithm for the Traveling Salesman Problem."J. oper. Res, Vol. 11, 1963, pp. 972-989。

[53] Bourgeois, F. et. al: "An Extension of the Munkres Algorithm for Assignment Problem to Rectangular Matrices" CACM, 14(1971), No. 12。

[54] R.H.Glaser, "A Quasi-Simplex Method for Designing Suboptimal Packages for Electronic Building Blocks" proc. 1959 Computer Appl. Symp., at Armour Research Foundation, Illinois Institute of Technology, pp. 100-111。

[55] R.G. Gamblin et. al: "Permutation Procedure for the Backboar dWiring Problem"proc.IEE(January 1968), pp. 27-30。

[56] 沙露、唐璞山："一种多元胞的自动布局算法"《半导体学报》, 1984, 第三期。

[57] M.Brever: "Min-Cut Placement" J. Des. Auto. Fault Tolerant comput.Vol. 1,no.4,pp."343—362 1977,Oct。

[58] B. T. Preas and C. W. Gwyn, "Methods for Hierarchical Automatic Layout of Custom LSI Circuit Masks" 1978, 15th D.A. conf., pp.206-212。

[59] H. Kato et. al: "An Automated wire Routing for Bu-

ilding-Block LSI" proc. ISCAS, pp.309 1974 April。

[60] R. Malladi et.al,"Automatic Placement of Rectangular Blocks with the Interconnection Channels" 1981, D. A., pp.419。

[61] G. Persky, "PDR-An Automated String Placement Program for Polycell Layout" 1976. 13 th D. A. conf., pp.417-424。

[62] D.N.Devtsch, G. Persky et.al, "LTX-A System for the Directed Automatic Design of LSI Circuits"1976,13th D. A. conf., pp.399-407。

[63] M. Hanan and J.M. Kurtzberg,"Force-Vector Placement Techniques" IBM Report RC 2843, April 1970。

[64] C.J.Fisk et. al, "ACCEL, Automated Circuit Card Etching Layout" proc. IEEE, Vol. 55, No. 11, (1967), pp. 1971-1982。

[65] L. Steinberg, "The Blackboard Wiring Problem, A Placement Algorithm" SIAM Reviow, Vol. 3, No.1, pp.37-50, 1961。

[66] R. A. Rutman, "An Algorithm For Placement of Interconnected Elements Based on Minimum Wire Length" proc. SJCC(1964), pp.477-491。

[67] S. Goto et. al, "Sub-Optimum Solution of the Back-Board Ordering with Channel Capacity Constraint" in proc. 10th Anrue. Asilomar conf., on Circuits. Systems and Computers, Nor. 1976, pp.267。

[68] P. C. G. Imore, "Optimal and Suboptimal Algorithm for the Quadratic Assignment Problem" J.SIAM Vol. 10, No. 2, pp. 305-313, 1962。

[69] E. L. Lewler, "The Quadratic Assignment Problem"

Management Sci., Vol.9, pp.586–599, 1963。

[70] J. W. Gavett and N. V. Plyter: "The Optimal Assignment of Facilities to Locations by Branch and Bound", 1966, opns. Res. Vol. 14, pp. 210–232。

[71] D. C. Wilson, R. J. Smith: "An Experimental Comparision of Force Directed Placement Techniques"1974. 11th D.A. conf., pp. 194–199。

[72] M.Hanan et. al; "Some Experimental Results on Placement Techniques" 1976, 13th D. A. conf., pp. 214–224。

[73] D.R. Johnson: "PC Board Layout Techniques", 1979, 16th D.A. conf., pp. 337–343。

[74] I. Nishioka: "An Approach to Gate Assignment and Module Placement for Pointed Wiring Boards", 1978, 15th D.A. conf., pp.60–63。

[75] E. P.Stabler et.al: "Placement Algorithm by Partitioning for Optimum Rectangular Placement" 1979, D. A.conf, pp.24–25。

[76] U. Lavther: "A Min-cut Placement Algorithm for General Cell Assemblies Based on Graph Representation" 1979, 16th D. A. conf., pp. 1–10。

[77] B. T. Preas and W.M. Vancleemput; "Placement Algorithm for Arbitrarily Shaped Blocks" 1979, 16th D. A. conf., pp. 474–481。

[78] 周电、唐璞山："LSI 二维布局的分析算法", 《半导体学报》, Vol. 5, No.4, July, 1984, pp. 396–403。

[79] K. H. Khokhani et. al; "Placement of Variable Size Circuits on LSI Masterslices" 1981, 18th D. A. conf". pp. 426–434。

[80] C.Y.Lee: "An Algorithm for Connections and Its Applications"1961, LRE, trans on Electronic computers, pp.346-365。

[81] 林守勋,庄文君: "LSI 一层半模型自动布线算法" 《计算机研究与发展》, 1984, 第五期。

[82] J. Vintr: J. Vintr is the inventor of the algorithm that was reported by R.Dutta at the 1980 CANDE workshop on hard ware for CAD, UNIV. Michigan. Ann Arbor. There were no proceedings published。

[83] F. RUBIN: "The Lee Connection Algorithm" IEEE trans Comput Vol.C-23, pp.907-914, 1974。

[84] J. Soukup: "Fast Maze Router" 1978, in proc. 15th D. A. conf., pp.100-102。

[85] R.K. Korn "An Efficient Variable Cost Maze Router" 1982, D.A. conf., pp.425-431。

[86] S.Futagam: "An Automatic Routing System for Single-Layer Printed Wiring Boards" 1982, IEEE CAS Vol. 29, No.1, pp. 46-51。

[87] P. P. Chavdhori: "An Ecological Approach to wire Routing"1979,IEEE Int. Symp. Circuits and Systems, pp. 854-857, July.

