# 芯片级布图规划中的缓冲器规划算法*

陈 松** 洪先龙 董社勤 马昱春 蔡懿慈

(清华大学计算机科学与技术系, 北京 100084)

Chung-Kuan Cheng

(University of California, San Diego, USA)

顾 钧

(香港科技大学计算机科学与技术系, 香港)

摘要 文中对纳米技术下, 互连驱动的芯片级布图规划问题中的缓冲器规划问题进行了研究, 提出了基于空白区重分布(redistribution)的缓冲器规划算法; 布局中的空白区是指在布局中不被任何电路模块占用的闲置区域; 该算法充分利用布局中的空白区插入缓冲器. 在基于拓扑的布图规划表示中, 可以把电路模块和空白区相关联, 在相应区域内移动一些电路模块就可以达到重分布空白区的目的. 在进行空白区重分布的过程中, 给定布局的总面积和拓扑结构将维持不变. 通过重分布布局中的空白区, 可以增加满足时延约束的线网数目; 实验证实, 满足时延约束的线网数的增长率平均达到 9%.

关键词 缓冲器规划 空白区 重分布 布图规划 **VLSI** 角模块序列

随着超大规模集成电路(VLSI)的不断发展, 纳米技术和高频技术的应用, 互连的设计与优化成为决定系统性能、成本和可靠性的关键问题. 为了确保电路设计的时序特性, 在设计的流程中, 互连的设计应该尽早加以考虑. 在过去 10 年里, 人们已经研究出了多种互连优化技术, 例如拓扑的构造, 缓冲器插入, 器件尺寸的优化, 线宽和线间距的优化, 以及上述各种优化技术的组合应用. 在文献[1]中

2004-03-13 收稿, 2004-06-07 收修改稿

*国家自然科学基金(批准号: 90307005)、国家自然科学基金委员会与香港研究资助局联合资助项目(批准号: 60218004)、国家自然科学基金国际合作项目(批准号: 60121120706)、美国国家自然科学基金项目(CCR-0096383)与“八六三”国家高技术研究发展计划(2002AA1Z1460)资助项目

** E-mail: chensoo@mails.tsinghua.edu.cn

对这些互连优化技术进行了全面的概述.

缓冲器插入对减少互连时延来说, 是一种非常有效的方法. 在Elmore时延模型下, 长连线的时延与线长的平方成比例增长, 而利用缓冲器插入技术, 可以控制连线的时延与线长呈线性增长关系. 为满足时延约束, 需要插入的缓冲器数目随工艺尺寸的减小而持续增长. 例如, 在 70 nm工艺下[2], 总线长估计为 10 km的芯片上平均需要插入 800, 000 个缓冲器以满足性能优化. 由于缓冲器需要占用硅资源, 所以在设计过程中, 缓冲器应该尽早规划. 近年来, 缓冲器规划问题受到广泛的关注, 人们已经提出了很多缓冲器规划的方法, 但是由于缓冲器规划问题本身的复杂性, 无法将其有效地集成到布图规划过程中, 因此基本上将其作为布图规划的后处理过程.

Cong等[3,4]引入了可行区域(feasible region)的概念, 并利用可行区域来产生缓冲器模块; 缓冲器的可行区域是指在满足线网的目标延时的条件下, 缓冲器可以放置的可能最大区域; 他们利用空白区插入缓冲器. Sarkar等[5,6]在可行区域的基础上引入了独立可行区域的概念, 同时试图优化布线拥挤度; 缓冲器模块都被放置在电路模块间的空白区域内. Tang和Wang[7,8]在假设每个线网只有一个缓冲器的情况下, 提出了求解最优解的算法, 同样模块间的空白区域用来插入缓冲器. Dragan等[9,10]用多商品流的方法, 将缓冲器分配到预先已存在的缓冲器模块. Alpert等[11,12]利用tile图和动态规划解决缓冲器模块规划问题; 他们假设在宏模块内部也允许插入缓冲器, 因此这种方法会使插入缓冲器的点分布在布局的各个位置. Sham等[13,14]发表了可布性驱动的布图规划系统, 该系统可以在考虑拥挤度约束的条件下估计缓冲器的用量和缓冲器资源. Rafiq等[15]提出了集成缓冲器/通道插入的布图规划系统, 用于基于总线的微处理器设计.

