申请上海交通大学工程硕士专业学位论文

考虑串扰效应与时延的多级布线器研究

学　　校： 上海交通大学

院　　系： 微电子学院

班　　级： Z0621091

学　　号： 1062109064

工程硕士生： 黄世杰

工程领域： 软件工程

导　师： 施国勇（教授）

上海交通大学微电子学院

2008 年 11 月

此研究由上海市浦江人才基金资助(No. 07pj14053)

A Dissertation Submitted to Shanghai Jiao Tong University for the

Maser Degree of Engineering

A SURVEY ON MULTILEVEL ROUTER CONSIDERING CROSSTALK AND TIMING OPTIMIZATION

Author：Shijie Huang

Specialty: Software Engineering

Advisor : Prof. Guoyong Shi

School of Microelectronics

Shanghai Jiao Tong University

Shanghai, P.R.China

November 18, 2008

[2] **This research is supported by 2007 Shanghai Pu Jiang Scholar Funding Project(No. 07pj14053)**

考虑串扰效应与时延优化的多级布线器研究

摘 要

随着 VLSI 深亚微米工艺的不断进步，集成电路中的芯片的互连延时，以及互连线之间的信号完整性问题已经成为决定电路性能的关键因素。相比于优化传统的自动化布线指标（比如布通率，拥挤）所采用的策略，考虑更精确时延估计，串扰优化等信号完整性指标的布线策略往往要用到电路的高阶矩。而传统布线框架所采用的 Elmore 时延模型其本质是电路的一阶矩，实验已证明它对快速变化的信号具有较低的精确度，且从电路的一阶矩也无法反映出电路上传输信号的波形质量。所以最近基于电路高阶矩的性能驱动布线得到广泛关注。

本文在目前流行的多级布线框架基础上，结合最近提出的高效的符号化矩计算方法（SMART 算法），设计了基于 SMART 算法的多级布线器。我们通过 SMART 算法，快速地得到线网结点的高阶矩信息，从而能采用更精确的基于高阶矩的时延模型进行时延驱动布线。同时我们也利用高阶矩信息进行全局布线阶段的串扰评估与优化。实验结果表明，我们的方法有效地改善了关键线网的时延与串扰。

关键词：多级布线，高阶矩，中心矩，性能驱动布线，串扰优化

A SURVEY ON MULTILEVEL ROUTER CONSIDERING CROSSTALK AND TIMING OPTIMIZATION

ABSTRACT

With the continuously development of VLSI sub-micron technology, the delay of interconnect and signal integrity have become the critical problem that affect the performance of chips.Compared with traditional routing strategies, high-order moments have to be considered when new routing strategies are designed to deal with more precise timing estimation and crosstalk optimization. The Elmore delay timing estimation model which was widely applied to traditional routing framework is actually the first order of circuit that has low precision for fast-transformed signal which has been proved. And the quality of waveform on nets can not reflected well only by first order moment of circuit.So recently moment-based global routing techniques have been concerned widely for performance driven routing.

This thesis is based on a popular multilevel routing framework,and we design a multilevel router using the highly efficient symbolic moment calculation method (SMART algorithm) proposed recently.Using SMART,we could quickly get the high order moments of circuits based on which high order timing estimation model is applied to timing driven routing. At the same time, we use high order moments of circuits to do crosstalk optimization in the stage of global routing. The experimental result showed that our method get a good improvement on critical net timing budget and crosstalk optimization.

Keywords: multilevel routing, high order moment, central moment, performance driven routing, crosstalk optimization

目 录

图片目录

表格目录

第一章　前言

1．1　布线研究前景

随着集成电路进入深亚微米时代，电路的设计规模越来越大。越来越多的功能，甚至是整个系统都被集成到单个芯片之中，出现了系统级芯片（SOC）的设计概念。而作为物理设计（Physical Design）中的重要阶段布线（Routing），其算法研究面临巨大挑战。其中一个挑战是：随着集成度的提高，芯片上模块间排列更加紧密，互连线的间距进一步减小，工作频率进一步提高。这都使得集成电路中互连线的时延和耦合效应越来越明显。因此，在布线阶段能恰当有效地估计线网长度，拥挤和串扰情况，并使线网均匀分布，使芯片的时延优化，串扰尽可能小，这是目前的研究热点。

我们知道，随着布线规模与复杂性的增加，使得面向线网的布线方法在解决所有线网在布线时相互影响上遇到了很大的问题。1971 年后，在布线问题上引入了“分级布线”（Heirarchical Routing）的概念。2001 年由 J.Cong**[1]**提出了“多级布线”（Multilevel Routing）的概念。这些布线方法都是为了解决大规模布线问题所提出的方法。其中，多级布线将布线区域一级级划分，并将总体布线，详细布线，延时和拥挤估计集成在了每一级中，使得每一级的时延和拥挤估计都有较高的精确度。每级的布线结果又为下一级布线提供良好的指导作用。因此它逐渐成为目前流行的布线框架。

1．2　性能驱动布线算法思想介绍

在大规模布线系统中，有许多需要优化的目标：芯片面积，布通率，布线分布的均匀性等等，性能优化是随着工艺的发展才被列入优化目标的。我们知道，随着 VLSI 深亚微米工艺的不断发展，集成电路中的互连延迟逐渐成为决定电路速度的重要因素。据统计，在高密度，高速度的超大规模集成电路中，内部连线延迟的平均值已经达到时钟周期的 50％到 70％以上。因此，芯片内部连线延迟已经成为决定芯片性能的一个主要因素。因此，近年来，对时延驱动布局布线算法进行了广泛的研究。

以往一些以连线延迟最小化为目标的总体布线算法，只是简单地将最小化线网连线延迟的目标转化为最短线网长度布线问题，以为求得了线网最短的连接，就保证了连线的最小延迟。实际上，只有在二端线网的情况下，或者连线电阻与输出驱动电阻相比可以忽略不计的情况下，最短线网连接才能保证最短的连线延

迟。对于多端线网，在深亚微米工艺下，连线电阻与输出驱动电阻相比不能忽略不计时，线网的漏端延迟不仅与线网总连线长度有关，而且与线网布线树的结构有关。

时延驱动总体布线算法的基本思想是在优化布线质量，减小布线区域面积，均匀线网分配的同时，最小化芯片的内部连线延迟，使之限制在用户要求的范围以内，时延驱动总体布线算法主要有两种基本策略：

（1） 基于线网的时延驱动总体布线策略

在这种策略中，将关键路径上的连线延迟约束分配到各相关线网中，作为对线网延迟的约束。在总体布线时，限制线网的布线延迟一定要满足各自的延迟约束。这样，在最后的布线结果中，就可以保证所有关键路径上的延迟都满足用户给出的延迟约束。这会做法的优点是控制简单，易于实现。但是由于对每个线网的连线长度和拓补结构都有严格的限制，而预先对关键路径的延迟的分配又有一定的盲目性，没有考虑到各布线区的拥挤情况，因此可能会影响布线质量的提高。

（2） 基于关键路径的时延驱动总体布线策略

这种算法在总体布线时，并不单一考虑每条线网的延迟约束，而是将线网的延迟约束检查放到与其相关的关键路径的延迟约束检查中，只要关键路径上总的延迟约束满足，线网的布线树就会被接受。这种方法虽然控制起来比较复杂，但是可以避免不合理的延迟分配，放宽线间的布线约束，得到更好的布线质量。

与时延相比，串扰是一个更新的优化目标。只是随着线网密度的增加，串扰优化才显得越来越重要。

串扰是由互连线间的耦合电容和耦合电感引起的耦合效应。它会带来信号噪声并导致信号延迟，严重的时候甚至会导致电路的功能错误。为了保证电路的正确性，提高电路性能，需要在物理设计阶段尽量避免过度串扰的产生。

导致串扰的因素有耦合电容和耦合电感两种。两相邻导体间，因电容的积蓄而引发彼此额外的电性作用，称为“电容耦合”。这种作用可以在物理上等效为两个导体间存在一个电容，也就是耦合电容。根据电容的电特性，电容一端电位的变化会导致电容另一端产生相应的电位变化。那么如果耦合电容两端分别位于两个金属信号线上，则一个信号线上的电位变化就会引起另一个信号线上的电位相应地变化。这就会在后者的信号中带来噪声，也会给前者的信号带来延迟。另外，相邻或不相邻导体间，因磁场中磁通量变化导致感应电流和感应电动势的现

象，称为“电感耦合”。这种作用可以用两个导体间存在电感来等效，也就是耦合电感。根据法拉第电磁感应定律，一个金属信号线上电流的变化会引起另一个信号线上产生相反方向的感应电流，从而会在后者的信号中引入噪声和延迟。

目前发表的研究成果主要是针对互连线间的耦合电容。在**[2][3]**中，作者通过求解偏微分方程来推导串扰的表达式，这种方法虽然比较精确，但是时间耗费大并且适用范围小。其实对于大规模布线，尤其是总体布线过程，很精确的串扰分析是没有必要的。只要能有效地估计串扰的影响，并且指导布线的走向以优化串扰就可以了。J.Cong 在**[4]**提出了 RATS 树算法来做串扰优化。RATS 算法利用了电路的高阶矩分量。电路的高阶矩可以刻画响应波形的质量，也可以反映时延信息等指标。传统的 Elmore 延迟其实就是电路的一阶矩分量。利用电路的高阶矩做时延与串扰优化已经成为研究的热点，而这方面依赖于快速有效的矩计算方法。Shi**[5]**提出的符号化矩计算算法（SMART 算法），算法利用二叉决策图共享子图的特点，高效地将电路结构存储为图的数据结构。图的结构与实际布线结构相对应，不但可以快速有效计算电路的各阶矩，计算结果也可以直接用来指导布线。

1．3　论文完成的工作和内容安排

在论文中，主要研究物理设计中的布线问题以及如何提高布线的时延性能与改善串扰影响。本文的研究基于 Shi**[5]**的符号化矩计算算法（SMART 算法），在目前流行的多级布线框架上，设计时延优化与串扰优化的性能驱动布线方法。

全文的组织如下：第二章介绍了布线问题的描述方法，说明了研究目的并概要介绍了基于 SMART 算法的性能驱动布线方法。第三章我们详细介绍了多级布线框架的概念，以及我们用到的基本数据结构，并对程序的输入文件格式进行了说明。第四章详细介绍了我们基于 SMART 算法的考虑串扰效应与时延优化的多级布线算法。第五章是程序的图形化显示与测试。第六章对全文进行总结。

第二章　问题描述方法

2．1　研究目的

我们知道，在大规模布线器的设计中，全局布线是第一个步骤，也是相当关键的一个步骤，它为后期的详细布线提供了初始的布线走向。这一阶段布线结构的好坏决定了芯片整体的性能。传统的性能优化目标多是考虑拥挤，总线长等**[6]**。但随着 VLSI 深亚微米工艺的不断进步，集成电路中的芯片的互连延时，以及互连线之间的信号完整性问题已经成为决定电路性能的关键因素。互连线对电路性能的影响主要体现在增大信号的时延以及带来了信号之间的串扰等问题。

因此，当时延和信号完整性问题变得越来越重要，在布线每一阶段运用的策略上就要有所改变以适应新问题的需要。我们这里所说的改变是兼顾原有布线指标的改变，即我们设计的布线方案不但要优化时延和串扰，而且要满足传统布线方案的优化指标，比如拥挤，总线长等。在拥挤优化的指标上，我们采用目前流行的布线框架－－多级布线框架。这种方法首先处理局部小区域的线网，再一次一级一级地处理大区域的线网，并且每一级我们都可以做详细的拥挤估计，这使得布通率得到很好的保证。

我们的研究就是在多级布线框架基础上，结合电路高阶矩高速求解，高阶时延模型等最新研究成果，提出有效的串扰评估模型。从而在全局布线阶段考虑时序优化和串扰优化指标。目前该方面的研究中，有效结合电路高阶矩同时考虑时序优化和串扰优化的研究并不多，本文也希望能在这方面的研究中贡献一个思路。

2．2　问题描述

传统的布线方法将布线流程划分成全局布线和详细布线两个阶段。全局布线为详细布线提供初始布线走向，这一阶段布线结构的好坏决定了芯片整体的性能，其布线结果直接影响到电路的内部连线延迟。传统全局布线的目标主要考虑线长和拥挤指标。因为线长直接影响芯片面积，时序特性等方面性能。而考虑拥挤指标可以均匀线网分配，提高布通率。

随着工艺的发展，时延优化，信号完整性问题已被列入优化的目标。而传统的时延模型往往采用基于 Elmore 延时的 RC 树，但 Elmore 模型对快速变化的信号具有较低的精确度，制约了它在深亚微米工艺上的应用**[7]**。为了得到更精确

的时序估计，就需要更精确的时延模型。而这需要借助于电路的高阶矩分量**[8]**。如何在布线调整过程中快速地得到电路的高阶矩分量，并借助这些信息指导布线走向以优化时延与信号线之间的串扰等性能指标，是一个重要的研究课题。

2．3 基于 SMART 算法的性能驱动布线方法介绍

基于 Shi 在**[5]**中提出的 RLC 树状电路各阶矩的符号化计算方法，我们可以得到基于高阶矩的时延模型和串扰评估模型。所以，本文的主要目的是在已有的多级布线框架基础上**[9]**，对时序延时和线上串扰做出更加精确的分析和优化。为了优化时延，我们从构建线长最优矩形斯坦纳树的全局布线阶段开始，在多级布线的每个阶段，我们都采用精确的时延模型进行关键路径上的时序估计，从而进一步调整最优矩形斯坦纳树，使关键路径上的时延满足时序约束，且总线长达到最小。为了优化串扰影响，我们在不改变全局布线树拓补结构（已满足时序约束）的基础上，结合串扰评估的模型，调整局部二端线网的布线走向以降低相邻线网的影响。

实现是基于部分开放的源代码，它们是台湾大学 CAD 实验室提供的基于多级布线框架的部分源程序**[9]**，UCLA CAD 实验室提供的 Layout Database library 与测试实例以及 LEDA 算法软件包**[10]**和 Qt4 图形开发软件包**[11]**。

算法输入：

UCLA CAD 实验室提供的测试例子，以及 ISPD04 IBM Benchmarks**[12]**测试例子。这些测试实例（GDF 文件）包含有标准单元信息，电路布局后的网表信息，这主要包括各个标准单元的端点位置信息和各个端点的互连信息。

算法输出：

输出布线后的电路信息，并报告所有线网的时延及运行时间情况。

2．4 本章小结

本章介绍了该论文的研究目的，即我们将结合电路高阶矩同时考虑布线的时序优化和串扰优化问题。我们也对论文中要解决的布线问题进行了描述，以及解决该布线问题采用的方法。本章也对我们的多级布线平台进行了简单的介绍，同时说明了我们算法的输入和输出。

第三章　现有线长最优总体布线算法与多级布线框架介绍

3．1　多级布线算法介绍

3．1．1　　多级布线框架介绍

多级布线的概念最早由UCLA 的CAD Lab 提出的[1]，台湾大学的Lin和Chang[9]在此基础提出了一个同时考虑了可布通性和时延优化的多级自动布线框架。