[88] M. Gardner: "Mathematical Games" Scientific American, November,1963。

[89] J.M.Geyer: "Connection Routing Algorithm for Printed Circuit Board" IEEE TCT, Vol. C-20 January, 1971。

[90] G.B.Dantzig et. al: "Solution of a Large-Scale Traveling Salesman Problem" J. oper. Res, Vol.2, (1954), pp.394-410。

[91] J.B. Kruskal, Jr. ,"On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem" proc. Amer. Math. Soe., Vol. 7(1965), pp.48-50。

[92] H. Loberman et.al,"Formal Procedures for Connecting Terminals with a Minimum Total Wire Length" J. ACM, Vol. 4(October, 1957)pp. 428-437。

[93] W. Miehle, "Link-Length Minimization in Networks" J. Oper. Res.,Vol. 6(1958), pp. 232-243。

[94] R. Courant, "What is Mathematics?" London and New York, Oxford Univerity Press,1941。

[95] E. N. Gilbert, et. al, "Steiner Minimal Trees" SIAM J.Math., Vol. 16(1968), pp.1-29。

[96] S. L. Hakimi, "Steiner's Problem in Graphs and its Implications" Networks 1, 1971, pp.113-133。

[97] F. K. Hwang,"On Steiner Minimal Trees with Rectilinear Distance" SIAM. J. Appl. Math., 30, 1976, pp.104-114。

[98] M. Hanan, "On Steiner's Problem with Rectilinear Distance"SIAM J. Appl. Math.,Vol. 14, No. 2, 1966, pp. 255-265。

[99] D. W. Hightower,"A Solution to line-Routing Problems on the Continuous Plane" 1969, D. A. workshop, pp.1-24。

[100] W. Heyns et. al, "A Line Expansion Algorithm for the General Routing Problem with a Guran-teed Solution" 1980 17th D. A. conf., pp. 243-249。

[101] 熊继光："以波的传播和绕射规律为理论基础的自动布线算法系列"《第二届半导体集成电路计算机辅助设计会议论文》1983。

[102] J. Soukop: "Global Router" 1979, 16th D. A. conf., pp. 481-484。

[103] 庄文君、王守觉："一种基于总体考虑的布线方法"王守觉赴美讲学报告，1984，（已经在1985，ICCAS-85 刊物上发表）。

[104] A. Hashimots and J. Stevens: "Wire Routing by Optimizing Channel Assignment within Large Apertures" 1971, SHARE. AC M. IEEE D. A. workshop, pp. 155-169。

[105] 杨瑞元："双面板布线分层与借孔压缩的图论方法"《第二届计算机及数字系统 CAD 学术会议论文》1982。

[106] K. R. Stevens et. al: "Implementation of an Interactive Printed Circuit Design System" 15th 1978 D. A. conf.,pp.74-81。

[107] RUEN-WU CHEN et.al: "A Graph-Theoretic Via Minimization Algorithm for Two-Layer Printed Circuit Boards" IEEE TRANS. on Circuits and Systems Vol. CAS-30, No.5 May 1983。

[108] H. Bothnger: "A Mature DA System for PC Layout" 1979 in proc. Ist Int. P.C. conf.,pp.85-99。

[109] W. A. Dees: "Automated Rip-up and Reroute Techniques" 1982, 19th D. A. conf., pp.432-439。

[110] W.R.Heller et.al: "Predilection of Wiring Space Requirements for LSI"。

[111] Won Kim: "Relational Database System" Acm Computing Surveys, Vol. 11, No. 3, September 1979。

[112] M. Wiesel et. al: "An Efficient Channel Model for Building Block LSI", proc. IEEE Int. Symp. Circuit and Systems, pp.118-121, Apr. 1981。

[113] 顾元："分级布线"，《计算机研究和发展》Vol. 20, No.6, 1983, pp.42。

[114] K. A. Chen et. al: "The Chip Layout Problem: An Automatic Wiring Procedure"Proc. D.A.conf. 14th, 1977, pp. 298-302。

[115] D.T. Lee et. al: "Number of Vias: A Control Parameter for Global Wiring of High-Density Chips",IBM J. Res. Develop., Vol. 25, No.4, pp. 261-271, 1981。

[116] J. H. Lee et.al: "Use of Steiner's Problem in Suboptimal Routing in Rectilinear Metrix", IEEE Trans. on CAS, Vol, CAS-23, No. 7, July, 1976, pp. 470-476。

[117] B.S. Ting et. al: "Routing Techniques for Gate Array", IEEE Trans. on Computer-Aided Design, Vol. CAD-2, No. 4, October, 1983, pp. 310。

[118] N.Nan and M. Fener: "A Method for Automatic Wiring of LSI Chips" IEEE Int. Symp. Circuits and Systems, 1978, pp.11-15。

[119] H. Voshizawa, et.al: "Automatic Layout Algorithms for Master Slice LSI" IEEE Int. Symp.on Cirvuits and Systems, 1979, pp. 470。

[120] Koji Sato et.al, "MIRAGE-A Simple Model Routing Program for the Hierarchical Layout Design of IC Masks", proc. of 16th D.A. conf., 1979, pp. 297。

[121] J.Soukup and J.C.Royle, "On Hierarchical Routing", J. Digital Systems, Vol. 5, No. 3, Sept.,1981。

[122] K. Kani et.al, "ROBIN, A Building Block LSI Routing Program"proc.ISCAS, 1976。

[123] 陈绍群、刘美轮、庄文君："一种新的四边通道区布线算法"

(已在 1985 ICCAS-85 发表)。

[124] H.Kawanishi et.al, "A Routing Method of Building Block LSI", Presented at the Rec. 7th Asilomar Conf. on Circuits, Systems and Computer,1973。

[125] K.W.Koller, "The Siemens-AVASTA-System for Computer-Aided Design of Mos-Standard Cell Circuits" proc. 14th D.A.conf.,1977, pp.153-157。

[126] G.Persky et.al, "The Hughes Automated Layout System-Automated LSI/VISI Layout Based on Channel Routing",proc. 18th D.A.conf.,1981, pp.22-29。