布局中的空白区是指在布局中不被任何电路模块占用的闲置区域. 在有较多 IP(知识产权)模块存在时, 把缓冲器插入模块间的空白区是达到集成系统性能目标的惟一选择. 作为布图规划的后处理过程, 缓冲器规划一定是在布图规划较优解(即较为紧凑的布图)的基础上进行; 因而其空白区一定有了确定的位置、大小和数量. 目前所有相关研究基本是在这个限定的前提下进行缓冲器规划. 如何突破这一限定的前提, 使布线阶段确定具体的缓冲器数目和位置时能有更好的布图规划和缓冲器规划的结果作为开始, 不外两种解决方案, 一种是在布图规划阶段有意增加空白区, 另一种就是改变现有空白区的分布. 前一种方法很难实现, 因为无法在布图规划过程中有效的集成缓冲器的规划, 从而无法有效的预留空白区: 留多少空白区及在何处增加空白区, 没有有效的标准和模型来计算判别. 后一种方法正是我们采用的方法, 对空白区进行重分布. 虽然在布图规划结果确定的情况下, 布局中的空白区的多少已经被限定, 但是我们提出的算法对可以处理的空白区的数量没有限制. 此前, 没有任何已有方法对布局中的空白区分布进

行改变, 以充分利用空白区改善芯片的时延特性.

在本文的工作中, 我们设计并实现了基于空白区重分布的缓冲器规划算法, 将空白区充分地利用于缓冲器的插入. 基于某些拓扑的布图结构表示, 可以把电路模块和空白区相关联; 例如, 在角模块序列[16,17]的布图结构表示方法下, 我们可以计算给定布局中的全部空白区, 并实现对空白区的重分布. 空白区和模块相关联以后, 在关联的区域内移动一些电路模块就可以达到重分布空白区的目的; 同时在重分布空白区的过程中, 给定布局的总面积和拓扑结构将保持不变. 在满足时延约束的条件下, 我们计算每一个缓冲器插入的独立可行区域**(IFR)**[5,6]; 所有的缓冲器可以在他们的IFR内任意移动, 但不会破坏时延约束; 每一个缓冲器都可以插入到该缓冲器的IFR和空白区相交的区域. 因此, 重分布布局中的空白区能够增加插入的缓冲器的数目; 换句话说, 就是可以增加满足时延约束的线网数目, 这一点已经被实验结果所证实. 满足时延约束的线网数的增长率平均达到9%.

本文各部分按如下组织: 第一节给出问题的定义, 介绍了独立可行区域的概念, 同时简单回顾了角模块序列的布图结构表示方法. 空白区的重分布问题在第二节讨论. 第三节给出了缓冲器规划和优化算法. 第四节和第五节分别是实验结果和结论.

## 1　预备知识

### 1.1　问题定义

本篇论文主要讨论缓冲器的规划问题: 给定初始的布图规划结果以及每一个线网的时延约束, 我们试图确定每条线网满足时延约束需要的缓冲器的数目和位置; 缓冲器插入电路模块间的空白区. 缓冲器规划中利用的空白区与模块或其间的通道紧邻, 但一定是不重叠的. 模块间的空白区可以分为两类(2.1 节), 一类是可以重分布的空白区; 另一类是布图规划结果中必然存在的“死区”这一类空白区是不可重分布的, 但可以用来插入缓冲器, 算法有效地利用了这一类空白区. 此外, 如果布图规划阶段为了某种需要预留了空白区; 对于这类预留的空白区和通道区域, 如果容许插入缓冲器, 所提算法可以很容易的将其纳入缓冲器的规划之中, 否则, 只需将其看成一种“假”模块, 从而阻止缓冲器的插入.

### 1.2　独立可行区域

Sarkar和Koh[5,6]介绍了缓冲器插入的独立可行区域(IFR)的概念, 并且给出了解析公式计算IFR. 缓冲器$b$的独立可行区域是指在满足$b$所属线网的时延约束条件下, 同时假设该缓冲器所属线网的其他缓冲器都被放置在它们各自的独立可行区域内, 可以插入缓冲器$b$的最大区域. 图 1 给出了某线网的一个缓冲器的IFR

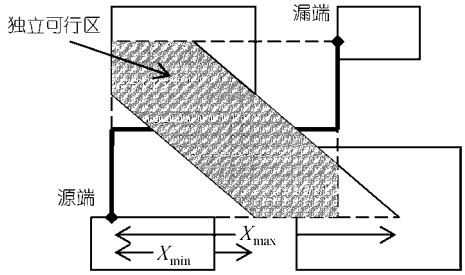



图 1 独立可行区域都是凸的多边形

示意图.