该框架将布线过程分成粗化布线（coarsening）和细化布线（uncoarsening）两个阶段，而每个阶段都包含了多级的总体布线和详细布线。这是一个开放式的布线框架，我们可以在这个框架中设计自己的总体布线和详细布线算法，然后利用提供的标准测试文件进行性能测试。下面对粗化布线和细化布线做详细介绍。

1　粗化布线

图 1展示了多级布线的流程。图中的左半部分为粗化布线流程。如图所示，这个阶段首先将布线区域划分成一个N＊N（N为2的n次方）的小区域，为每个仅在这些小区域的局部线网分别做总体布线和详细布线，对一些无法布通的线网进行标记，之后分析每个区域内的拥挤情况，并对线网进行时延分析。这个过程叫第一级布线。当小区域布线完成后，将邻近的小区域合并成较大的区域，并重复之前在小区域的步骤。这些合并小区域成大区域的布线阶段是个反复迭代的过程，称为第二级，第三级等多级布线。最终合并成的区域就是整个布线区域了。此时粗化布线阶段结束，进入细化布线。

粗化布线阶段能完成大部分的布线任务，只是将局部暂时无法布通的线网留给细化布线阶段来处理。粗化布线的优点是它将总体布线和详细布线集成到了每一级，这样当每一级布线在为下一级布线做布线资源预估及时延预估的时候，预估结果会更加准确。另外，因为每一级都需要进行总体布线，总体布线的质量往往直接影响整体布线的性能。

2　细化布线

图 1 右边部分为细化布线的过程。细化布线阶段会对粗化布线中做了标记无

法布通的线网进行布线，主要采用迷宫算法，拆线重布算法进行布线。细化布线也是个多级布线的过程，只是这个阶段的区域划分和粗化阶段正好相反，是由整个布线区域逐级划分成小区域。当所有层级都处理完毕，则布线结束。

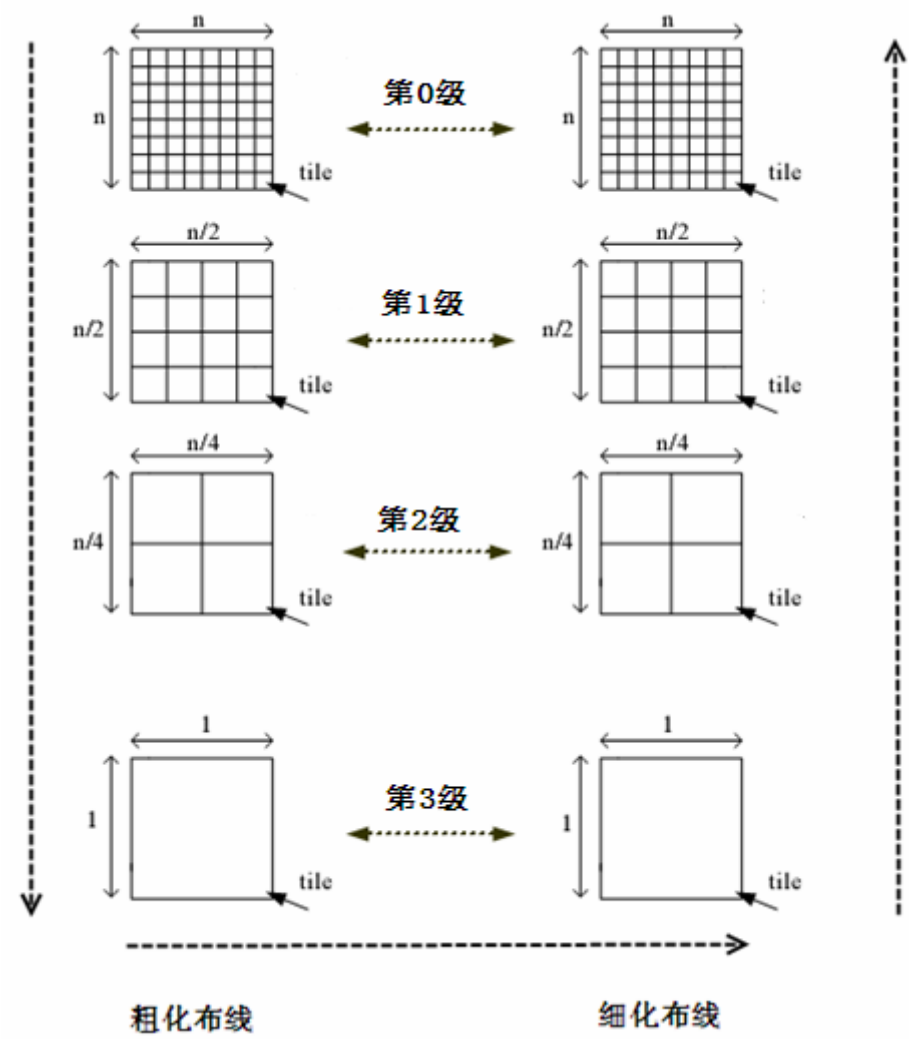


图 1　　多级布线流程

Fig.1　　multilevel routing flow

3．1．2　　输入文件格式

我们处理的输入文件格式为标准 gdf 文件格式，该文件如图 2 所示，标识了基本单元的几何信息。在 gdf 文件格式中，规定了一些关键字。我们对与本文设计有关的关键字用黑体字标识了出来，并加以说明。

```
( Cell: name
    ( Text:  number_of_layers   string)
    ( Text:  wire_widths        string )
    ( Text:  via_widths         string )
    ( Text:  wire_spacings      string )
    ( Text:  via_spacings       string )
    ( Text:  vertical_wire_costs      string )
    ( Text:  horizontal_wire_costs    string )
    ( Text:  via_costs          string )
    ( Path: name
        (new)(layer  type)(width  number)(pt  x1)(pt  y1)(pt  x2)(pt  y2)
        …
    )

    ( Port: name (pt  x  y) )
    ...
    ( net : name )
     (PortRef  name)
    ...
)
```

图 2　输入 gdf 文件格式

Fig.2　format of gdf file

关键字说明：

Cell:　表示一个模块的开始，后面跟模块的名字；

Text:　表示对一个公共参数的定义开始；

Path:　表示该模块各边界的数值范围。图 2 中的 pt 关键字后定义了每条边端点的坐标信息，每一行表示一条从(x1, y1)到(x2, y2)的边。另外，layer 和 width 分别表示该边的层类型和层宽度，一般采用默认值；

Port:　表示线网某节点的位置信息。后跟该节点的名字和横纵坐标值；

net:　表示一条线网。后跟该线网的名字；

PortRef:　表示线网某节点。后跟该节点的名字；

number_of_layers:　布线的层数；

wire_spacings:　导线间的最小距离，每层可以不同，所以它的值是一个列表，每层对应一个值；

via_spacings:　通孔间的最小距离，每层可以不同，所以它的值是一个列表，每层对应一个值；

wire_widths: 导线的宽度，每层可以不同，每层可以不同，所以它的值是一个列表，每层对应一个值；

via_widths: 通孔的宽度，每层可以不同，所以它的值是一个列表，每层对应一个值；

vertical_wire_costs: 垂直方向布线的代价，每层可以不同，所以它的值是一个列表，每层对应一个值；

horizontal_wire_costs: 水平方向布线的代价，每层可以不同，所以它的值是一个列表，每层对应一个值；

via_costs: 通孔的代价，每层可以不同，所以它的值是一个列表，每层对应一个值；

一个线网的例子如下：

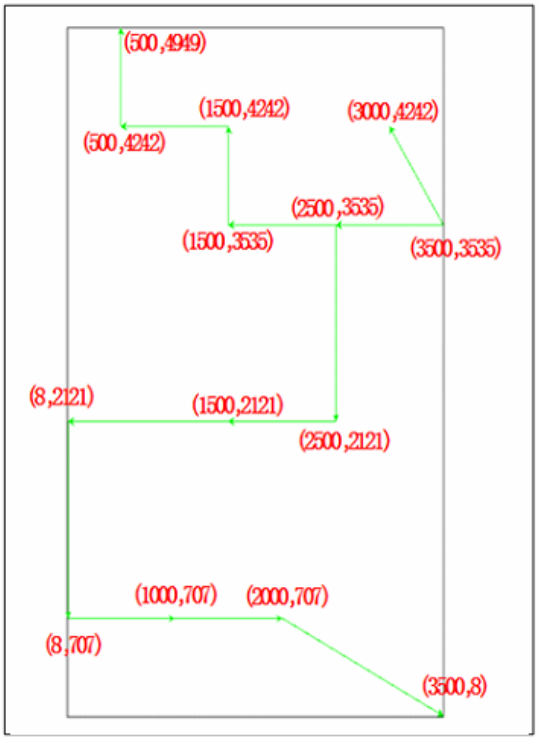


图 3 一个线网的例子

Fig. 3 a net case

该线网对应的 gdf 文件在附录 1 中。

3．2　现有的全局布线阶段的生成树构造方法

我们在第一章的布线算法中提到，在大规模的布线器设计中，布线流程分全局布线和详细布线两个阶段。全局布线是布线的第一个步骤，也是相当关键的一个步骤，因为全局布线结果决定了线网整体的布线走向，而线网拓补结构的好坏直接决定了芯片整体的性能。传统的全局布线算法主要考虑了线长和拥挤等指标**[6][13]**。我们知道，线长直接影响布线的时延。但最短路径布线虽然可以保证源点到所有漏点的距离都是最短的，却会带来布线树总线长的增加，从而增加了芯片面积。

基于最小生成树 MST（Minimal Spanning Tree）的全局布线算法可以使生成的布线树的总线长最短，且关于最小生成树的算法研究相对成熟，所以传统的许多全局布线算法都是基于最小生成树构建的**[6]**。比如台大的 Lin 和 Chang**[9]**提出的多级布线框架中就是采用基于 MST 的 RMST 算法进行全局布线，之后再通过对布线树进行局部修正来满足时序约束。这种方法没有考虑到源点和漏点之间的关系，这会使一些关键路径上的负载过大，从而导致关键路径上的时延不能符合设计要求。而且因为芯片上布线的特殊性，它决定了芯片上布线走向是基于曼哈顿结构的，所以虽然最小生成树算法可以得到总线长最短的布线树，但经过树边的矩形化后，此时的总线长已不是最优的了。

因此，为了寻求总线长最短且具有较好时延性能的布线方法，往往要通过引入“斯坦纳点”，构造斯坦纳布线树。而斯坦纳树的构造是个 NP 难的问题，需要采用各种启发式算法**[14]**。由 Chu 和 Wong[15]提出的基于快速查找表的 FLUTE 算法因有较高的查找效率与易实现性，使其成为流行的斯坦纳树构造方法。

3．2．1 矩形最小生成树 RMST 的构造原理

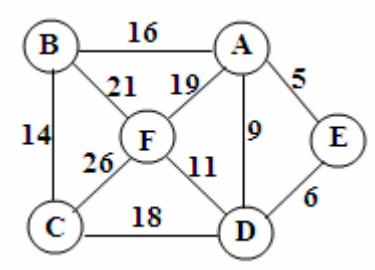
1.　　最小生成树 MST 构造原理

我们知道，具有 n 个顶点的无向网络 N 的每个生成树具有 n-1 条边，于是，问题是如何通过某种方法选择 n-1 条边使它们形成 N 的最小生成树。构成最小生成树的算法主要有 Prim 算法，Kruskal 算法。它们共同的理论基础是：

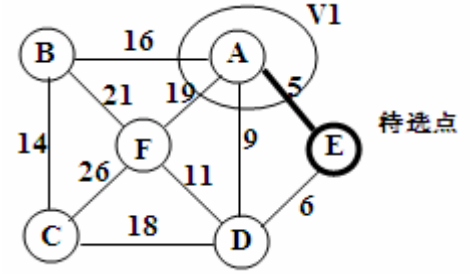
假设 N=（V，E）是一个连通网，U 是顶点集 V 的一个非空子集，若（u，v）是一条具有最小权值的边，其中 u∈U，v∈V－U，则必存在一棵包含边（u，v）的最小生成树。

a） Prim 算法

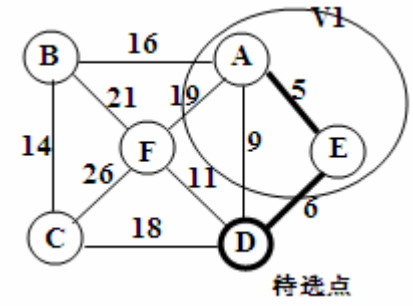
在无向连通图 N=(V，E)中，假设（V1，V2）是顶点集 V 的一个划分，即 V1∪V2 = V，V1∩V2 = ∅，V1 的初始值为{v0}（v0∈V），是最小生成树的顶点集，V2 的初始值为 V。将 V1 与 V2 之间的最短边选入最小生成树的边集 T 中，并将其在 V2 中所连接的顶点的从 V2 中删除，然后并入到 V1 中。如此重复执行，直至 V2 空为止。图 4 说明了 Prim 算法的步骤。



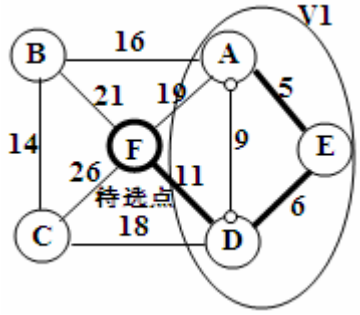
（1）带权的无向连通图



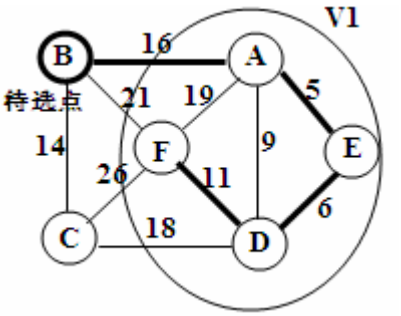
（2）确定最短边 AE 和待选点 E



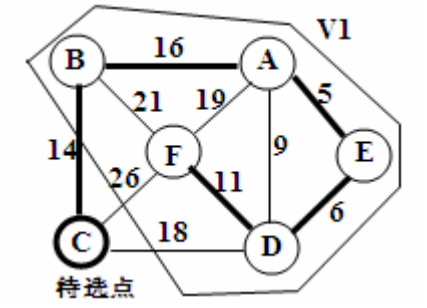
（3）确定最短边 ED 和待选点 D



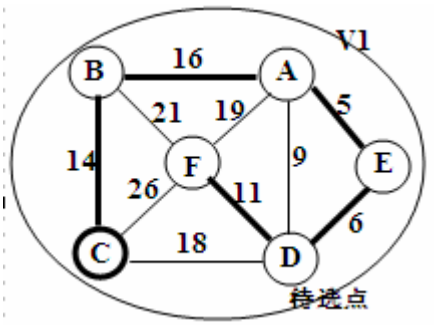
（4）确定最短边 DF 和待选点 F



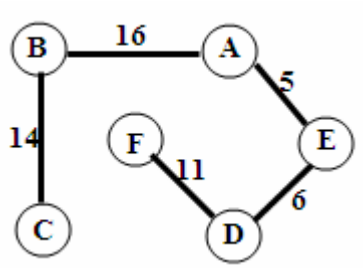
（5）确定最短边 AB 和待选点 B



（6）确定最短边 BC 和待选点 C



（7）确定最小生成树各边和顶点



（8）用 Prim 算法确定的最小生成树

图 4　Prim 算法步骤

Fig.4　Step of Prim algorithm

b)　Kruskal 算法

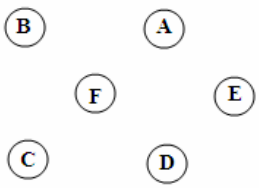
Kruskal 算法是一种按照网中边的权值递增的顺序来构造最小生成树的方法。其基本思想是：在无向连通网 N = (V，E)中，设其含有 n 个顶点，一次性地选出所有点作为最小生成树的结点，然后依次在 E 中选取 n － 1 条边来连接生成树的结点。选取边的原则是：