[127] M. Terai et. al, "A Consideration of the Number of Horizontal Grids Used in the Routing of A Master-slice Layout" Proc.19th D.A. conf.,1982, pp.121-128。

[128] S.Tsukiyama et.al, "A New Global Routing for Gate Array LSI"IEEE Trans.on Computer-Aided Design, Vol. CAD-2, No. 4, 1983。

[129] B.W.Kernighan et.al, "An Optimum Channel-Routing Algorithm for Polycell Layout of Integrated Circuit", Proc. 10th. D. A. conf., 1973, pp.50-60。

[130] D. Deutsch, "A Dogleg Channel Router", Proc. 13th D.A.conf.,1976, pp.425-433。

[131] M.Burstein and R.Pelavin, "Hierarchcal Wire Routing". IEEE Trans. on Computer-Aided Design, Vol. CAD-2, No.4, 1983。

[132] 庄文君："不规则边界通道区的布线算法"，半导体所科研报告,1983。

[133] 刘美轮、陈永康："Double Layer Channel Routing with Irregular Boundary""《天津大学学报》,1983 No.2。

[134] 胡祖增："最优崎岖通道布线算法"，全国第二届数字系统

D.A 会议论文，1982。
[135] 陈永康等："不等距通道布线的有效算法"，《清华大学学报》，1983。
[136] K.Sato et.al, "A'Grid-Free' Channel Router", Proc. 17th D.A.conf.,1980, pp.22-31。
[137] Y.K.Chen and M.L.Lin, "Three-Layer Channel Routing", IEEE Trans.on CAD/CAS Vol.3, No. 2, April 1984。
[138] W.Heyns, "The 1-2-3 Routing Algorithm or the Single Channel 2-step Routing on 3 Interconnection Layers",Proc. 19th D.A.conf.,1982, pp.113。
[139] M.Wiesel, "Two-Dimensional Channel Routing and Intersection Problems",Proc.19th D.A. Conf., 1982, pp. 733。
[140] Chi-Ping Hsu, "A New Two-Dimensional Routing Algorithm",Proc, 19th D.A.conf.,1982, pp. 46-50。
[141] Y.K. Chen, "An Approach to Two-Dimensional Channel Routing"UCB/ERL M81/83, Nov.1981。
[142] M.M. Sadowska and E. S. Kuh, "A New Approach to Channel Routing"IEEE Int.Symp.Circuits and Systems,1982, pp.764-767。
[143] M. R. Garey et.al, "Computer and Intactability-A Guide to the Theory of Np-Completeness", W.H. Freeman and Company,1979。
[144] D.S.Johnson, "The Np-Completeness Column, An Ongoing Guide"J. Algorithms, Vol.3, pp. 381-395, 1983。
[145] A.Hashimoto and J.Stevens, "Wire Routing by Optimizing Channel Assignment Within Large Apertu-

res",Proc.8th D.A.Workshop,1971, pp.155–159。

[146] J.Soukup and J. Fournier: "Pattern Router", Proc. 1978 ISCAS,pp.486–489, 1979。

[147] T.Kawamoto et.al: "The Minimum Width Routing of A 2-Row 2-Layer Polycell-Layout", Proc. 16th D. A. Conf.,1979, pp.290–296。

[148] D. W. Hightower: "A Generalized Channel Router", Proc.of 17th D.A.conf.,1980, pp.12–21。

[149] M. M. Wada: "An Optimal Dogleg Channel Router with Completion Enhancements" Proc. 18th D. A. conf., 1981, pp. 762。

[150] T.Yoshimura and E. S. Kuh: "Efficient Algorithms for Channel Routing", IEEE Trans. on CAD of IC and Systems,Vol.CAD-1, No.1, 1982, pp.25–35。

[151] 庄文君："通道区布线的通道损益分析法"《计算机学报》, Vol.7, No.3, 1984, pp.217-227。

[152] P. I. Jennings et.al,"A Highly Routable ULM Gate Array and Its Automated Customization", IEEE Trans on CAD of IC and Systems, Vol. CAD-3, No. 1, January 1984。

[153] 胡祖增："快速最优通道布线算法",《半导体学报》, Vol. 5, No. 4, 1984。

[154] K.J. Supowit,"A Minimun-Impact Routing Algorithm", Paoc. 19th D. A. Conf., 1982, pp.104–112。

[155] Y.Kajitani,"Order of Channels for Safe Routing and Optimal Compaction of Routing Area",IEEE Trans. on CAD, Vol. CAD-2, No. 4, October 1983。

[156] M. F. Oakes: "The Complete VLSI Design System", Proc.16th D.A.Conf., 1979, pp.452-460。

[157] C.A.Collins, "IBM 3081 System Overview and Technology", Proc. 19th D. A.conf., 1982, pp.75–82。

[158] T.Matsuda et.al, "LAMBDA: A Quick, Low Cost Layout Design System for Masterslice LSI'S", Proc。19th D.A. conf., 1982, pp.802–808。

[159] T.Adach et.al, "Hierarchical Top-Down Layout Design Method for VLSI Chip," Proc. 19th D. A. conf., 1982,pp.785–791。

[160] 庄文君、程可行："双边单元的 LSI/VLSI 布图设计算法系统"，中国科学院半导体所科研报告，1984（已在 1985 ICCAS-85 发表）。

[161] J. E. Hassett, "Automated Layout in ASHLAR: An Approach to the Problem of 'General Cell' Layout for VLSI", Proc.19th D.A.Conf.,1982, pp.777–784。

[162] H. Shiraish et.al, "ICAD/PCB: Intergrate Computer Adied Design System for Printed Circuit Boards", Proc.19th D.A.Conf.,1982, pp.727–732。

[163] J.L.Sanborn, "Evolution of the Engineering Design System Data Base", Proc.19th D. A. Conf., 1982, pp. 214。

[164] H.Beke et.al, "CALMOS, A Portable Software for the Automatic and Interactive Layout of MOS/LSI", Proc.16th D.A.Conf.,1979, pp.102–108。