长度为$l$的互连线满足时延约束需要的最小的缓冲器数[3,4]可通过下式计算[3,4]:

$$k_{\min} = \left\lceil \frac{-K_5 - \sqrt{K_5^2 - 4K_4K_6}}{2K_4} \right\rceil, \quad (1)$$

其中

$$K_4 = R_bC_b + T_b$$

$$K_5 = (rC_b + cR_b)l + T_b + R_dC_b + R_bC_l - T_{req} - \frac{r}{2c}(C_b - C_l)2 - \frac{c}{2r}(R_b - R_d)^2,$$

$$K_6 = \frac{1}{2}rcl^2 + (rC_l + cR_d)l + R_dC_l - T_{req}$$

文中用到的和互连线和缓冲器有关的物理参数符号意义如下: $r$: 单位长度线电阻, $c$: 单位长度线电容, $T_b$: 缓冲器固有时延, $C_b$: 缓冲器输入电容, $R_b$: 缓冲器输出电阻; $C_l$: 漏点负载电容, $R_d$: 驱动电阻, $l$: 两端线网长度.

特定线网中每一个缓冲器的 IFR 的计算方法可以参考文献[5,6].

### 1.3 角模块序列表示

角模块序列(CBL)是由Hong等[16,17]提出的一种拓扑的布图结构表式方法. 角模块序列通过三元组($S$, $L$, $T$)来表示布图结构; 这种表示根据三元组($S$, $L$, $T$)将芯片分割成矩形区域, 称为room; 并且为每个room分配一个且仅一个电路模块; 三元组中, $S$代表模块的分配方案, $L$和$T$确定了分割芯片所用的正交线段. Dong等[18]对角模块序列表示进行了扩展, 在角模块序列表示中加入空room, 并且为每一个空room分配一个伪模块. 如图 2(a)所示, 伪模块 0 被分配到了空room. 在本文以后的部分, 如果不做特别说明, 角模块序列均指扩展后的角模块序列.

序列 $S$ 按照从左下角到右上角的顺序记录模块, 序列中位置靠左的模块一定在位置靠右的模块的左边或者下边.

序列 $L$ 记录了模块的方向. 在布图结构中, 角模块左边界和下边界的线段构成 $T$-连接, 该 $T$-连接定义了角模块的方向; 该 $T$-连接可以有两种不同的方向: 一种是 $T$ 逆时针旋转 90°(‘ ┣ ’)表示垂直方向, 另一种是 $T$ 逆时针旋转 180°(‘ ┻ ’)表示水平方向; 图 2(a)给出了表示垂直方向的 $T$-连接.

$T$ 序列记录了各模块覆盖的 $T$-连接的数目. 以‘ 0 ’结尾的二进制序列 $T_i$ 用来记录角模块 $b_i$ 所覆盖的 $T$-连接的数目. 例如图 2(a)中, 模块 $d$ 覆盖了一个 $T$-连

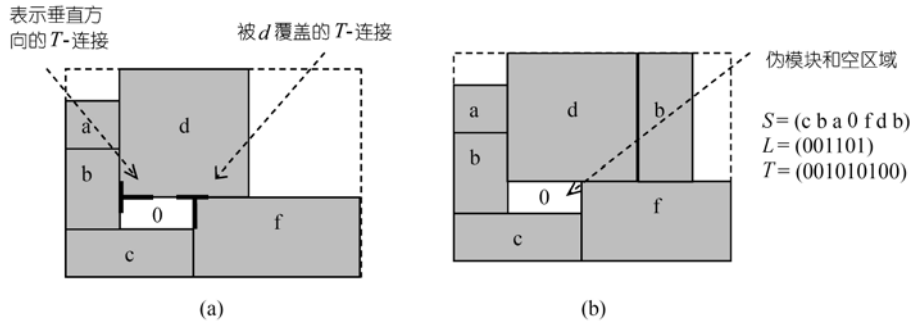



图 2　角模块序列表示

(a) 模块 $d$ 的方向和模块 $d$ 所覆盖的 $T$-连接; (b) 布局及其角模块序列表示

接, 在 $T$ 中有二进制序列‘10’记录模块 $d$ 所覆盖的 $T$-连接的数目.