其权值为 E 中所存边中最小的；

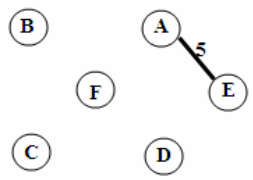
在选取并连接生成树的结点后不能使子网中构成环；

每条边最多只能被选取一次。

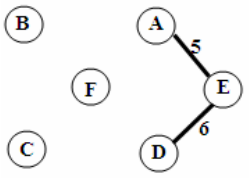
图 5 说明了 Kruskal 算法的步骤。



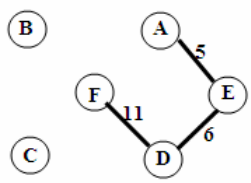
（1）确定最小生成树各边和顶点



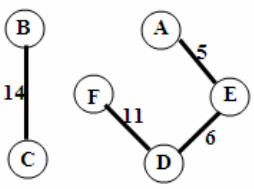
（2）选出最小边 AE



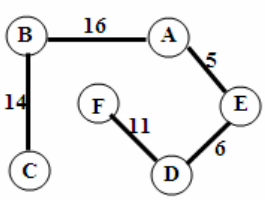
（3）选出最小边 ED



（4）选出最小边 DF



（5）选出最小边 BC



（6）选出最小边 AB

图 5　　Kruskal 算法步骤

Fig.5　　Step of Kruskal algorithm

2.　　基于最小生成树的矩形最小生成树（RMST）方法

大部分构造斯坦纳树的启发式算法是基于寻找最小生成树的，这是由于斯坦纳树与最小生成树之间有着特殊的关系。Hwang 已经论证过最小生成树（MST）的代价与一棵优化的矩形斯坦纳树的代价比不大于 3/2[16]。

设 S 是一条待布的线网，具体的做法是先在平面上产生一个基础网格图 G（S），该 G（S）是通过对 S 中的待连端点进行水平和垂直划线而得到的。如图 6 左图所示，虚线网格就是划分而得得基础网格。之后对该待步线网进行最小生成树布线。很显然，一条给定的线网也许存在多棵 MST 树。根据 Hwang 的结论，一棵近似优化的矩形斯坦纳树可以由 MST 树经过对每条边的矩形化而得到。而树的每条边的矩形转化，因其所采用的方法不同便会得出不同的接近优化的矩形斯坦纳树结构。如果树边矩形化只允许一个直角转弯，这种方式叫做 L 形布线，允许两个直角转弯的方式叫做 Z 形布线。

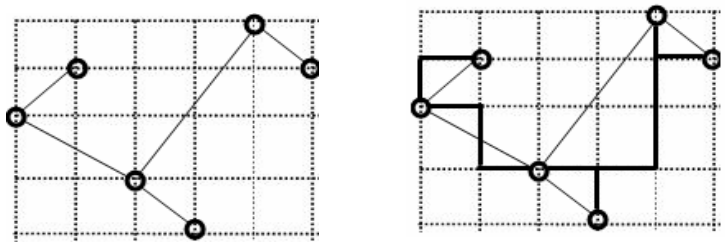


图 6　　最小生成树（MST）与对应的 RMST

Fig.6　　Minimal spanning tree(MST) and its corresponding RMST

3．2．2 矩形斯坦纳树（RSMT）构造方法

对于多端线网的整体布线，斯坦纳树（SMT）算法是一种具有广泛应用的常见算法。当限制其布线走向为曼哈顿正交方向，则称该问题为矩形斯坦纳树（RSMT）问题。RSMT 树构造方法通过引入斯坦纳点，可以使整体布线树的连线总长更短，而线长是全局布线阶段很重要的指标之一。所以，好的全局布线算法往往总是优先构造最小矩形斯坦纳树。

而构造 RSMT 的问题是个 NP 难问题，往往要采用启发式的算法。Hwang 在[16]中对 RSMT 的构造算法做了总结。目前较好的解决方法有 GeoSteiner

package[17]，Griffith[18]和 Mandoiu[19]是两个优秀的近似算法，但其较高的计算复杂度限制了这些算法在大规模布线系统中的应用。

我们观察可以发现，在实际的布线系统中，待布的线网中以端口数较少（<9）的小规模线网为主，因此，相比于设计一个较低时间复杂度的解决方案，算法的简易性和易嵌入性也很重要。一个具有低时间复杂度且易实现的算法可以方便地应用到已有的布线系统中去。

FLUTE（Fast Lookup Table Based RSMT Algorithm）算法是 2007 年由 Chris Chu 和 Yiu-Chung Wong 提出的基于快速查找表的算法。这种方法通过将小规模布线方案（degree < 9）存储在表格里，具体布线时，根据实际的几何信息通过查表得到相应的最优 RSMT 的方法，可以快速有效地处理小规模线网（端口数 < 9）。对于端口数大于 9 的线网，因为它们在所有布线任务中所占比重较小，于是采用 net breaking 的技术，将大线网进行分解成若干小线网，使其可以利用查找表的方式。这种算法的时间复杂度为 O(nlogn)，其中 n 是线网的度数，即端口数。

3．2．3 FLUTE 算法介绍

a. position sequence，POWV，POST 的概念

对一个度为 n 的线网，过每个端口划横竖线，这样就形成一张 Hanan 网格，我们考虑的点是 Hanan 网格上的结点，又称为 Hanan 点。如**图 7** 所示，虚线表示的网格就是 Hanan 网格。假设 x 轴和 y 轴坐标都是递增有序的，那么我们定义 position sequence 为$\{s_1, s_2 \ldots s_n\}$。

其中s_i 为按 y 轴递增方向遍历端口，对应端口在 x 轴的位置序号。也就是说，线网端口的坐标信息可以表示成$\{x_{s_i}, y_i\}$。比如图 7 所示，它的 position sequence 可以表示为{3 1 4 2}。这样，对度为 n 的任意线网，就有 n!组 position sequence。

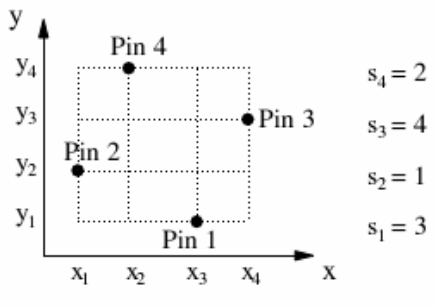


图 7　线网的 position sequence

Fig. 7　position sequence of a net

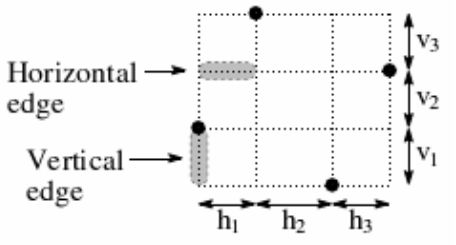


图 8　线网的所有横边和竖边

Fig. 8　all the horizontal and vertical edge of a net

wirelength vector 表示一个向量。它代表 Hanan 网格所有横边和竖边长度的某种线性组合。如图 8 所示，该 Hanan 网格的横边有 3 种取值 h1 h2 h3，竖边也有 3 种取值 v1 v2 v3。那么对应于图 9 中（a）（b）（c）的 3 种线网就有 3 种 wirelength vector。线网（a）总长可以表示为 “1×h1+2×h2+1×h3+1×v1+1×v2+2×v3”,所以它对应的 wirelength vector 就为（1，2，1，1，1，2)。同样道理，（b）和（c）则分别对应（1，1，1，1，2，3)（1，2，1，1，1，1)。注意到有的 wirelength vector 并不能产生线长最优的的布线树，比如（1，2，1，1，1，2)，用它产生的布线树总长一定小于（1，2，1，1，1，1）的结果。所以，将能产生线长最优的 wirelength vector 称为 POWV（potentially optimal wirelength vector)。对图　所示的 position sequence 对应两个 POWV，分别为（1，2，1，1，1，1）和（1，1，1，1，2，1)。具体哪个 POWV 能产生最短线长则取决于 h2 和 v2 的长度。

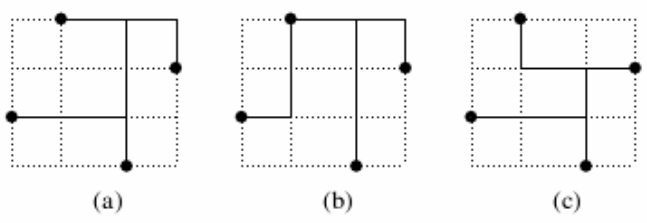


图 9　同一线网对应不同 wirelength vector 的拓补结构

Fig. 9　topological structure corresponding to different wirelength vectors of the same net

和每一个 POWV 对应的拓补结构不是唯一的，如图 10 所示，可能有多个拓补结构对应同一个 POWV，我们将每一个拓补结构称为 POST（potentially optimal Steiner tree）。

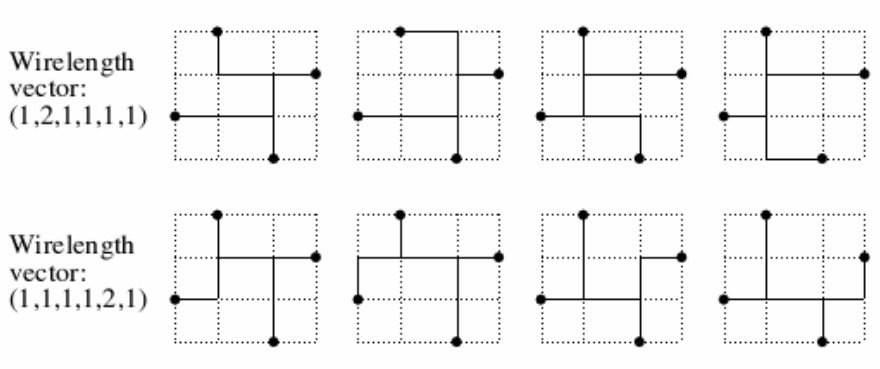


图 10　对应于同一 POWV 的多个拓补结构

Fig. 10　topological structures corresponding to the same POWV

b.　FLUTE 算法的基本思想

FLUTE 算法就是将所有这些 POWV 与 POST 信息预先存放在一个表里。我们根据实际待布线网的端口位置信息求出相应的 position sequence。然后去查表得到对应的 POWV 和 POST，再根据实际 Hanan 网格的横边和竖边长度信息，可以得到不同 POST 下的总线长，最后得到线长最优与拓补结构合适的矩形斯坦纳树结构。Chris Chu 和 Yiu-Chung Wong 还在文章[15]中提到许多对该查找表的优化方法。

3．5　本章小结

本章首先介绍了多级布线框架，并描述了算法输入文件的格式。从对多级布线框架的论述中可以知道，总体布线集成在多级布线的每一级中，因此设计有效的总体布线算法直接影响芯片的整体性能。之后我们讨论了现有的线长最优总体布线算法，这些算法包括如何构造矩形最小生成树，以及矩形最小斯坦纳树。通过论述可以了解到构造最小斯坦纳树会比构造最小生成树具有更优的总线长，而总线长直接影响整体的时延特性。最后我们介绍了构造矩形最小斯坦纳树的高效算法 FLUTE 算法，该算法也是我们后续章节中进行时延优化与串扰优化布线的起点。

第四章　考虑串扰效应与时延的布线方法

在第三章的论述中，我们知道了采用构造矩形最小斯坦纳树的总体布线策略，可以获得线长最优的布线树。因此我们采用最近的基于查找表的 FLUTE 算法**[15]**构造斯坦纳树，保证了最优的初始总线长。之后再采用“移动边”的技术，对该斯坦纳树进行调整以满足时序约束。移动的过程结合基于 SMART 算法的矩求解，并采用更精确的时延模型进行关键路径的时延评估。我们的方法相比于传统的基于 Elmore 时延模型的评估方法，具有更好的精确性。

另外，我们在整体布线树拓补结构确定的前提下，通过基于 SMART 算法的矩求解，采用电路的三阶中心矩作为串扰评估模型，进一步调整局部树边的走向以优化和相邻线网的串扰影响。下面将详细介绍 SMART 算法。基于 SMART 算法的时延驱动布线方法和串扰优化方法也会详细地介绍。

4．1　SMART 算法介绍

4．1．1 电路的矩（moment）和中心矩（central moment）介绍

我们一般将一段互连线等效为分段 RLC 电路，如**图 11** 所示。我们将互连线上某个结点的冲击响应 h(t) 进行拉氏变换，可以得到该点的频率响应：

$$H(s)=\int_0^{\infty} h(t)e^{-st}dt \qquad (4.1)$$

经过 Taylor 展开，$e^{-st}=\sum_{i=0}^{\infty}\frac{(-t)^i s^i}{i!}$，我们可以得到：

$$H(s)=\sum_{i=0}^{\infty}(-1)^i m_i s^i \quad （其中，\quad m_i=\frac{1}{i!}\int_0^{\infty} t^i h(t)dt \quad ）$$

我们称 m_i 为冲击响应的 i 阶矩（ith moment）（i=0,1,2…）为了方便分析，我们假定冲击响应的 i 阶矩的大小都是非负的。并且我们也已知道，其中的一阶矩就是 Elmore 延时[2]。

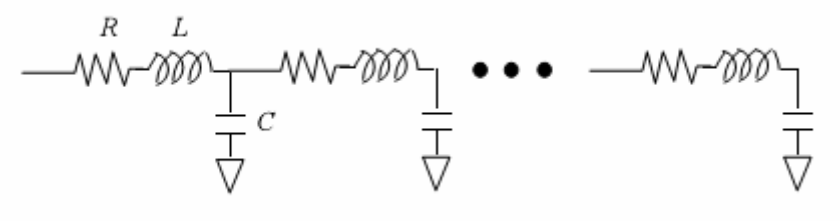


图 11　分段 RLC 电路

Fig. 11　distributed RLC circuit

冲击响应的中心矩（central moment）是由概率论得到的概念。

首先，冲击响应的均值(mean)被定义为：

$$\eta = \frac{\int_0^\infty th(t)dt}{\int_0^\infty h(t)dt} = \frac{m_1}{m_0} \tag{4.2}$$