[165] L.W.Nagel, "SPICE 2, A Computer Program to Simulate Semicondutor Circuits", University of California Berkeley Electronics Reseach Laboratory Report SPICE No. ERL–M520, May 1975。

[166] "Advanced Statistical Analysis Program (ASTAP)", Progran Ref. Manual, Pub. No, SH20–1118–0, IBM

Corp.Data Proc.Div.,White P/ains,Ny。

[167] A.R.Newton, "Technigues for the Simulation of LSI Circuits",IEEE Trans. on Circuits Syst. Vol.CAS-26, No.9, Sept.1979, pp.741。

[168] L.W.Nagel, "ADVICE FOR Circuit Simulation",IEEE Int.Symp.Circuits and Systems Houston,TX.Apr. 1980。

[169] R.R.Chawla,et.al, "MOTIS-An MOS Timing Simulator",IEEE Trans. Ciruits Syst.Vol. CAS-22, No. 12, Dec.1975, pp.901-910。

[170] S.F.Fan et.al, "MOTIS-C, A New Circuit Simulator for MOS LSI Circuits", IEEE ISCAS, Apr. 1977, pp. 700-703。

[171] R.B.Hitchcock et.al, "Timing Analysis of Computer Hardware"IBM J. of Res. and Develp·January 1982, Vol.26, No.1, pp.100-116。

[172] Toshiro Akino et.al, "Circuit Simulation and Timing Verification Based on MOS/LSI Mask Information", Proc.16th D.A.Conf.,1979, pp.88-94。

[173] B.T. Preas et. al, "Automtic Circuit Analysis Based on Mask Information". Proc.13th D. A. conf., 1976, pp.309-317。

[174] T. Mitsuhashi et. al, "An Integrated Mask Artwork Ana LSI System", Proc.17th D. A. conf., 1980, pp. 277-284。

[175] "AGS/862 Design Rule Check Package(DRC)" User's Manual No.A-21099 Applicon, July 1981。

[176] "GDSII Product Specification",CALMA,March 1981。

[177] U. Lauther "An O (N*LogN) Algorithm for Boolean

Hask Operations"Proc.18th D.A.Conf.,1981, pp.555-562。

[178] D.W. Hightower, "SLEUTH-A Metal-to Metal Autil Program in An Interactive Environment"Proc. 13th D.A.Conf.,1976, pp.318-326。

[179] J. A. Wilmore, "Efficient Operations on IC Hasks", Proc.18th D.A.Conf.,1981, pp.571-579。

[180] J.L.Bentley and T.A. Offmann, "Algorithms for Reporting and Counting Geometric Intersections",IEEE Trans. Comp., Vol. 6-28, No. 9, Sept. 1979, pp. 643-647。

[181] M. Yamin, "XYTOLR-A Computer Program for Integrated Ciruit Mask Dosign Checkout", Bell System Technical Journal 51-7, Sept.1972, pp.1581-1593。

[182] H.S. Baird et. al, "An Artwork Design Verification System"Proc.12th D.A.Conf.,1975, pp.414-420。

[183] L. M. Rosenbery et. al, "CRITIC: An Integrated Circuit Design Rule Checking." Proc. 11th D. A. Conf., 1974。

[184] M. H. Arnold et.al, "LYRA A New Approach to Geometric Layout Rule Checking",Proc.19th D.A.Conf., 1982, pp. 530-536。

[185] M.Takashima et.al, "Programs for Verifying Circuit Connec ting of MOS/LSI Mask Artwork" Proc. 19th D.A.Conf.,1982, pp.544-550。

[186] W.R.Heller et.al, "Predilection of Wering Space Requirements tor LSI", Proc. 14th D. A. Conf.,1977, pp. 32-42。

[187] W.M.Newman and R.F.Sproull, "Principles of Inte-

ractive Computer Graphics",MCGRAW-HILL, 1973。

[188] A.Barone et.al;"An Interactive Graphics System for LSI Design"Proc.Int.Conf.on Interactive Techniques Aided Design,Bologha Italy,September 1978。

[189] 胡大统:"布局、布线和制版中的交互型显示" 全国第二届数字系统 D.A. 会议论文,1982。

[190] 赵致格等:"用于 LSI CAD 的交互式图形编辑软件—IGES" 清华大学科学报告论文, TH 82002 (no. 109), 1982。

[191] L.Szanto, "Network Recognition of An MOS Integrated Circuit from the Topography of Its Mask",Computer Aided Design, Vol.10, No.3, 1978, pp.135。

[192] 中国科学院半导体所 209 组:"版图编辑软件—ZBCX 743"《半导体通讯》,1976, No.2, pp.1-17。

[193] 洪先龙、钟龙保、徐庆林、薛舒:"一个多功能的 LSI版图校验及处理软件系统—JC81"1982,《清华大学鉴定报告》。

[194] 洪先龙等:"LSI 版图图形编辑软件系统 ZB792"《计算机研究和发展》, Vol.21, No.4, 1982, pp.45-51。

[195] 中国科学院半导体所计算机辅助制版组:"LSI 版图编辑软件系统 JBY-2"《中国科学院、四机部联合鉴定报告》, 1981。

[196] 甘骏人:多边形覆盖的一种优化算法",全国第二届数字系统 D. A. 会议论文, 1982。

[197] 庄文君等:"LSI CAD 制版中的图形换转"《半导体学报》, Vol.1, No.4, 1980, pp.304-310。

[198] 庄文君:"LSI CAD 制版中的基本单元自动选择"《半导体学报》, Vol.3, No.2, 1982, pp.127-135。

[199] 小山田哲治, "LSIS-メクークパタニの最小分割"《情报处理》, Vol.16, No. 7, 1975.7。

[200] E.B.Eichelberger et.al, "A Logic Design for LSI Testability"Proc.14th Annual Design Automation Conf., New Orlens,LA.June 1977, pp.462-468。