给定三元组($S$, $L$, $T$), 可以构造相应的布图结构. 图 2(b)所示, 是某个布局和相应的角模块序列表示. 有关 CBL 的细节可以参考文献[16,17].

## 2　空白区的重分布

在这一部分, 我们将论述空白区的重分布, 然后讲述如何计算布局中的空白区.

### 2.1　空白区重分布的实现

为简化起见, 我们所提空白区是指在布局中不被任何电路模块所占用的区域. 在布图规划中, 芯片总是可以被分割成 room, 每一个 room 中至多可以有一个模块. 因为并不是所有的 room 都完全被电路模块占用, 同时可能会有一些空的 room 存在(或者看作分配了伪模块), 所以就存在一些空白区. 根据空白区的产生方式, 可以将其以下两种类型:

定义 **1**　如果空白区是因为空 room 的存在而产生的, 那么该空白区称为隔离空白区(Detached Dead-Space), 简记为 DDS.

隔离空白区不能和任何电路模块相关联. 例如, 图 3(a)所示的伪模块 0 所属的空 room 就是隔离空白区, 该空白区不能和周围的电路模块相关联, 也就是说该空白区和其周围的电路模块不可能属于同一个 room.

定义 **2**　如果空白区是由于某 room 没有完全被电路模块占据而产生的, 那么该空白区称为附属空白区(Attached Dead-Space), 简记为 ADS.

附属空白区可以和相应 room 中的电路模块相关联. 如图 3(a)所示, 空白区 $a_1$, $e_1$ 和 $e_2$ 都是附属空白区. 空白区 $a_1$ 可以和模块 $a$ 相关联, 空白区 $e_1$, $e_2$ 可以和模块 $e$ 相关联.

隔离空白区是不能移动的, 而附属空白区的分布是可以通过移动相应的模

块而改变的. 因为存在附属空白区, 所以一些电路模块可以在特定的一维或二维区域(例如图 3(b)中的模块 $e$)内自由移动, 同时保持布图结构的拓扑和总面积不变. 通过布图结构的拓扑表示, 附属空白区可以分别与电路模块相关联; 在下面我们会谈到如何将附属空白区和电路模块相关联. 空白区的重分布就可以通过移动这些电路模块来实现. 在图 3(a)中, 附属空白区 $e_1$ 在模块 $e$ 的上方, $e_2$ 在模块的右方, 它们都和模块 $e$ 相关联. 图 3(b)给出了图 3(a)中附属空白区的一个重分布. 通过移动模块 $a$, 空白区 $a_1$ 分成了 $a_{11}$ 和 $a_{22}$, 移动模块 $e$, 空白区 $e_2$ 分成了 $e_{21}$ 和 $e_{22}$, 同时 $e_1$ 被移到了模块 $e$ 的下方.

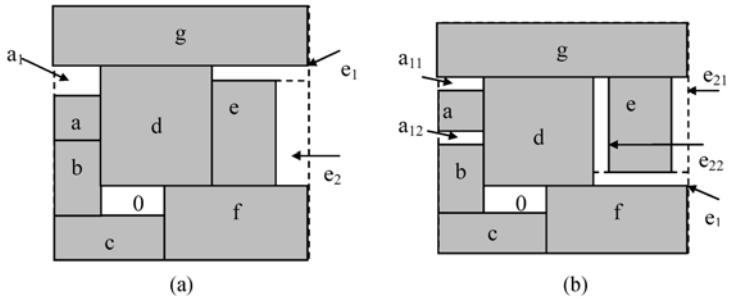



图 3 布局中的空白区

(a) 两种类型的空白, (b) 重分布空白区

因此我们很容易得到下面的定理.

定理 **1** 重新分布布局中的附属空白区可以完成空白区的重分布而保持布局的拓扑结构和总面积不变.