则冲击响应的 k 阶中心矩就被定义为：

$$\mu_k = \int_0^\infty (t-\eta)^k h(t)dt \tag{4.3}$$

经过简单的推导，前四阶中心矩可以方便地由前四阶矩来表示：

$$\begin{cases} \mu_0 = m_0 \\ \mu_1 = 0 \\ \mu_2 = 2m_2 - \dfrac{m_1^2}{m_0} \\ \mu_3 = -6m_3 + 6\dfrac{m_1 m_2}{m_0} - 2\dfrac{m_1^3}{m_0^2} \end{cases} \tag{4.4}$$

4．1．2　电路中心矩（central moment）的几何意义

根据[2]的论述，电路的高阶中心矩具有特殊的几何意义。其中，我们比较关心的是电路的三阶中心矩，即 μ_3。μ_3的大小可以反映出电路响应波形的信号质量。**图 12** 是 μ_3不同时电路的阶跃响应波形，可以看出，当 μ_3<0 的

时候，输出波形会有明显的振荡，波形的时延减小但输出同时带来了较大的上冲（overshoot）和下冲（undershoot）。而当 μ_3>0 的时候，输出波形没有振荡，但时延增加了。只有在 μ_3＝0 的时候，波形具有较好的质量，此时的波形相比于前两种情况，在时延和振荡幅度之间取得一个较好的平衡点。

电路的三阶中心矩形 μ_3 不但可以反映出输出波形的延迟和振荡，也能反映输出冲击响应波形的对称性。如**图 13** 所示， μ_3＝0 的时候，电路的冲击响应波形是对称的。

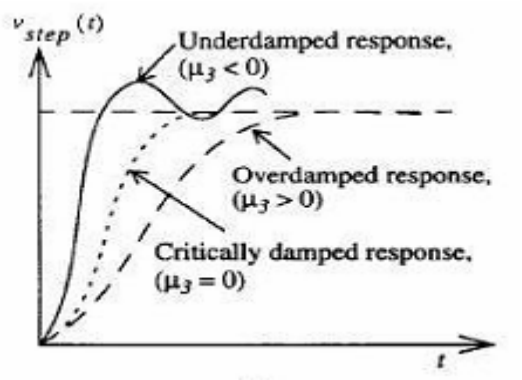


图 12　μ_3 不同时电路的阶跃响应波形 [2]

Fig. 12　step response waveform corresponding to different μ_3

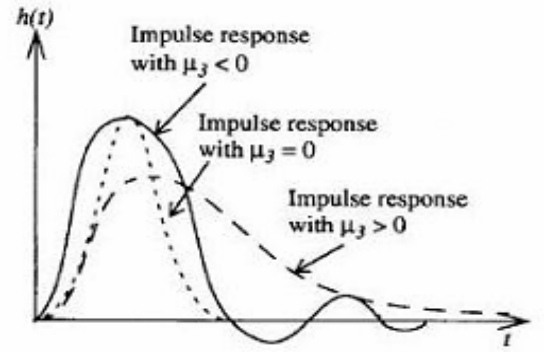


图 13　μ_3 不同时电路的冲击响应波形 [2]

Fig. 13　impulse response waveform corresponding to different μ_3

4．1．3　　对电路中心矩几何意义的验证

既然从电路的三阶中心矩能够反映输出信号的波形质量，我们就可以将三阶中心矩的大小作为我们衡量信号波形质量的一个指标，本文也将利用其作为信号串扰评估的模型来指导实际的布线。J.Cong 在**[4]**提出的 RATS 树算法中采用 $\lambda_i = 4m_i^2 - 3(m_i^1)^2$ 作为波形质量的一个评估模型，在布线中通过让线网各点的 λ_i 接近于 0 来得到最佳的波形质量。这和我们采用三阶中心矩做波形质量评估有相似的结果，J.Cong 在**[4]**中也做过相应的描述。

我们通过实际布线系统在 0.18um 工艺下提取出的一组典型的参数值，利用三阶中心矩做波形质量评估，或者说对串扰评估模型进行验证。如**图 14** 所示，

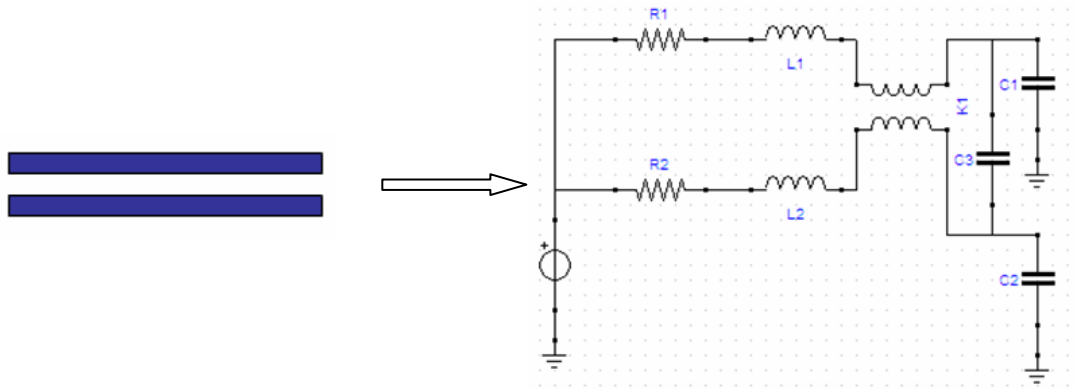


图 14　一段平行互连线及其等效 RLC 模型

Fig. 14　a segment of parellel nets and its equivalent RLC model

图 14 是一段平行互连线的 RLC 电路模型。我们考虑的是相邻互连线上电流方向相同的情形，其中，k1 是互连线之间的互感，C3 是互连线之间的耦合电容。在实际的布线系统中，互连线上的电感以及互连线之间的互感往往是影响波形质量的主要因素，尤其是会带来信号的振荡。因此，我们主要验证在互连线电感值影响较严重的情形下，用电路三阶中心矩进行串扰评估的正确性。

测试参数 1：

R1 = 7.64
R2 = 7.64
L1 = 1.32n
L2 = 1.32n
K1 = 0.72n
C1 = 17.9p
C2 = 17.9p
C3 = 12f

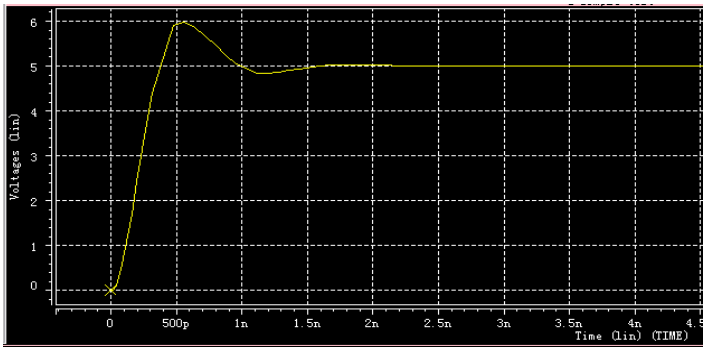


图 15　测试参数 1 与其输出波形

Fig. 15　test parameter 1 and its output waveform

计算得电路得三阶中心矩为：

$$\mu_3 = -6m_3 + 6\frac{m_1 m_2}{m_0} - 2\frac{m_1^3}{m_0^2} = -1.674\times10^{-18}$$

其中电路各阶矩 m_i 的计算方法在下一部分描述。

测试参数 2：

R1 = 7.64
R2 = 7.64
L1 = 5.32n
L2 = 5.32n
K1 = 0.72n
C1 = 17.9p
C2 = 17.9p
C3 = 12f

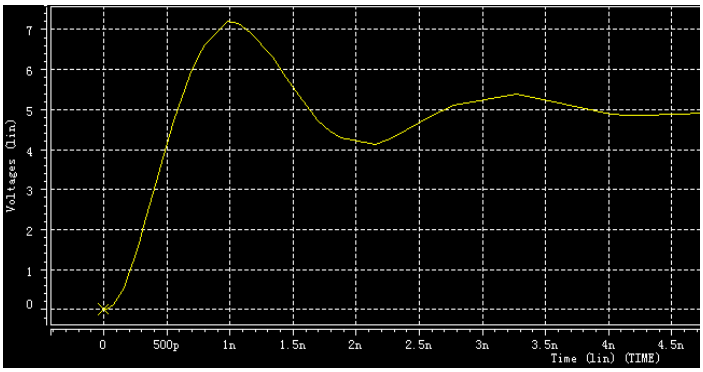


图 16　测试参数 2 与其输出波形

Fig. 16　test parameter 2 and its output waveform

这组测试参数中，我们特意增加了互连线上的电感值，并查看此时的输出波形。
计算得电路得三阶中心矩为：

$$\mu_3 = -6m_3 + 6\frac{m_1 m_2}{m_0} - 2\frac{m_1^3}{m_0^2} = -4.956\times10^{-18}$$

测试参数 3：

R1 = 7.64
R2 = 7.64
L1 = 10.32n
L2 = 10.32n
K1 = 0.72n
C1 = 17.9p
C2 = 17.9p
C3 = 12f

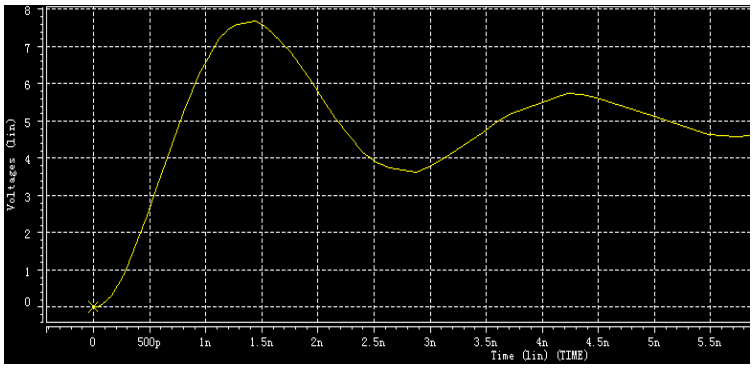


图 17　测试参数 3 与其输出波形

Fig. 17　test parameter 3 and its output waveform

计算得电路得三阶中心矩为：

$$\mu_3 = -6m_3 + 6\frac{m_1 m_2}{m_0} - 2\frac{m_1^3}{m_0^2} = -9.059\times10^{-18}$$

结论：我们对 3 组与实际布线参数数量级相当的参数进行测试，通过不断提高互连线电感值的影响来观察此时的电路三阶中心矩的大小和实际观测输出波形质量的对应关系。第一组参数中，电感的影响最小，此时计算得到的电路三阶中心矩的绝对值最小，对应的波形质量在 3 组测试中最为理想，体现在过冲电压（overshoot）最小，延迟最小。第三组参数的电感影响最大，此时计算得到的电路三阶中心矩的绝对值最大，对应的波形质量在 3 组测试中最差，体现在过冲电压（overshoot）最大，延迟最大。因此我们验证了通过计算电路的三阶段中心矩绝对值的大小，可以反映输出波形的质量。

4．1．4　　耦合 RLC 树状电路各阶矩的数值计算方法

由之前的分析可知，如何快速有效地求出电路的各阶矩是进行更精确时延分析与串扰评估的基础。而计算电路高阶矩的方法主要有“数值”的方法[6]，和“符号化”的方法[7]。

Ratzlaff 和 Pillage 在**[20]**提出 Rapid Interconnect Circuit Evaluation（RICE）算法用以计算 RLC 线网各阶矩。它充分地利用了树状传输线地特殊结构，运用正反两次路径追踪（path-tracing）和一系列电源置换地直流分析反复求

解同样结构的一个电阻网络。基于这种路径追踪算法，又产生了一系列高效的矩的递归数值计算方法。这方面具有代表性的是 Qingjian Yu 和 Ernest S.Kuh 在**[21][22]**提出的计算 RLC 树状电路高阶矩的递归数值计算方法。

图 18 表示一棵 RLC 树状电路，m(i , j)表示第 i 个结点的第 j 阶矩，它表示该结点上的电压值。当求 m(i ， j)的时候，只需把 RLC 树状电路的接地电容替换成电流源，电感替换成电压源，重新进行直流分析，并求出此时结点的电压值即可。其中，对于结点 k 的电容，将其替换成大小为 $C_k \bullet m(k, j-1)$ 的电流源，而电感则替换成大小为 $L_k \bullet \sum_k C_k \bullet m(k, j-2)$ 的电压源。

于是，可以得到求解 RLC 树状电路各结点高阶矩的迭代公式如下：

$$m(i,j) = \sum_{R_l on\ path(src->i)} R_l \sum_{\substack{all\ C_k \\ downstream}} C_k m(k, j-1) - \sum_{L_l on\ path(src->i)} L_l \sum_{\substack{all\ C_k \\ downstream}} C_k m(k, j-2)$$

可见，计算某点的高阶矩，需要利用该点前两阶矩的信息。

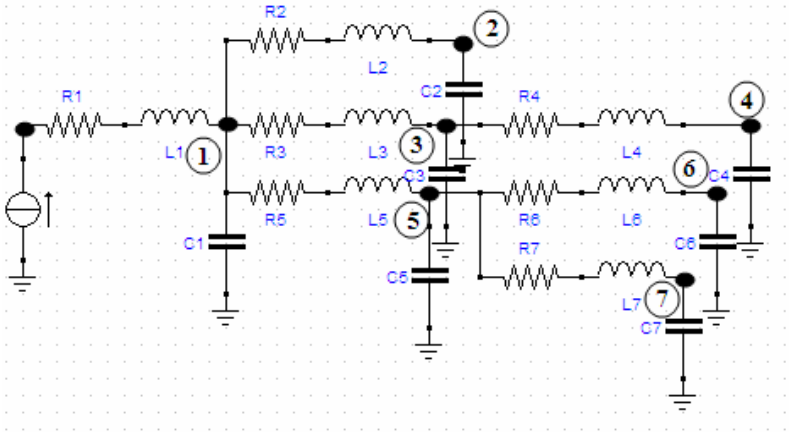


图 18　RLC 树

Fig. 18　RLC Tree

该方法可以扩展到计算耦合 RLC 树的高阶矩。即考虑互连线间的耦合电容与互感效应。此时我们需要根据电源分配理论，在进行矩计算的过程中对其进行相应的电流源或电压源替换。**图 19** 举了个例子表示相邻树边的耦合情况，$K_{1,2}$ 表示相邻树边的互感，$C_{1,2}$ 表示耦合电容。 根据电源分配理论，我们可以将由耦合电容产生的浮动电流源（floating current source）分成两个对地电流源，分别是从结点 1 流向地的 I(j)和从结点 2 流向地的- I(j)。其中，I(j)表示求第 j 阶矩时等效的电流，因为矩的值代表对应结点的电压值，