[201] B.W.Kernighan and S. Lin: "An Efficient Heuristic Procedure for Partitioning Graph"Bell Syst.Tech.J., 1970, Vol. 49, No. 2, pp. 291-307。

[202] D.G.Schweikert and B.W.Kernighan, "A Proper Model for the Partitioning of Electrical Circuits" Proc. 9th D.A.Workshop, pp.57-62, 1972。

[203] 中国科学院半导体所自动制版研究组:《物理学报》25,88, 1976。

[204] Gerald M.Henrikoen, "Reticles by Automatic Pattern Generation"Proc.of the Society of Photo-Opticat Instrumentation Engineerns Vol. 100, Developments in Semiconductor Microlithograph II.86.1977。

[205] T.Ohtsuki et.al, "One-dimensional Logic Gate Assignmennt and Interval Graphs", IEEE Trans. Circuits Syst.,Vol.CAS-26, pp.675-684, Sept.,1979。

[206] C.G.Leker and J.CH. Boland, "Representation of a Finite Graph by a Set of Interval on the Real Line" Fund.Math.,Vol.51, pp.45-64, 1962。

[207] 洪先龙:"一种快速的线段裁剪算法", 全国第二届数字系统 D. A. 会议论文, 1982。

[208] C. J. DATE, "An Introduction to Database System" Second Edition, 1977。

[209] J.D.Ullman: "Principles of Database Systems",1980。

[210] C.C.Gotlieb and L.R.Gotlieb, "Data Types and Structures",1978。

[211] Gio Wiederhold, "Database Design"1977。

[212] 王鸿武、郑若忠：《数据库》1983。

[213] 郭玉钗、林宗楷："数字系统 CAD 数据库概况"，《全国第二届数字系统 D.A 会议论文》。

[214] H. Kawanishi, et. al.: "A Routing Method of Building Block LSI", REC 7th Asilomer Conf., On Circuits, Systems and Computers, 1973。

[215] 陈立东，张良震，庄文君，"Building Block 布图设计中的通道区布线序的确定"，《全国电路与系统年会论文集》，1986。

附 录

英—中名词对照表

A

algorithm 算法

efficient ～ 有效算法。

polynomical complexity ～ 多项式计算;复杂性算法

exponential complexity ～ 指数计算;复杂性算法

heuristic ～ 启发式算法

intractable ～ 难解的算法

inefficient ～ 非有效算法

ALU Arithmetic-Logical Unit 算术及逻辑;运算部件

assignment 分配

AP ～ Problem 分配问题

linear ～ 线性分配

quadratic ～ 二次分配

arc 弧;边

B

backtracking 回溯法

BFS Breadth-First Search 广度优先搜索

bipartition 对分

bisection 对分

block 块;围住

subblock 子块

building ～ 积木块

bound 界;界限

lower ～ 下界

static ～ 定界

dynamic ～ 动界

branch 支路;分支

～ and bound method 分支限界法

bristle block structure 竖块结构

C

CAA Computer-Aided Analysis 计算机辅助分析

CAD Computer-Aided Design 计算机辅助设计

CAE Computer-Aided Engineering 计算机辅助工程

CAM Computer-Aided Making 计算机辅助制造

CAT Computer-Aided Test 计算机辅助测试

cell 单元

polycell 多元胞

macro ～ 宏单元

functional ～ 功能单元

channel 通道区

vertical ～ 垂直通道区

horizontal ～ 水平通道区

adjustable ～ 可调整的通道区

irregular ～ 不规则边界通道区

chip 芯片

chromatic number 着色数

clearance 间隙;间距

contact-to-contact ～ 接点间距

clique 团；团图

cluster 群；结群

cluster-development method 成群展开法

CODASYL Conference on Data Systems Languages 美国数据系统语言协议会

column 列

complexity 复杂性

order of ～ 复杂度

connection 连接

connectivity 连接度

interconnection 互连

contact 接点

～ window 接触孔；接点孔

converge 收敛

convergence 收敛性

cover 覆盖

efficient ～ 有效覆盖

CPU Central Processing Unit 中央处理机

crossunder 渡桥

custom cell 任意元胞

semicustom cell 准任意元胞

custom design 用户设计

semi-custom design 半用户设计

cut 切割

～ line 割线

～ vertex (node) 割点

cutset 割集

min-cut algorithm 最小割算法

cycle 圈；回路

D

D. A. Design Automation 设计自动化

data 数据

～ base 数据库

～ base administrator 数据库管理员

～ base control system 数据库控制系统

～ base management system 数据库管理系统

～ base task group 数据库任务组

～ description language 数据描述语言

～ manipulation language 数据操作语言

degree 度,度数

out going ～ 出度

in goming ～ 入度

delay 延时

density 密度

channel ～ 通道区布线密度

design 设计

physical ～ 实体设计

logical ～ 逻辑设计

hierarchical ～ 分级设计

top-down ～ 自顶向下设计

bottom-up ～ 自底向上设计

DFS Depth-First Search 深度优先搜索

dimension 维;维数

one-dimensional 一维的

two-dimensional 二维的

distance 距离

manhattan ～ 曼哈坦距离
euclidean ～ 欧几里得距离

dogleg 曲干

domain 域

E

edge 边

escape 逸出
～ point 逸出点
～ line 逸出线

exchange 交换

F

feasible 可行的
～ solution 可行解

feed-through 过渡通道；冗余通道
(jumper)
～ Cell 过渡通道单元；冗余通道单元

flop-flip 触发器

foreign key 外来关键字

G

gate 门；逻辑门；栅极

gate array 门阵列
master slice 母片式
prediffused array 预布阵列

gate matrix 栅阵列

graph 图

linear ～ 线图

bipartite ～ 二分图；偶图

oriented ～ 有向图

(directed ～) (有向图)