证 当模块在所属的 room 内移动时, 布局中模块间的拓扑关系和布局的总面积保持不变. 芯片由水平和垂直的线段划分为 room, 每一个 room 中至多有一个模块. 这些水平的和垂直的线段确定了 room 之间的拓扑关系. 因此, room 间的拓扑关系也就反映了电路模块间的拓扑关系. 电路模块的移动只限于相应的 room 内, 并不带动其他模块的移动, 因此整个布局的拓扑结构和总面积保持不变.

### 2.2 计算给定布局中的空白区

在这一小节里, 我们讨论在角模块序列(1.3 节)表示下, 如何找到布局中所有的空白区, 怎样把空白区和电路模块(或伪模块)相关联.

对 $n$ 个模块的布局, CBL 通过水平的和垂直的线段把芯片分割成 $m(m \geq n)$个矩形的 room, 这些线段决定了 room 间的拓扑关系. 因此, 每一个空白区必然属于特定的 room. 因为每一个 room 都分配有一个电路模块或伪模块, 因此每一个空白区必然和某一个电路模块或伪模块相关联. 显然, 所有的隔离空白区必然和

伪模块相关联, 而所有的附属空白区必然和电路模块相关联.

在角模块序列表示到布局的转换过程中, 当插入一个角模块时, 该角模块必然覆盖一些其他的已插入的模块. 在某模块的插入过程中, 检查该模块覆盖的所有模块, 通过比较该模块的坐标和其所覆盖模块的坐标可以确定该模块和其所覆盖的模块间是否存在空白区; 同时将这些空白区与该模块所覆盖的相应模块关联. 例如, 在图 3(a)中, 模块 $a$, $d$, $e$ 都被模块 $g$ 所覆盖. 分别比较模块 $g$ 的坐标和 $a$, $d$, $e$ 的坐标可以找到他们之间的空白区. 找到的空白区 $a_1$ 与模块 $a$ 相关联, 空白区 $e_1$ 与模块 $e$ 相关联. 隔离空白区会自动与伪模块相关联, 并且该伪模块的尺寸就是隔离空白区的大小; 伪模块的大小可以在布局过程中自动确定. 通过检查伪模块尺寸, 可以找到隔离空白区. 如图 3(a)中, 与伪模块 0 关联的隔离空白区.

## 3　缓冲器规划和优化算法

本节详述了基于空白区重分布的缓冲器规划算法. 首先讨论了如何计算缓冲器的候选位置; 其次, 通过二分图表示所有可能的从缓冲器到 tile 的分配, 并求取二分图中的最大势匹配以最大限度的插入缓冲器; 最后, 改变布局中空白区的分布, 以优化前面两步得到的结果.

给定布局, 假设缓冲器只可以插入空白区, 也就是说在电路模块内部不允许插入缓冲器. 如文献[16]所述, 布局的 CBL 表示可以通过递归的删除角模块获得; 获得布局的 CBL 表示后就可以按照第二节中所描述的方法来计算布局中的空白区, 同时把空白区与电路模块或伪模块相关联. 在计算结果中, 空白区均以矩形区域的形式给出.

### 3.1　缓冲器候选位置的计算

布局中的空白区均被划分为 tile, 每个 tile 提供一个可以插入缓冲器的位置. 为达到满足目标延时的目的, 缓冲器应该插入到它们的独立可行区域内的 tile 中. 在缓冲器规划过程中, 缓冲器候选 tile 的计算是较为困难而且很耗时的操作. 最简单的做法就是检查所有的 tile, 看其是否在某缓冲器的独立可行区域内; 但是这种方法的时间复杂度依赖于 tile 的数量. 在这里, 我们设计了一个快速的计算缓冲器候选位置的方法.