所以：$I(j)=C_{1,2}(m(1,j-1)-m(2,j-1))$

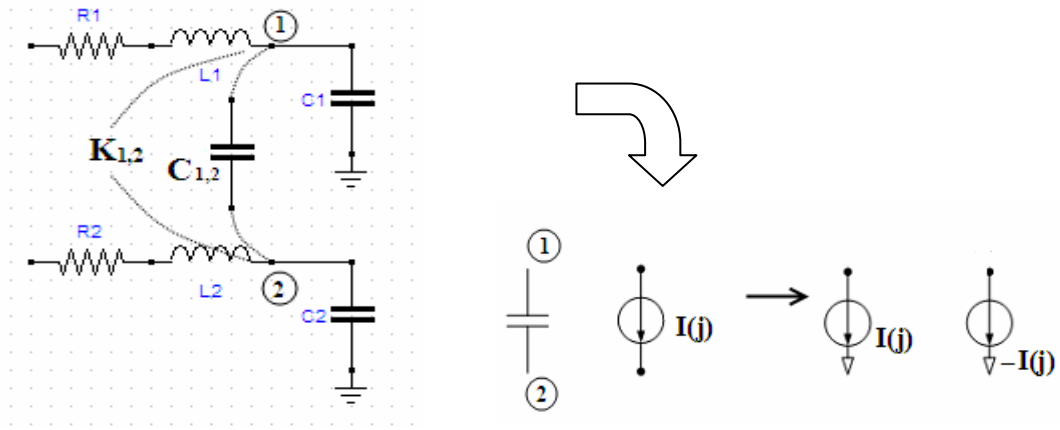


图 19　耦合电容的等效

Fig. 19　equivalence of coupling capacitance

4．1．5　耦合 RLC 树状电路各阶矩的符号化计算方法（SMART 算法）

由上面的分析可知，用数值迭代的方法可以计算电路各阶矩。但其复杂性在于矩计算时候必须识别所有相关的路径和路径上的电路元件。对于大规模电路而言，必须要有特定的机制记录相关的追踪路径。Ratzlaff 和 Pillage**[20]**提出的虚拟路径追踪（virtual path-tracing）算法在计算一阶矩的同时，用伪指令（pseudo-instruction）存储树枝的位置和阶数。然而这种方法需要复杂的编程实现，不易于符号化（symbolic）。

Shi**[5]**提出基于矩决策图 MDD（Moment Decision Diagram）的单棵 RLC 树状电路矩的符号化计算方法 SMART。它利用二叉决策图共享子图的特点，高效地将电路结构存储为图的数据结构。图的结构与实际布线结构相对应，计算结果可以直接用来指导布线。Shi 又将该算法进行扩展以实现针对多棵耦合 RLC 树高阶矩的计算。

SMART 算法的关键是构建一棵矩决策图 MDD 和矩的计算过程，下面通过一个示例来说明。该例针对图 7 的 RLC 树进行说明。

首先是 MDD 的构建，这个过程主要分为两个步骤：

步骤 1：　构建和原 RCL 树拓补结构相同的电容树。如**图 20** 所示。

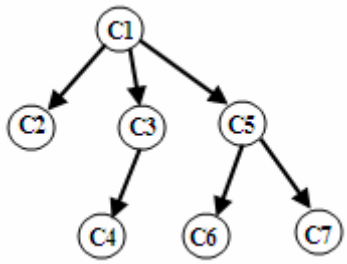


图 20　和 RLC 树拓补结构相同的电容树

Fig. 20　C-tree of the same topology with RLC tree

步骤 2：　构建一个多根的二叉决策图 BDD。如图 21 所示。该 BDD 和电容树有相同的拓补结构，特点是它的每个 Ri 都是 BDD 的根节点，它的”Then”边指向和其同序号的 Ci，”Else”边指向父节点 Rj。其中 R1 的”Else”边指向 0。

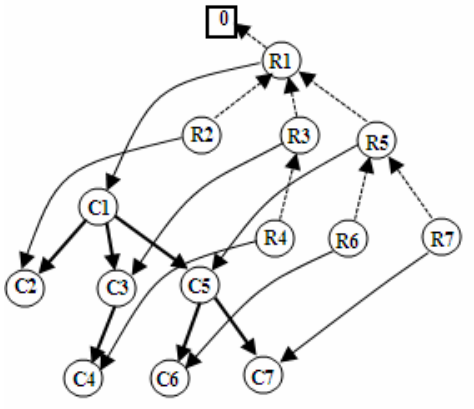


图 21　矩决策树（MDD）

Fig. 21　Moment decision diagram (MDD)

矩计算的过程也分成两个步骤：

步骤 1：　后序遍历电容树，使电容树中各节点存储其子树结点的电容值的和。

比如对于图 构建的 MDD，我们可以电容树中各结点的值为：

cap7[1] = C7,

cap6[1] = C6,

cap5[1]= C5+C6+C7

cap4[1] = C4,

cap3[1]= C3+C4,

cap2[1]= C2,

cap1[1] = C1 + C2 + C3 + C4 + C5 + C6 + C7

步骤 2：　BDD 计算。BDD 中的每个根节点 k 阶矩的值为：

val(Ri) [k] := Ri * val(Ri → Then)[k] + val(Ri → Else) – Li * val(Ri → Then)[k-1] ,

其中，val(Zero) = 0，val(Ri → Then)[k] = capi[k]，

通过步骤 1 和步骤 2，第一次遍历后就可以求出电路的一阶矩。对于更高阶的矩，只需要稍微修改电容树，将电容数中节点的电容值 Ci 改为 Ci * m(i,1)，再重复步骤 1 和步骤 2，就可以计算 i+1 阶的矩。

4．2 基于 SMART 算法的时延驱动布线

4．2．1 D2M 时延模型

已经证明，Elmore 时延模型的本质是电路的一阶矩**[8]**。它对快速变化的信号具有较低的精确度，这体现在用 Elmore 模型估算出的时延往往具有靠近源点的结点延时偏大，而远离源点的结点延时才相对准确的特点。Elmore 模型的这个特性制约了它在深亚微米工艺下的应用。而各种基于电路高阶矩分量的延时模型得到广泛重视**[6]**。

D2M 延时模型**[23]**是一个经验公式，它只利用了电路的一阶矩和二阶矩，却得到了比 Elmore 模型更准确的时延估计。D2M 的公式是：

$$D2M = \frac{m_1^2}{\sqrt{m_2}} \ln 2 \qquad (4.5)$$

表 1 是我们利用 D2M 和修正的 Elmore 模型对大规模 RC 树上各结点的时延估计进行比较。我们将其与 SPICE 仿真得到的值进行对比。其中，表格中结点的编号按照与源点的距离大小进行递增编号，结点 1 离源点最近而结点 20 离结点最远。从表格中我们可以发现，Elmore 模型对离源点近的结点，其时延估算偏大，而 D2M 模型算出的值介于真实值和 Elmore 估算的值之间。Elmore 模型对距离源点较远结点的延时估算比真实值偏小，而 D2M 的延时估算都比较准确。

表格 1 D2M 模型与 Elmore 模型的时延估算值比较

Node	Spice	ln2 * m1	D2M
	$Delay(\times e^{-9})$	$\ln(2) * m_1(\times e^{-9})$	$(\times e^{-9})$

1	1.175	11.42	5.80
2	4.97	19.57	12.90
3	5.57	19.99	13.40
4	10.2	26.89	20.54
5	21.2	33.80	28.67
6	33.8	40.29	37.03
7	44.2	46.27	45.30
8	45.1	46.83	46.12
9	46.8	47.94	47.74
10	48.8	49.19	49.60
11	49.88	50.02	50.85
12	50.5	50.43	51.47
13	52.0	50.93	51.97
14	56.6	53.93	56.40
15	58.5	55.17	58.28
16	59.7	56.01	59.55
17	60.3	56.42	60.18
18	58.6	55.26	58.40
19	60.4	56.48	60.25
20	61.1	56.98	61.01

4．2．2　　时延驱动斯坦纳树布线算法

由之前的分析可知，构造线长最优的斯坦纳全局布线树是关键。FLUTE 算法因其具有较低的时间复杂度，且算法简单，易集成。所以我们在布线的初始阶段，运用 FLUTE 算法得到一棵线长最优的布线树。但 FLUTE 算法是以线长最短为布线目标的，它没有考虑到源点和漏点的关系，这可能造成关键路径上负载过大，从而导致关键路径上的时延不满足约束条件。因此需要对布线结果进行调整。

我们的调整方法针对“可移动边”进行，并且结合 SMART 算法实时对布线过程进行时延估计，从而得到最优的布线方案。“可移动边”的技术最早在**[24]**中提出。首先，可移动边必须是在一棵斯坦纳树中连接一对斯坦纳点的树边。如**图 22** 所示，它是一棵连接 4 个结点的斯坦纳树，图中 S1 与 S2 表示斯坦纳点。则连接 S1 和 S2 的树边为可移动边，在**图 22** 中用红边进行了标识。当对可移动边在移动区域内进行平移时，不会改变斯坦纳树的拓补结构。比如**图 22** 中，我们在允许的范围内对 S1S2 进行水平移动，并不会改变该斯坦纳树的拓补结构。可移动区域的大小可以通过斯坦纳点与其临近的线网结点的位置关系计算出来。

如**图 23** 所示，先计算出每个斯坦纳点的可移动范围。斯坦纳点的可移动范

围介于与它临近的两个结点之间。在**图 23** 中，斯坦纳点 S1 的可移动范围为 D1，斯坦纳点 S2 的可移动范围为 D2。则可移动边的可移动范围为它的两个端点的可移动范围的交集。**图 23** 中，S1S2 的可移动范围则为 D，在图中用红色箭头做了标识。

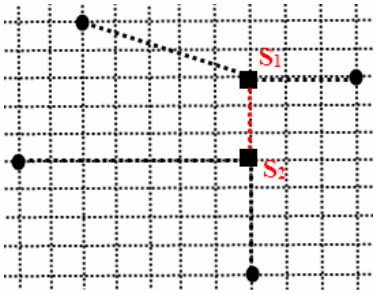


图 22　斯坦纳树与可移动边

Fig. 22　Steiner tree and movable edge

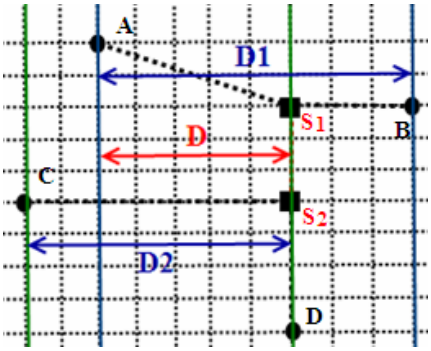


图 23　可移动边的可移动区域

Fig. 23　movable edge and its movable region

我们的方法首先将每条树边用 RLC 电路等效，如**图 18** 所示。其中，R，L，C 的值与树边的曼哈顿距离成正比。并已事先知道线网单位长度的 R，L，C 值。利用 SMART 算法得到各个漏点的矩，结合 D2M 模型，从而计算出从源点到各个漏点的延时。之后找出关键路径，即不满足时序约束的路径。比如**图 24** 所示从 S 到 T 的路径，之后在可移动范围内移动“可移动边”，每次移动的距离为一单位网格。每移动一次“可移动边”，就计算一次关键路径上的时延，直至时延满足时序约束。

该方法在对关键路径上的可移动边进行调整的时候，因为不破坏整体布线树的拓扑结构，则不用重新构建矩决策图（MDD），从而可以快速得到各结点更新过后的矩信息。

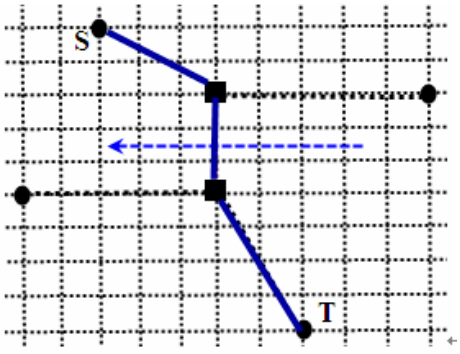


图 24　通过调整“可移动边”对关键路径进行时延优化

Fig. 24　Shifting “movable edge” to do timing optimization for critical path

4．3　基于 SMART 算法的串扰优化

我们知道，随着芯片上布线密度的增加，相邻线网之间的相互影响是不能忽略的，相邻线网主要是通过耦合电容和互感来影响自身线网，产生串扰，而串扰不但会影响波形质量，也带来时延恶化。

4．3．1　　要解决的问题

我们要解决的基本问题如**图 25** 所示。AB 为待布的二端线网，连接 AB 的二端线网可以有多种走向，我们只考虑 L 型与 Z 型模式。图中箭头表示线网上的电流方向。从图中可以看出，AB 连线会和相邻线网产生不同程度的耦合，如图中 C1 和 C2 区。这些耦合区会对 AB 上传输的信号产生串扰。所以，我们要解决的问题是，选择连接 AB 最佳的一种走向，使得它与相邻线网产生的串扰影响最小。

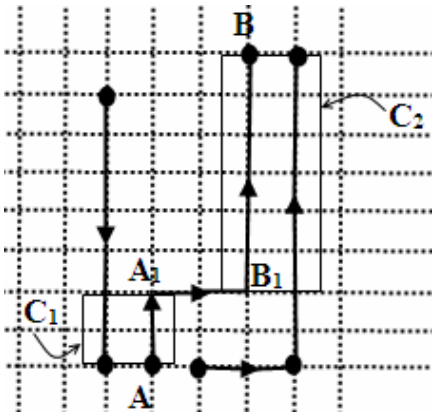


图 25　二端线网的连接与耦合区

Fig. 25　two-pin net and its coupling region

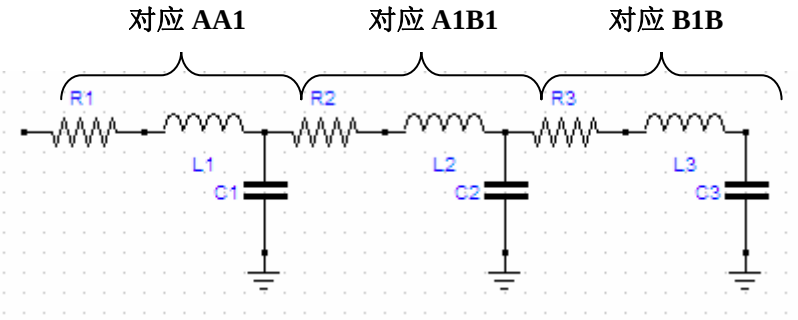


图 26　二端线网的等效模型

Fig. 26　two-pin net and its equivalent model

4．3．2　二端线网的 RLC 等效

为了对图 25 中二端线网 AB 进行建模，我们将其等效为 3 段 RLC 电路串联。如**图 26** 所示。分别对应 AA1，A1B1，B1B。我们已知单位长度的电阻 r，单位长度电感 l，以及单位长度电容 c。则每段 RLC 电路的元件值大小均与其曼哈顿距离成正比。**图 25** 中，设单位网格的边长为 h，则 AA1 段等效电阻 R1 = 2 * h * r，电感 L1 = 2 * h * l，电容 C1 = 2 * h * c。

另外，我们也考虑耦合区的耦合电容。如**图 25** 中，因为我们只考虑相距一个网格的线网间的耦合影响，所以图中的耦合区有 C1 和 C2 两部分。我们知道，耦合电容的大小和相邻平行线的重叠长度成正比，而和它们之间的间距成反比。所以可以设单位耦合长度的耦合电容为 cc，则图 22 中耦合区 C1 的耦合长度为 2 * h，对应的耦合电容 Cc1 = 2 * h * cc，耦合区 C2 的耦合长度为 6 * h，对应的耦合电容 Cc2 = 6 * h * cc。