vertical constraint ～ 垂直限制图

interval ～ 间隔图；区间图

subgraph 子图

weighted ～ 加权图

position ～ 位置图

color ～ 色图

clique ～ 团图

plane ～ 平面图

complete ～ 完全图；完备图

null ～ 空图

computer graphics 计算机图形学

grid 网格

H

heuristic method 启发式方法

hierarchical model 层次模式

I

IC Integrated Circuit 集成电路

metal-oxide-semiconductor ～ MOS 集成电路

thick film ～ 厚膜集成电路

thin film ～ 薄膜集成电路

hybrid ～ 混合集成电路

bipolar ～ 双极集成电路

independent 独立的;无关的

IEEE Institute for Electrical and Electronic Engineers 电气和电子工程师学会(美国)

integrated data store 集中数据存贮

interactive 交互

～ system 交互式系统

～ terminal 交互式终端

interchange 交换;互换

pairwise ～ 成对交换

neighborhood ～ 接交换

force-directed ～ 力矢量交换

interdependent 互依

intersection(∩) 交

I/O Input/Output 输入/输出

ISCAS International Symposium of Circuits And Systems 国际电路及系统会议

isolation 隔离

iterative 迭代

L

layer 层

multilayer 多层

buried ～ 埋层

epitaxial ～ 外延层

layout 布图设计

library 库

light pen 光笔

link 链;链接

pair-linking method 对联结法

list 表;目录;列表

adjacency vertex listing 邻接编目法

successor listing 邻接顺序法

edge listing 边目录法

local 局部的

～ optimal 局部优化的

logic 逻辑

ECL Emitter-coupled ～ 发射极耦合逻辑电路

combinational ～ 组合逻辑

sequential ～ 时序逻辑

TTL Transistor-Transistor ～ 晶体管-晶体管逻辑电路

loop 回路

LSI Large Scale Integration Circuit 大规模集成电路

VLSI Very Large Scale Integration Circuit 超大规模集成电路

SLSI Super Large Scale Integration Circuit 超大规模集成电路

M

mask 版;掩膜版

～ making 制版

mask-alignment tolerance 掩膜对准误差容限

match 匹配

maximun ～ 最大匹配

minimum ～ 最小匹配

matrix 矩阵

submatrix 子矩阵

incidence ～ 关联矩阵

circuit ～ 回路矩阵

adjacency ～ 邻接矩阵

cutset ～ 割集矩阵

path ～ 路径矩阵

connected ～ 连接矩阵

sparse ～ 稀疏矩阵

weight ～ 权矩阵

cost ～ 价格矩阵

maze 迷宫

merge 合并；归并

minicomputer 微机

MIPS Million of Instructions Per Second 百万条指令/秒

modul 模块

MOS Metal-Oxide-Semiconductor 金属氧化物半导体

CMOS Complemetary MOS 互补 MOS

NMOS N-Channel MOS N 沟道 MOS

PMOS P-Channel MOS P 沟道 MOS

N

necessary and sufficient conditions 充分必要条件

net 线网

network model 网络模式

subnet 子线网

node 顶点

O

objective 目标
~ function 目标函数
order 次序；顺序；阶
overflow 溢出
overlap 重迭
overlay 覆盖

P

pack 压缩
package 组件；插件；包
parasitic(al) 寄生的
~ transistor 寄生晶体管
~ capacity 寄生电容
partition 划分
path 路径
shortest ~ 最短路径；路径
manhattan ~ 曼哈坦路径；曼哈顿路径
critical ~ 关键路径
P.C.B.Printed-Circuit Board 印刷电路板
permute 排列
p.G.Pattern Generator 图形发生器
pin 针孔
pLA Programmable Logic Array 可编程序逻辑阵列
placement 布局

linear ～ 行式布局

constructive ～ 构造布局

iterative improvement of ～ 布局的迭代改善

polycell 多元胞

(standard cell) (标准单元)

Polygon 多边形

primary key 主关键字

priority 优先权;优先度

program 程序;规划

integer programming 整数规划

linear programming 线性规划

dynamic programming 动态规划

problem 问题

traveling salesman ～ 巡回售货员问题;货郎担问题

P-problem 多项式计算复杂性问题

NP-problem 非多项式计算复杂性问题

NP-complete ～ 非多项式计算复杂性完备问题

matching ～ 匹配问题

Q

quadrature 正交

R

RAM Random Access Memory 随机存取存贮器

range 区域;子线网形式长度参量

rate 比率;速率

completion ～ of routing 布线完成率;布通率

convergence ～ 收敛速率

rectangle 矩形

recurrence 递归

region 区域

diffused ～ 扩散区

register 寄存器

RALU Register, Arithmetic and Logic Unit 寄存器；运算和逻辑部件

relational model 关系模式

relaxation 松弛

force-directed pairwise ～ 力矢量成对松弛法

force-directed ～ (FDR) 力矢量松弛法

generalized-force-directed ～ (GFDR) 广义力矢量松弛法

ROM Read-Only Memory 只读存贮器

ROS Read-Only Storage 只读存贮器

routing 布线

wiring 布线

global(Loose) ～ 总体布线

final(detail) ～ 最终布线；细布

channel ～ 通道区布线

rerouting 重布

linear ～ 线探法布线

row 行

S

scan 扫描

linear ～ 行扫描

schema 模式

segment 段

track ～ 通道段

set 集合；系

subset 子集

empty ～ 空集

SIAM (J.SIAM) Society for Industrial and Applied Mathematics 工业及应用数学学会

side 边

signal 信号；信号网

silicon 硅

monocrystalline 单晶硅

polycrystalline 多晶硅

simulation 模拟

logic ～ 逻辑模拟

fault ～ 故障模拟

circuit ～ 电路模拟

timing ～ 延时模拟

slack 余量

slice 条；片；切条；切片

slot 空位

span 广度

channel ～ 通道区内具有最大布线密度的列的总数

structure 结构

data ～ 数据结构

substrate 衬底

switching box 四边通道

symbolic layout 符号法布图

system 系统

digital ～ 数据系统
CAS Circuit and ～ 电路和系统
DMS Data Management ～ 数据管理系统

T

terminal 端点；接点；计算机终端
electrically equivalent ～ 电学等价接点
logically equivalent ～ 逻辑等价接点
topology 拓扑学
～ synthesis 拓扑综合
track 通道
transistor 晶体管
bipolar ～ 双极晶体管
field-effict ～ 效应晶体管
tree 树
subtree 子树
～ branch 树枝
minimal ～ 最小树
(MST Minimum Spanning ～)
DFS ～ 深度优先搜索树
linear ～ (Chain ～) 线形树；链接树
steiner ～ 斯坦尼树；最小连接树
directed ～ 有向树
rooted ～ 有根树；有源树
clustering ～ 结群树
2-tree 2-树；二叉树
tuple 元组

U

union(∪) 并

V

vertex 顶点

ascendant ～ 上辈顶点

descendant ～ 下辈顶点

via 通孔

～ hole 通孔

W

weight 权重

window 孔；窗口

emitter contact ～ 发射极接点孔

collector contact ～ 集电极接点孔

base contact ～ 基极接点孔

gate contact ～ 栅极接点孔

source contact ～ 源极接点孔

drain contact ～ 漏极接点孔

wirability 可布性

workstation 工作台；工作站

Z

zone 带区

中—英名词对照表

三　画

大规模集成电路　LSI Large Scale Integration Circuit

超大规模集成电路　VLSI Very Large Scale Integration Circuit(SLSI Super Large Scale Integration Circuit)

门；逻辑门　gate

门阵列　gate array

母片式　master slice

预布阵列　prediffused array

广度　span

通道区内具有最大布线密度的列的总数　channel span

广度优先搜索　BFS Breadth-First Search

工作台；工作站　workstation

孔；窗口　window

发射极接点孔　emitter contact window

集电极接点孔　collector contact window

基极接点孔　base contact window

栅极接点孔　gate contact window

源极接点孔　source contact window

漏极接点孔　drain contact window

四　画

中央处理机　CPU Central Processing Unit

计算机辅助设计 CAD Computer-Aided Design
计算机辅助工程系统 CAE Computer Aided Engineering (system)
计算机辅助制造 CAM Computer Aided Making
计算机辅助分析 CAA Computer-Aided Analysis
计算机辅助测试 CAT Computer Aided Test
支路,分支 branch
　　分支限界法 branch and bound method
区域 region
　　扩散区 diffused region
区域;子线网形式长度参量 range
分配 assignment
　　分配问题 AP Assignment Problem
　　线性分配 linear assignment
　　二次分配 quadratic assignment
比率;速率 rate
　　布线完成率;布通率 completion rate of routing
　　收敛速率 convergence rate
互依 interdependent
匹配 match
　　最大匹配 maximum match
　　最小匹配 minimum match
元组 tuple

五　画

主关键字 primary key
只读存贮器 ROM Read-Only Memory (ROS Read-Only Storage)

可编程序逻辑阵列 PLA Programmable Logic Array

布线 routing

总体布线;概略布线 global(loose) routing

最终布线;细布 final(detail) routing

通道区布线 channel routing

重布 rerouting

线探法布线 linear routing

布图设计 layout

布局 placement

行式布局 linear placement

构造布局 constructive placement

布局的迭代改善 iterative inprovement of placement

可布性 wirability

电气和电子工程师学会（美国） IEEE Institute for Electrical and Electronic Engineers

目标 ojective

目标函数 ojective function

正交 quadrature

边 edge

边 side

四边通道 switching box

对分 bisection(bipartition)

切割 cut

割线 cut line

割点 cut vertex (node)

割集 cutset

最小割算法 min-cut algorithm

可行的 feasible

可行解 feasible solution

外来关键字 foreign key

六 画

关系模式 relational model

设计 design

实体设计 physical design

逻辑设计 logical design

分级设计 hierarchical design

自顶设计 top-down design

自底设计 bottom-up design

用户设计 custom design

半用户设计 semi-custom design

设计自动化 D. A. Design Automation

印刷电路板 P. C. B. Printed-Circuit Board

问题 problem

巡回售货员问题；货郎担问题 traveling salesman problem

多项式计算复杂性问题 P-problem

非多项式计算复杂性完备问题 NP-complete problem

匹配问题 matching problem

团图 clique

回溯法 backtracking

任意元胞 custom cell

准任意元胞 semicustom cell

多元胞 polycell

标准单元 standard cell

多边形 polygon

光笔 light pen

百万条指令/秒 MIPS Million of Instruction Per Second
权重 weight
扫描 scan
　行扫描 linear scan
过渡通道;冗余通道 feed-through (jumper)
　过渡通道单元 feed-through cell
曲干 dogleg
交 intersection(n)
交换 exchange
交换;互换 interchange
　成对交换 pairwise interchange
　邻接交换 neighborhood interchange
　力矢量交换 force-direeted interchange
交互 interactive
　交互式系统 interactive system
　交互式终端 interactive terminal
网格 grid
网络模式 network model
收敛 converge
　收敛性 convergence
压缩 pack
合并;归并 merge
并 union(∪)
划分 partition
充分必要条件 necessary and sufficient conditions
优先权;优先度 priority
次序;顺序;阶 order
列 column
行 row

回路 loop

七 画

系 set

系统 system

数字系统 digital system

电路和系统 CAS Circuit and System

数据管理系统 DMS Data Management System

库 library (base)