当我们需要考虑耦合电容对关键线网 AB 的影响时，还需要通过密勒等效【数电课本】，将线网间的耦合电容，拆分成接地电容。对于耦合区 C1，因为相邻线网的电流方向相反，则等效电容为两倍的耦合电容值，即此时的 Cc1 = 4 * h * cc。对于耦合区 C2，因为相邻线网的电流方向相同，则此时的 Cc2 = 0。考虑了耦合影响后，我们对二端线网的等效模型为**图 27** 所示。应该说，这种等效不能很精确地等效实际的耦合值，这在全局布线阶段也是不必要的。但它可以粗略反映线网间耦合影响的程度，在全局布线阶段，指导走线绕开串扰影响严重的区域。

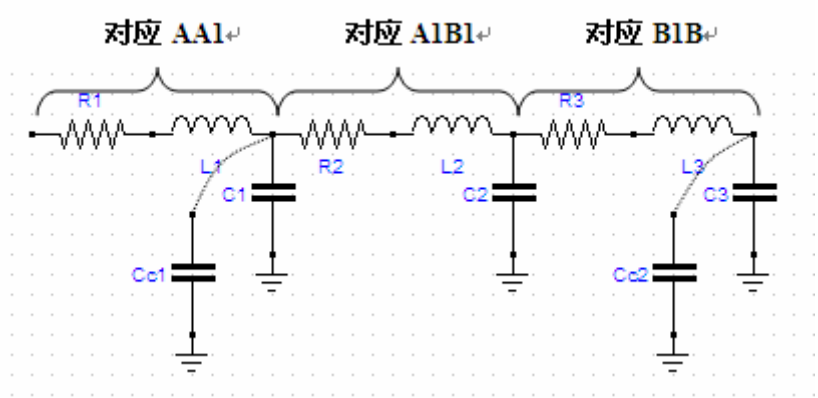


图 27　考虑耦合效应的二端线网等效模型

Fig. 27　two-pin net's equivalent model considering coupling effect

4．3．3　基于 SMART 的串扰优化算法

1．　代价函数的选择

由前面分析可知，根据二端线网实际走向的不同，其等效 RLC 模型的电路参数也随着变化，而电路参数的变化直接影响了输出节点的高阶矩。按照 3.2 节中对利用电路三阶中心矩进行串扰评估的验证，我们知道，电路三阶中心矩 μ_3 绝对值的大小能反映出波形的质量。当 μ_3 的绝对值接近于 0 时，波形具有较小的振荡。由之前的分析也知，当考虑了电感因素，电路的三阶中心矩往往是负数，从而导致波形的振荡。因此，我们在确定布线走向的时候，应该要考虑其引起的电路三阶中心矩的变化。所以应该定义一个代价函数作为判断布线走向优劣的依据。我们的布线目标是，让线网所有结点的电路三阶矩都接近于 0。因为考虑电感因素时，电路三阶矩往往是负数，所以我们定义的代价函数取所有结点三阶中心矩的几何平均。如下所示：

$$C_{cost} = \sqrt{\frac{\sum_{i=1}^{n} (\mu_3^i)^2}{n}} \qquad (4.6)$$

我们设要处理的生成树的节点数为 n。其中 μ_3^i 表示第 i 个节点的三阶中心矩。

2. 算法描述

算法输入： 已满足基本时延约束的初始布线树

我们利用之间介绍的基于 SMART 算法的时延驱动布线方法，得到一棵已满足基本时序约束的斯坦纳布线树。布线结果如**图 28** 所示。

算法输出： 得到各节点信号波形质量最优的布线树

步骤 1： 构造初始 MDD 树。

从源点开始，将其依次拆分成各个二端线网。

我们对得到的每一个二端线网先用 L 型模式布线。并根据之前描述的二端线网的 RLC 模型等效，将其用 RLC 电路等效，R，L 和 C 值的大小既考虑到两端点之间的曼哈顿距离，也考虑和相邻线网的耦合关系。之后我们就可以得到一棵表示该网络的 RLC 树，并根据该 RLC 树的拓补结构构造对应的 MDD。如**图 29** 所示。

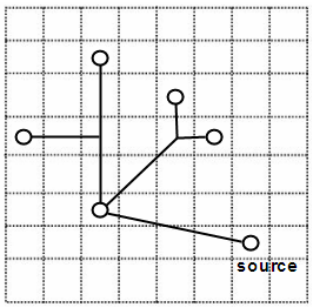


图 28 算法输入（满足基本时延约束的生成树）

Fig. 28 Input of algorithm (spanning tree satisfying the basic timing constrain)

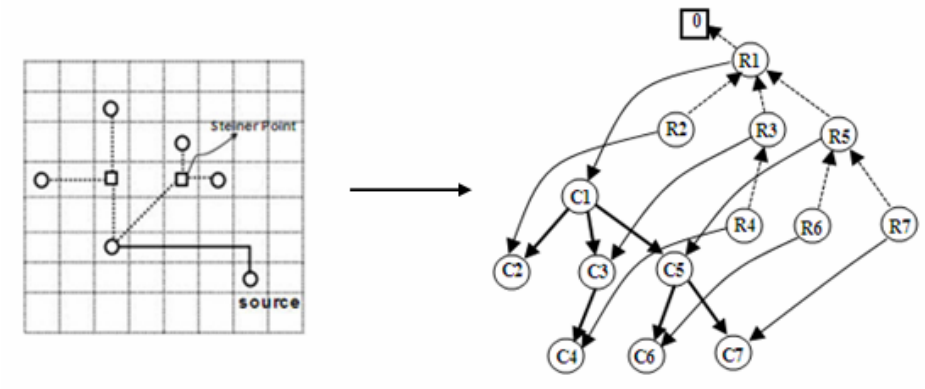


图 29 所构造的初始 MDD 树

Fig. 29 Initial MDD tree constructed

步骤 2： 计算各个节点的高阶矩，并求出初始代价函数

我们在 MDD 上通过 BDD 的计算过程，首先求得每个节点的高阶矩。并根据式（**4.6**）求出每个节点的三阶中心矩 μ_3^i。计算此时的代价函数（**4.6**），并设此时的代价函数为 C1。

步骤 3： 从源点开始，按照深度优先的方式，选取第一条二端线网。

步骤 4： 调整该二端线网的连接方式。调整的方式如**图 30** 所示。即由初始的 L 型布线模式调整为 Z 型布线模式。并重新计算此时该二端线网 RLC 等效模型的参数值。

步骤 5： 修改 MDD 树对应节点的参数值。重新计算此时该线网各节点的高阶矩和中心矩。计算此时的代价函数，设此时代价函数为 C2。之后回到步骤 4，重新调整连接方式，并通过步骤 5 计算对应的代价函数。设一共有 k 种连接方式，对应的代价函数分别为 {C1, C2, C3,…, Ck}。

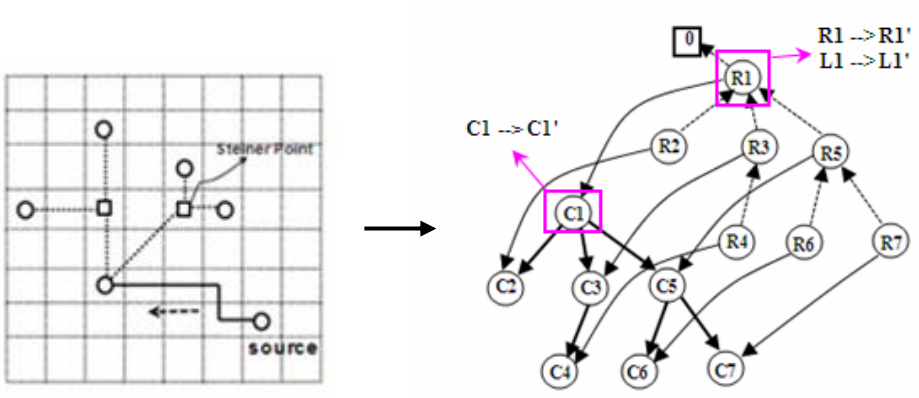


图 30 调整布线走向后和对应的 MDD 树

Fig. 30 MDD tree after rectifying routing orientation

步骤 6： 选取{C1, C2, C3,…, Ck}中的最小值，并将其连接方式确定为该二端线网的布线方式。重复步骤 3，选取下一条二端线网。

该算法通过调整线网的局部二端线网的连接方式，寻找串扰最优的，也即各点信号质量最优的布线树。在调整的过程中，所构造的 MDD 树拓补结构没有变化，而只是如算法中步骤 5 那样改变 MDD 树节点的值。因为不用重复地构造 MDD 树，使算法能很快地进行。

4．4 本章小结

本章详细描述了 SMART 算法。SMART 算法利用图的结构，能巧妙地快速计算电路的高阶矩。图的结构与布线结构相对应，计算结果可以直接用来指导布线。因此本章也介绍了电路矩以及电路中心矩的概念。利用电路的高阶矩，可以得到更精确的时延估计模型，比如 D2M 模型。利用电路三阶中心矩，也可以用来刻划电路的串扰效应。用电路三阶中心矩刻划串扰效应也在本章中进行了验证。之后该章节论述了结合 SMART 算法进行时延优化与串扰优化的总体布线方法。

第五章　实验数据与实验结果

通过以上布线算法的研究，结合其他同学在快速矩计算的研究成果，编制了一个图形界面的自动布线界面，以实现时延与串扰优化的布线。

5．1　程序的图形化显示

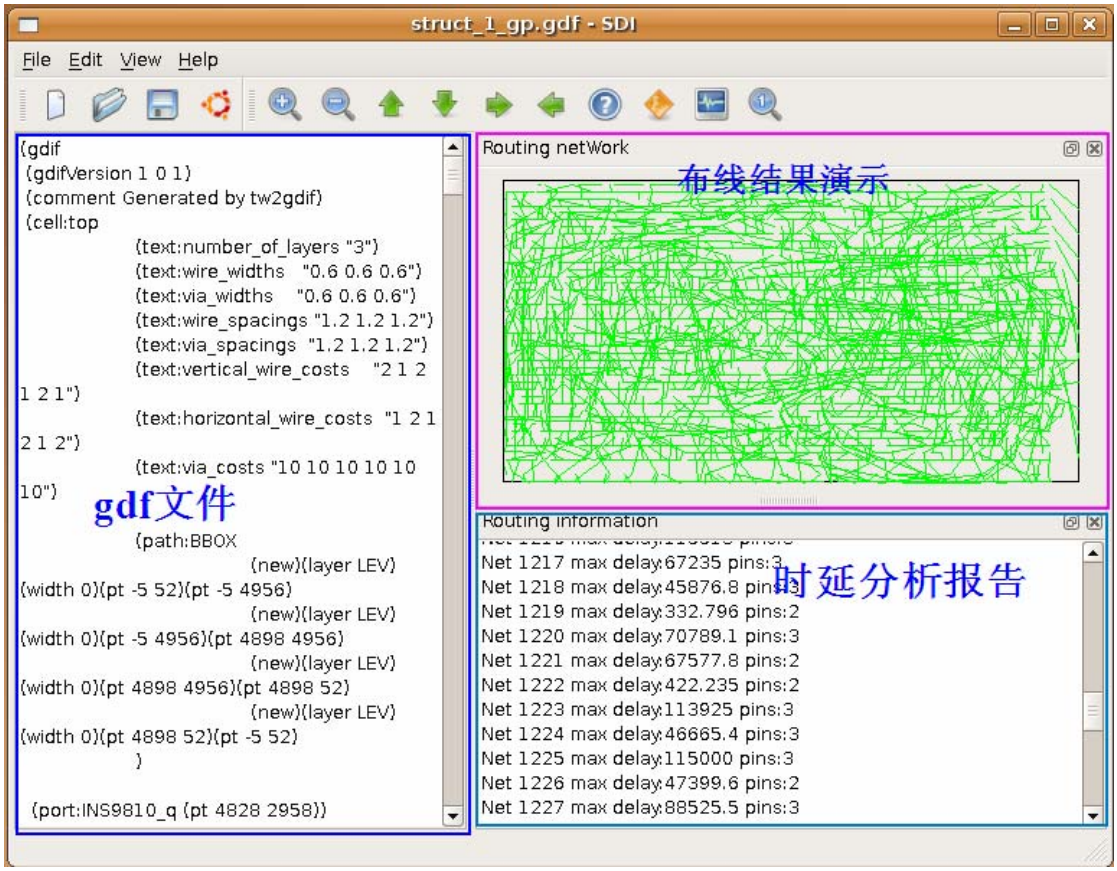


图 31　程序的图形化显示

Fig. 31　graphical interface of the program

该布线器的开发是在 ubuntu 7.04 下，采用 Qt4 软件包，用 C++语言编写完成。算法实现是基于部分开放的源代码，它们是台湾大学 CAD 实验室提供的基于多级布线框架的部分源程序**[9]**和 LEDA 算法软件包**[10]**。整个软件界面分成 3 个区域，左边的是输入文件（gdf 文件）显示区，右上角为布线结果演示区，右下角为针对各个线网的时延分析报告。

5．2 数据测试

我们的测试实例采用 UCLA CAD 实验室提供的测试例子，以及 ISPD04 IBM Benchmark**[12]**的 ibm01 测试电路。我们对 ibm01 测试电路进行基于 SMART 的时延驱动布线方法测试，针对 UCLA CAD 提供的测试例子，我们则进行基于 SMART 的时延驱动布线与串扰优化布线。

测试平台：

计算机的 CPU 是 Intel Celeron（R） 1.86G，32 位，内存为 2GB。

5．2．1 基于 SMART 算法的时延驱动布线测试

测试用例：

我们选取 ibm01 测试电路做测试用例。它由 13000 条线网组成。

测试方法：

我们根据线网的规模进行分组测试。在每一组中，我们随机选择一定数量的线网作为关键线网，比较其在优化前与优化后的线网平均时延。从改进率可以看出，我们的方法较好地改善了时延。另外，我们针对引脚数目为 4 和 5 的线网组，挑选单根线网观察它在不同布线算法下的布线结果和时延。

测试结果：

表格 2 基于 SMART 算法的时延驱动布线测试

线网组	线网数	关键线网数	优化前的平均时延	优化后的平均时延	改进率
Pins 4	1044	38	364.8	210.5	42.3%
Pins 5 – 7	1379	31	766.7	550.5	28.2%
Pins 8 - 10	574	22	715.1	631.4	11.7%
Pins 11 - 13	371	16	1414.9	1141.8	19.3%
Pins 14 – 16	142	8	418.8	237.5	43.3%
Pins 17 - 21	99	4	1096.9	908.2	17.2%