启发式方法 heuristic method

局部的 local

局部优化的 local optimal

余量 slack

连接 connection

连接度 connectivity

互连 interconnection

层 layer

多层 multilayer

埋层 epitaxial layer

外延层 epitaxial layer

层次模式 hierarchical model

针孔 pin

间隙 clearance

接点间距 contact-to-contact clearance

块;围住 block

子块 subblock

积木块 building block

芯片 chip

条片；切条；切片 slice
延时 delay

八 画

组件；插件；包 package
拓扑学 topology
　拓扑综合 topology synthesis
金属氧化物半导体 MOS Metal-Oxide-Semiconductor
　互补 MOS CMOS Complementary MOS
　N 沟 MOS NMOS N-Channel MOS
　P 沟 MOS PMOS P-Channel MOS
单元 cell
　多元胞 polycell
　宏单元 macro cell
　功能单元 functional cell
线网 net
　子线网 subnet
迷宫 maze
迭代的 iterative
表；目录；列表 list
　邻接编目法 adjacency vertex listing
　邻接顺序法 successor listing
　边目录法 edge listing
图 graph
　线图 linear graph
　二分图；偶图 bipartite graph
　有向图 oriented graph(directed graph)
　垂直限制图 vertical constraint graph

间隔图;区间图 interval graph
子图 subgraph
加权图 weighted graph
位置图 position graph
色图 color graph
团图 clique graph
平面图 plane graph
完全图 complete graph
空图 null graph
计算机图形学 computer graphics
图形发生器 P. G.Pattern Generator

空位 slot

弧;边 arc

顶点 vertex(node)
上辈顶点 ascendant vertex
下辈顶点 descendant vertex

衬底 substrate

松弛 relaxation
成对力矢量松弛法 force-directed pairwise relaxation
力矢量松弛法 force-directed relaxation(FDR)
广义力矢量松弛法 generalized-force-directed relaxation(GFDR)

九 画

信号;信号网 signal

结构 structure
数据结构 data structure
竖块结构 bristle block structure

度 degree

出度 out going degree

入度 incoming degree

界限 bound

下界 lower bound

定界 static bound

动界 dynamic bound

独立的；无关的 independent

栅阵列 gate matrix

版；掩膜版 mask

制版 mask making

膜对准误差容限 mask-alignment tolerance

树 tree

子树 subtree

树枝 tree branch

最小树 minimal tree(minimum spanning tree)

深度优先搜索树 DFS tree

线形树；链接树 linear tree

斯坦尼树最小连接树 steiner tree

有向树 directed tree

有根树；有源树 rooted tree

结群树 clustering tree

2-树；二叉树 2-tree

带区 zone

段 segment

通道段 track segment

重迭 overlap

覆盖 cover

有效覆盖 efficient cover

复杂性 complexity

复杂度 order of complexity

美国数据系统语言协议会 CODASYL Conference on Data Systems Languages

十 画

递归 recurrence

通道区 channel

垂直通道区 vertical channel

水平通道区 horizontal channel

可调通道区 adjustable channel

不规则边界通道区 irregular channel

通孔 via (via hole)

通道 track

矩形 rectangle

矩阵 matrix

并联矩阵 incidence matrix

回路矩阵 circuit matrix

邻接矩阵 adjacency matrix

割集矩阵 cutset matrix

路径矩阵 path matrix

连接矩阵 connected matrix

稀疏矩阵 sparse matrix

权矩阵 weight matrix

子矩阵 submatrix

价格矩阵 cost matrix

十 一 画

逸出 escape

逸出点 escape point

逸出线 escape line

逻辑 logic

发射极耦合逻辑电路 ECL Emitter-Coupled Logic

晶体管-晶体管逻辑电路 TTL Trainsistor-Trainsistor Logic

组合逻辑 combinational logic

时序逻辑 sequential logic

深度优先搜索 DFS Depth-First Search

维(数) dimension

一维的 one-dimensional

二维的 two-dimensional

随机存取存贮器 RAM Random Access Memory

密度 density

通道区布线密度 channel density

圈,回路 cycle

硅 silicon

单晶硅 monocrystalline

多晶硅 polycrystalline

接点 contact

接点孔 contact window

着色数 chromatic number

寄生的 parasitic (al)

寄生晶体管 parasitic transistor

寄生电容 parasitic capacity

寄存器 register

寄存器；运算和逻辑部件 RALU Register Arithmetic and Logic Unit

域 domain

十 二 画

链；链接 link

对链接法 pair-linking method

程序；规划 program

整数规划 integer programming

线性规划 linear programming

动态规划 dynamic programming

距离 distance

曼哈坦距离 Manhattan distance

欧几里得距离 Euclidean distance

集成电路 IC Integrated Circuit

MOS集成电路 metal-oxide semiconductor IC

厚膜集成电路 thick film IC

薄膜集成电路 thin film IC

混合集成电路 hybrid IC

双极集成电路 bipolar IC

集合 set

子集 subset

空集 empty set

集中数据存贮 integreted data store

晶体管 transistor

双极晶体管 bipolar transistor

场效应晶体管 field-effect transistor

渡桥 crossunder

隔离 isolation

十 三 画

微机 minicomputer

路径 path

最短路径 shortest path

曼哈坦路径; manhattan path

关键路径 critical path

触发器 flop-flip

输入/输出 I/O Input/Output

溢出 overflow

群;结群 cluster

成群展开法 cluster-development method

数据 data

数据库 data base

数据库管理系统 DBMS Data Base Management System

数据库管理员 DBA Data Base Administrator

数据描述语言 DDL Data Description Language

数据操作语言 DML Data Manipulation Language

数据库控制系统 DBCS Data Base Control System

数据库任务组 DBTG Data Base Task Group

十 四 画

端点;计算机终端 terminal

电学等价接点 electrically equivalent terminal

逻辑等价接点 logically equivalent terminal

算法 algorithm

有效算法 efficient algorithm

多项式计算复杂性算法 polynomical complexity algorithm

指数计算复杂性算法 exponential complexity algorithm

启发式算法 heuristic algorithm

难解的算法 intractable algorithm

非有效算法 inefficient algorithm

算术及逻辑运算部件 ALU Arithmetic-Logical Unit

十 五 画

模拟 simulation

逻辑模拟 logic simulation

故障模拟 fault simulation

电路模拟 circuit simulation

延时模拟 timing simulation

模块 modul

模式 schema