测试线网 1：

采用最小生成树布线的结果（优化前的布线方法）：

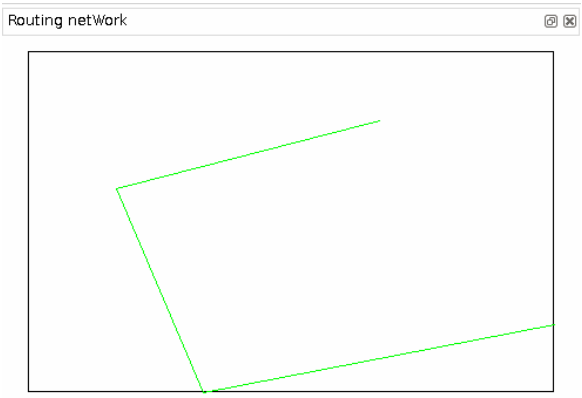


图 32　采用最小生成树布线的结果（线网 1）

Fig. 32　Routing result using MST (net 1)

线网最大延时：　372.8 ns

采用 FLUTE 算法布线的结果（优化过程中的阶段性结果）：

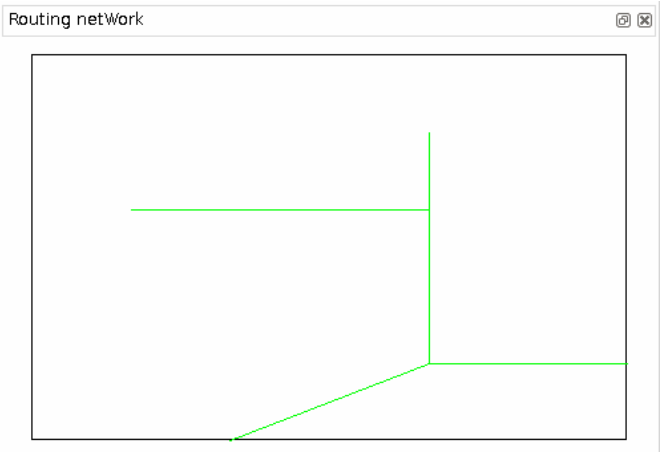


图 33　采用 FLUTE 算法布线的结果（线网 1）

Fig. 33　Routing result using FLUTE algorithm (net 1)

线网最大延时：　232.3 ns　　改进率：37.7%

采用基于 SMART 算法进行“可移动边”调整后的结果（优化后的结果）：

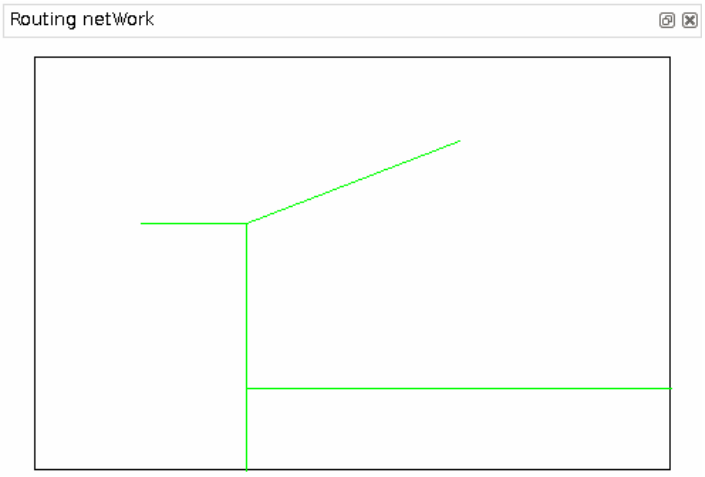


图 34　采用基于 SMART 算法进行“可移动边”调整后的结果（线网 1）

Fig. 34　Routing result using SMART based “movable edge” technology (net 1)

线网最大延时：　217.7 ns　　改进率：41.6%

测试线网 2：

采用最小生成树布线的结果（优化前的布线方法）：

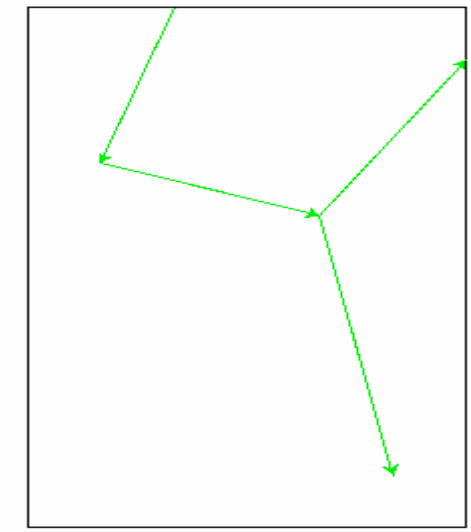


图 35　采用最小生成树布线的结果（线网 2）

Fig. 35　Routing result using MST (net 2)

线网最大延时：　729.2ns

采用 FLUTE 算法布线的结果（优化过程中的阶段性结果）：

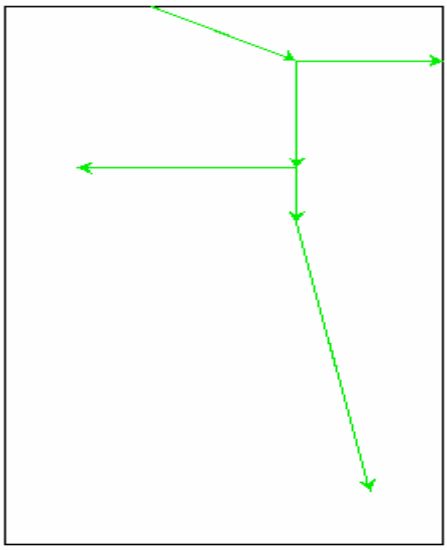


图 36　采用 FLUTE 算法布线的结果（线网 2）

Fig. 36　Routing result using FLUTE algorithm (net 2)

线网最大延时：　571.7ns　　改进率：21.6%

采用基于 SMART 算法进行“可移动边”调整后的结果（优化后的结果）：

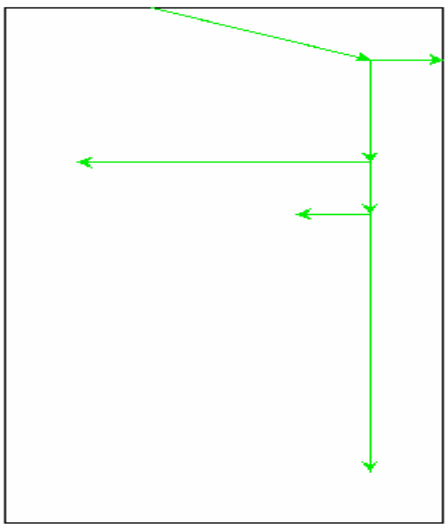


图 37　采用基于 SMART 算法进行“可移动边”调整后的结果（线网 2）

Fig. 37　Routing result using SMART based “movable edge” technology (net 2)

线网最大延时：　500.9 ns　　改进率：31.3%

测试结论：

我们通过对 ibm01 标准测试电路进行分组测试，并随机挑选任意线网观察它们在优化前后的布线结果与时延报告，可以看出，未做优化前，因为采用的是最小生成树布线，没有考虑到源点和漏点的关系，导致关键路径的负载过大，从而恶化了时延。而我们的布线方法的起点是先利用 FLUTE 算法构建斯坦纳生成树，通过引入斯坦纳点，它在总线长方面具有比最小生成树布线更大的优势。但 FLUTE 也没有考虑关键路径上的时延优化，我们通过基于 SMART 算法与“可移动边”的方法调整布线，使线网的延时得到极大的改善。

5．2．2　　基于 SMART 算法的串扰优化测试

测试用例：

我们采用 UCLA CAD 提供的一组标准测试电路，进行布线测试。

测试方法：

我们在多级布线器(MR)中**[9]**集成了我们设计的全局布线算法。而原有的多级布线器是不考虑串扰优化的，只是基于最小生成树的布线。出于比较的直观性，我们在基于最小生成树布线的基础上，进行利用 SMART 算法做串扰优化。从而观察优化前后线网在信号上升时间与过冲电压方面的改善情况。具体的做法是：我们随机挑选一组线网，观察这组线网中每一条线网在优化前后的布线情况，通过导出它的网表文件，进行 HSPICE 仿真以观察它们在平均上升时间与平均过冲电压方面的改善情况。

测试结果：

1.　整体布线结果

我们针对 UCLA CAD 提供的 struct_1.gdf 进行布线。图 38 是原有多级布线器（MR）基于最小生成树的布线结果，图 39 是在最小生成树布线结果基础上，采

用基于 SMART 算法的串扰优化布线方法得到的布线结果。从直观上可以看出，图 39 线网分布比较均匀，而图 38 线网分布相对拥挤。可见经过优化后，有效地将线网从串扰严重的区域移开。

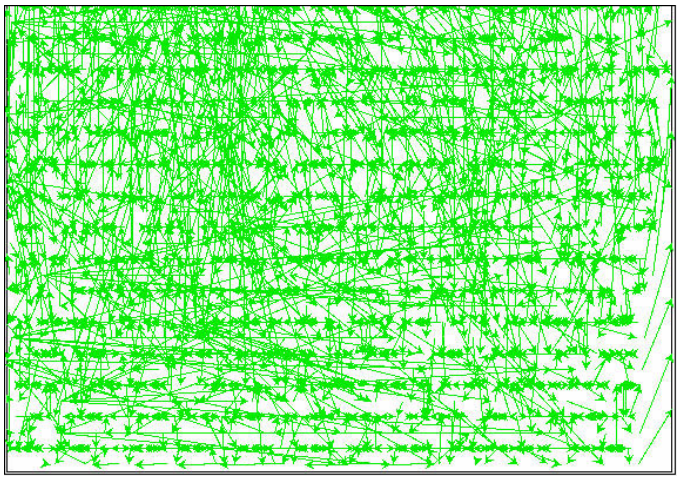


图 38　串扰优化前的布线结果

Fig. 38　Routing result before crosstalk optimization

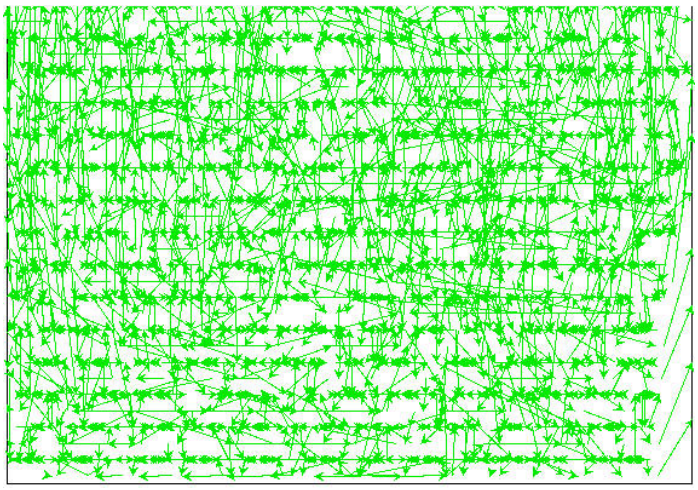


图 39　串扰优化后的布线结果

Fig. 39　Routing result after crosstalk optimization

2. 上升时间和过冲电压的比较

挑选的关键线网数：20

表格 3 基于 SMART 算法的串扰优化测试

电路	平均上升时间（ns）			平均过冲电压(%)		
	MR	优化后	Imp(%)	MR	优化后	Imp(%)
Struct	0.38	0.31	22.6	44.7	32.8	36.3
S5378	0.24	0.19	18.2	31.9	21.3	33.1
S9234	0.31	0.24	21.9	37.1	26.5	28.6
Primary1	0.27	0.21	20.7	36.2	23.3	35.7
Primary2	0.62	0.51	18.3	42.5	29.2	31.2

测试结论：

从整体布线结果看，我们基于 SMART 算法的串扰优化有效地均匀线网分布。从统计的测试结果数据来看，我们的方法有效地降低了线网的上升时间，改善了过冲电压。

5．1 本章小结

本章先是对程序的图形化界面进行演示，并介绍了该软件界面的组成部分。之后选用 UCLA CAD 和 ISPD 04 的 ibm01 测试电路进行基于 SMART 算法的时延驱动布线方法以及串扰优化布线的测试。在测试基于 SMART 算法的时延驱动布线方法时，我们选的是 ibm01 测试电路，在对测试数据做统计分析之后，我们随机挑选了个别线网演示了算法对树结构的调整过程。在测试基于 SMART 算法的串扰优化时，我们选择基于 UCLA CAD 提供的测试例子进行测试统计，主要统计串扰优化前后线网的平均上升时间和平均过冲电压方面的改善情况。结果证明，我们的方法在时延和串扰优化方面都有较好的提高。

第六章　总结与展望

6．1　主要结论

本文在多级框架的基础上，以时延优化和串扰优化为两个主要的优化目标。提出了基于符号化矩计算算法（SMART 算法）的总体布线方法。之所以利用SMART 算法，是为了快速得到电路的高阶分量，从而能进行更为精确的时序估计与串扰评估。而且 SMART 算法巧妙地将矩信息存在于树结构中，树结构与实际布线树结构一致，计算结果可以直接用来指导布线。针对时延优化和串扰优化两个指标，我们分别提出基于 SMART 算法的时延驱动布线方法和串扰优化方法。对于时延驱动布线，我们采用**[24]**提出的“可移动边”技术，在先利用**[15]**提出的 FLUTE 算法构建线长最优的矩形斯坦纳树之后，通过“可移动边”的调整，找到时延优化的布线树，调整过程实时利用 SMART 算法更新矩信息。在串扰优化方面，我们利用电路三阶中心矩作为串扰评估模型，并首先对该模型的正确性进行了验证。通过将布线树拆分成二端线网，在通过调整局部二端线网的方式，我们找到串扰优化的布线结构。我们对时延驱动与串扰优化的方法都进行了测试和验证，实验证明我们的方法有效地改善了关键线网的时延，并且优化了串扰影响。

6．2　研究展望

随着深亚微米工艺的进步，电路连线中所产生的问题如布线时延，线网串扰等问题对芯片性能的影响越来越大，这给传统的自动化布线技术带来了许多新的挑战。如何在大规模布线中同时对多个指标如拥挤，时延，串扰等等问题进行优化是很重要的研究课题。

从电路结构中得到矩分量信息来反映结构的时延特性，波形质量的方法以及如何利用矩分量来指导布线逐渐成为研究的热点。结合已进行的相关工作和取得的阶段性成果，进一步的研究工作将包括以下方面。（1）采用更精确的串扰模型，提高总体布线阶段串扰优化的性能。（2）改进 SMART 算法，结合**[9]**提出的 reroot 技术，通过有效地移动挂载子树的方法，找到性能更优的布线树。（3）在有效的线网参数提取方面可做更深入的研究，包括如何有效地提取耦合电容和互感值。（4）在布线阶段，更多考虑信号完整性的其它问题。

参考文献

[1] J. Cong, J. Fang, and Y. Zhang, “Multilevel approach to full-chip gridless routing”, in Proc. of Int. Conf. Computer-Aided Design, pp. 396-403, Nov.2001.

[2] Sakurai T. “Closed-form expression for interconnection delay,coupling,and crosstalk in VLSI”. *IEEE Trans.Electron Devices*, pp. 118-124, 1993

[3] Kawaguchi H, Sakurai T. “Delay and noise formulas for capacitively coupled distributed RC lines”. *Proc. Asia-pacific Design Automation Conf.* pp. 35-43, 1998

[4] J. Cong, Cheng-Kok Koh, Patrick H.Madden, “Interconnect Layout Optimization Under Higher Order RLC Model for MCM Designs”, *IEEE Trans. on Computer-Aided Design of Intergrated Circuit and Systems*, vol. 20, No. 12, pp. 1455-1463, Dec. 2001

[5] G. Shi, “A Symbolic Moment Calculator for RLC Tree Circuits with Application,”submitted to ISCAS, 2009

[6] J. Cong, L. He, C.-K. Koh, and P.H.Madden, “Performance optimization of VLSI interconnect layout,” *Integr. VLSI J.*, vol. 21, no. 1-2, pp. 1-94, Nov. 1996

[7] J. M. Rabaey, A. Chandrakasan, and B. Nikolic , *Digital Integrated Circuits – A Design Perspective (2nd Ed)*. Englewood Cliffs, NJ: Pearson Education, Inc., 2003.

[8] M. Celik, L. Pileggi and A. Odabasioglu. *IC Interconnect Analysis*. Kluwer Academy Publishers, 2002.

[9] S.-P. Lin, Y.-W. Chang, “A Novel Framework for Multilevel Routing Considering Routability and Performance”, *Proc. ICCAD*, pp. 44-50, Nov. 2002

[10]`K.mehlhorn, s.Naher. “LEDA A Platform for Combinatorial and Geometric Computing”, Cambridge University Press, 1999

[11] “Qt Cross-Platform Whitepaper” http://trolltech.com/products

[12] http://www.public.iastate.edu/nataraj/ISPD04_Bench.html

[13] 洪先龙，严晓浪，乔长阁.”超大规模集成电路布图理论与算法”，科学出版社，1998

[14] F.K.Hwang, D.S.Richards, and P.Winter. The Steiner tree problem. *Annals of Applied Mathematics*, pp.104 – 114, 1976

[15] Chris Chu, “FLUTE: Fast Lookup Table Based Wirelength Estimation Technique”,*Proc. International Conference on Computer Aided Design*, pp. 696-701, 2004

[16] F.K.Hwang, An O(nlogn) Algorithm for Rectilinear Steiner Trees, *Journal of the Association for Computing Machinery*, pp. 177-182, 1976.

[17] GeoSteiner – software for computing Steiner trees. http://www.diku.dk/geosteiner/

[18] J. Griffith, G. Robins, J.S.Salowe, and T.Zhang. Closing the gap: Near-optimal Steiner trees in

polynominal time. *IEEE Trans. Computer-Aided Design*, 13(11):1351-1365, November 1994.

[19] I. I. Mandoiu, V. V. Vazirani, and J. L. Ganley. “A new heuristic for rectilinear Steiner trees,”. In *Proc.IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 157-162, 1999

[20] C. L. Ratzlaff and L. T. Pillage, “RICE: rapid interconnect circuit evaluation using AWE”,[J] *Computer-Aided Design of Intergrated Circuits and Systems, IEEE Transactions on* vol.13, pp.763-776, 1994

[21] Q. Yu and E. S. Kuh “Exact Moment Matching Model of Transmission Lines and Application to Interconnect Delay Estimation,”. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*,3(2), pp. 311-322, Jun. 1995.

[22] Q. Yu and E. S. Kuh “Moment Models of General Transmission Line with Application to MCM Interconnect Analysis”, *Proc. IEEE Multi-Chip Module Conf.*, pp.594-598, 1995.

[23] C. J. Alpert, A. Devgan, and C. Kashyap, “A two moment RC delay metric for performance optimization,” in *Proc. International Symposium on Physical Design(ISPD)*, San Diego, CA, pp. 73-78, 2000.

[24] E.Bozorgzadeh, R.Kantner, and M.Sarrafzadeh, “Creating and exploiting flexibilty in rectilinear Steiner trees”, *IEEE Trans. on Computer-Aided Design*,vol.22, no.5, pp. 255-265, May 2003.

线网对应的 gdf 文件（附录 1）

下面是图 3 线网对应的 gdf 文件：

```
(gdif
 (gdifVersion 1 0 1)
 (comment Generated by tw2gdif)
 (cell:top
                (text:number_of_layers "3")
                (text:wire_widths   "0.6 0.6 0.6")
                (text:via_widths    "0.6 0.6 0.6")
                (text:wire_spacings "1.2 1.2 1.2")
                (text:via_spacings  "1.2 1.2 1.2")
                (text:vertical_wire_costs    "2 1 2 1 2 1")
                (text:horizontal_wire_costs  "1 2 1 2 1 2")
                (text:via_costs "10 10 10 10 10 10")

                (path:BBOX
                                (new)(layer LEV)(width 0)(pt -5 52)(pt -5 4956)
                                (new)(layer LEV)(width 0)(pt -5 4956)(pt 4898 4956)
                                (new)(layer LEV)(width 0)(pt 4898 4956)(pt 4898 52)
                                (new)(layer LEV)(width 0)(pt 4898 52)(pt -5 52)
                )

  (port:pad_26_A_0_A_0 (pt 8 707))
  (port:INS241_a (pt 8 2121))
  (port:INS225_a (pt 500 4242))
  (port:INS209_a (pt 500 4949))
  (port:INS193_a (pt 1000 707))
  (port:INS177_a (pt 1500 2121))
  (port:INS161_a (pt 1500 3535))
  (port:INS145_a (pt 1500 4242))
  (port:INS129_a (pt 2000 707))
  (port:INS113_a (pt 2500 2121))
  (port:INS97_a (pt 2500 3535))
  (port:INS81_a (pt 3000 4242))
  (port:INS65_a (pt 3500 8))
  (port:INS49_a (pt 3500 3535))
  (net:A_0
   (portRef pad_26_A_0_A_0)
   (portRef INS241_a)
   (portRef INS225_a)
   (portRef INS209_a)
   (portRef INS193_a)
   (portRef INS177_a)
   (portRef INS161_a)
   (portRef INS145_a)
   (portRef INS129_a)
   (portRef INS113_a)
   (portRef INS97_a)
   (portRef INS81_a)
   (portRef INS65_a)
   (portRef INS49_a)
  )
 )
)
```

程序实现的基本数据结构（附录 2）

1 Tile

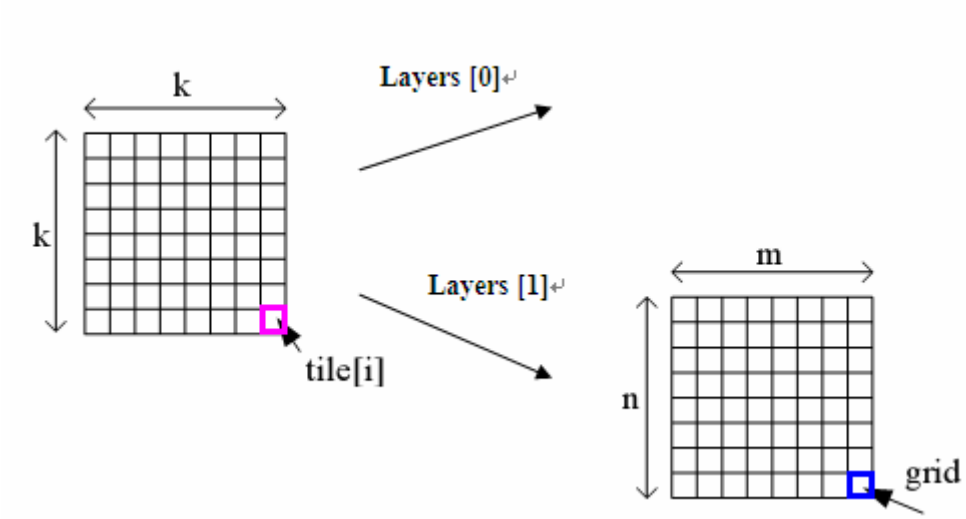


图 40 Tile 和 grid 实例

Fig. 40 Tile and grid

我们将布线区域划分成 K × K 的方格，其中每个小方格（如图 中左边所示）称为一个 tile。每个 tile 中又进行多层的布线，我们对每个 tile 又细分为 m × n 的网格，其中每个网格称为一个 grid，如图 40 中右边所示。

```
typedef struct {                Layer 结构的定义；
    HV_Type    type;            该 Layer 的走线方式，水平或者垂直；
    vector<int> grids;          该 Layer 中包含的 grids 向量列表，每个
                                grid 对应一个整数编号；
} Layer;

typedef struct {                DetailTile 结构的定义
    int grids_used;             该 DetailTile 包含的 grid 数目；
    vector<Layer> layers;       该 DetailTile 包含的 Layer，存储在向量表中；
} DetailTile;

typedef struct {                Tile 结构的定义；
    int lx,rx,by,ty;            该 Tile 的左下角和右上角坐标；
    int hdemand,vdemand;        该 Tile 水平或垂直方向上已使用的布线通道
                                数；
    DetailTile*  dt;            该 Tile 所指向的详细的 grid 结构；
```

} Tile;

2 Cellinstance

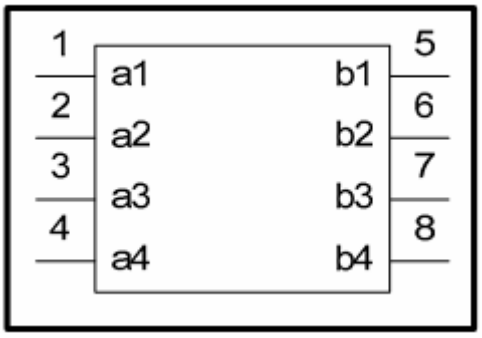


图 41 Cellinstance 实例

Fig. 41 Cellinstance instance

int uid:	每个 Cellinstance 对应的唯一的 ID;
const char* name:	该 Cellinstance 的名字;
double max_delay:	该 Cellinstance 的最大时延;
vector<int> pins_uid:	该 Cellinstance 所有引脚对应的 ID，存储在向量表中

3 CellinstancePin

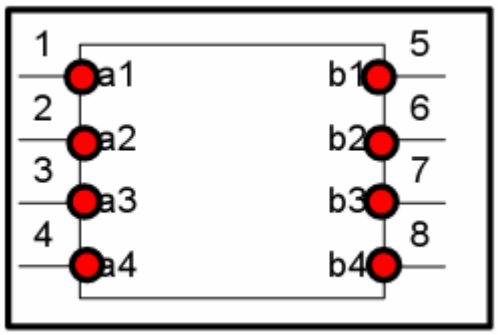


图 42 CellinstancePin 实例

Fig. 42 CellinstancePin instance

int uid:　　该 CellinstanPin 对应的唯一的 ID
char *name:　　该 CellinstanPin 的名字
int ci_uid:　　该 CellinstanPin 所在 Cellinstance 对应的 ID;
int net_uid:　　该 CellinstanPin 所在的 Net 对应的 ID;
lead_node　n:　　该 CellinstanPin 在图的数据结构中所对应的顶点
int ax,ay:　　该 CellinstanPin 的 x，y 坐标;
PinType　type:　　该 CellinstanPin 的类型，为 IN，OUT 或 UNKNOWN。

vector<double>　weightedd_c:
　　该 CellinstanPin 对应的各阶电容树上节点的值
vector<double>　treemoment:
　　该 CellinstanPin 对应的各阶矩，存储在向量表中
double u2:　　该 CellinstanPin 对应的二阶中心矩;
double u3:　　该 CellinstanPin 对应的三阶中心矩;

double delay:　　该 CellinstanPin 到源点的时延;

4　Net

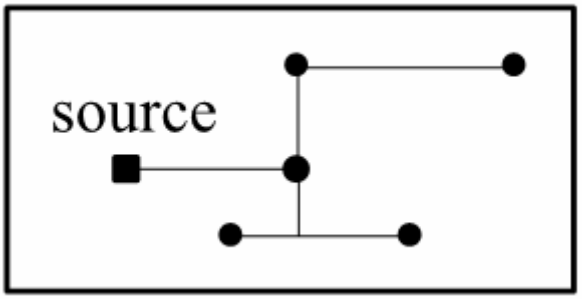


图 43　Net 实例

Fig. 43　Net instance

int uid:　　该 Net 对应的唯一的 ID;
char*　name:　　该 Net 的名字;
int source:　　该 Net 的源点的 ID，每个 Net 仅有一个源点;
vector<int> pins_uid:　　该 Net 所有引脚对应的 ID，存储在向量表中;
int edge_routed:　　该 Net 对应的布线方法;
double max_delay:　　该 Net 所允许的最大时延，即时延约束;
double　delay_lower_bound:　　最短路径布线时漏点到源点的最大时延;
double delay_mst_routing:　　最小生成树布线时漏点到源点的最大时延;
double　delay_smt_routing:　　采用斯坦纳布线时漏点到源点的最大时延;

5 Edge

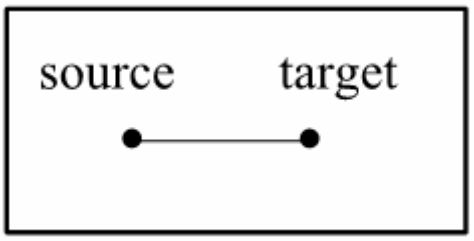


图 44　Edge 实例

Fig. 44　Edge instance

int uid:	该 Edge 对应的唯一的 ID;
CellinstancePin　*source:	该 Edge 的源点;
CellinstancePin　*target:	该 Edge 的漏点;
char*　name:	该 Edge 的名字;
vector<int> tile_routing:	该 Edge 在总体布线时所经历的 tile 序号，存储向量表中;
int dt_grids:	该 Edge 在详细布线时所经历的 grid 的数目;
vector<int*> dt_route_grids:	该 Edge 在详细布线时所经历的 grid 序号的指针;
int distance:	该 Edge 源漏点间的曼哈顿距离;
bool routed:	该 Edge 是否已布线成功;
int level:	该 Edge 在多级布线框架中的级数;

致谢

在我的硕士论文即将结束时，我真心感谢很多给予我学习和生活关怀帮助的人。我首先要感谢的是我的导师施国勇教授。在我攻读硕士期间，从硕士研究课题的选择，到硕士论文的撰写，施老师都给了我悉心的指导。作为施老师带领的EDA 研究小组中的一员，我觉得非常的幸运，两年半组内的学习研究极大地拓宽了我的知识范围。在课题研究过程中，施老师从硬件和软件方面尽最大的可能提供满足，每周的组会及时地了解我们的研究进展，并且悉心地帮助解决课题进展中遇到的问题。施老师渊博的学识和对待科研创新进取的精神都是我学习的榜样。在这里我再一次感谢恩师在这两年半学习上的指导和帮助。

同时要感谢所有陪伴我一同成长的组内同学，他们是于雪红，刘安，徐迪，孟晓旋，祝翔宇。也要感谢黄伟坚学弟在这些日子的陪伴和论文撰写与程序实现等等方面的协助。

还要感谢我的父母，家人以及学院领导对我生活上的关心和学业上的理解和支持。

最后再一次感谢给过我帮助的老师，同学和家人，谢谢你们的支持和帮助。

攻读硕士学位期间已发表或录用的论文

[1] 黄世杰. 考虑串扰效应的性能驱动全局布线器研究. 微电子学（已录用）