线网中缓冲器的独立可行区域为规则的凸多边形(图 4(a)), 该多边形由两条斜率为−1 或+1 的平行线和线网的源端和漏端所确定的矩形区域相交而成. 直接计算凸多边形和空白区之间的交较为复杂, 本文把这个问题分解为两步: 第一步计算空白区和缓冲器所在线网的源端和漏端所确定的矩形区域的交; 第二步计算缓冲器可行区域中两条斜率为−1 或+1 的平行边之间的区域与第一步中计算所

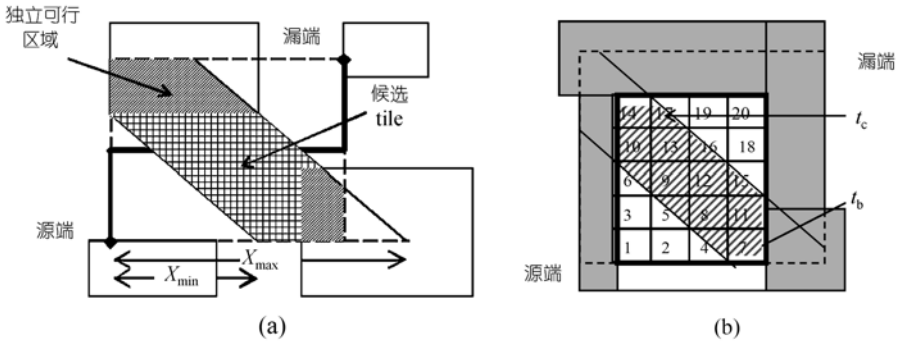



图 4 缓冲器候选位置及其计算

(a) 独立可行区与候选 tile, (b) 缓冲器候选位置计算

得矩形区域的交以确定缓冲器的候选位置.

步骤 **1** 计算空白区和缓冲器所在线网的源端和漏端所确定的矩形区域的交.

在计算几何中, 计算两个矩形的交是一个很基本的问题. 我们通过两次计算线段的交求出矩形的交. 假设矩形空白区的左下角和右上角坐标分别为 $(x_{ds}^{lb}, y_{ds}^{lb})$, $(x_{ds}^{rt}, y_{ds}^{rt})$; 线网所确定的矩形区域左下角和右上角坐标为 $(x_s^{lb}, y_s^{lb})$, $(x_t^{rt}, y_t^{rt})$, 如果下列不等式成立, 则表示这两个矩形是相交的.

$$\begin{cases} x^{rt} - x^{lb} > 0 \\ y^{rt} - y^{lb} > 0, \end{cases}$$

其中, $x^{lb} = \max(x_{ds}^{lb}, x_s^{lb})$; $y^{lb} = \max(y_{ds}^{lb}, y_s^{lb})$; $x^{rt} = \min(x_{ds}^{rt}, x_t^{rt})$; $y^{rt} = \min(y_{ds}^{rt}, y_t^{rt})$.

交区域的左下角顶点和右上角顶点分别为$(x^{lb}, x^{lb})$, $(y^{rt}, y^{rt})$. 下面把本步的计算结果简记为交空白区.

步骤 **2** 计算交空白区与缓冲器可行区域的交. 如果交空白区和可行区域不相交, 那么该缓冲器在当前空白区内没有候选位置; 相交的情况下可以通过如下方法计算相交区域.

在计算中, 如果某 tile 的左下角位于缓冲器的独立可行区域内, 那么我们认为该 tile 是相应缓冲器的候选位置.

为了计算缓冲器的候选位置, 我们沿独立可行区域的平行边方向对交空白区内的 tile 进行编号, 如图 4(b)给出了斜率为–1 的情况. 在这里, 就斜率为–1 的情况进行说明, 另一情况可以类似处理. 为了便于说明, 对斜率为–1 的两条边, 左边的边称为左平行边, 右边的边称为右平行边.

假设在交空白区的下边界或右边界上, 靠近左平行边的第一个候选 tile 为 $t_b$,

在交空白区的左边界或上边界上, 离右平行边最近的第一个候选 tile 为 $t_e$, 那么缓冲器的候选位置就是 $t_b$ 和 $t_e$ 之间的所有 tile. 例如图 4(b)中缓冲器的候选位置为 7 和 17 之间的所有 tile.

### 3.2　缓冲器规划算法

缓冲器规划算法的目标是确定缓冲器的数目和位置, 并且尽可能多的插入缓冲器, 使得满足时延约束的线网数目达到最大.

首先, 计算每一个缓冲器的候选 tile; 然后构造二分图来表示从缓冲器到 tile 的所有可能的分配; 最后通过寻找二分图中的最大势匹配确定缓冲器到 tile 的分配方案. 算法 1 给出了缓冲器规划算法的步骤.

**算法 1**　缓冲器规划

1. 对所有的空白区建立 tile 数据结构, 即对空白区进行细划分;
2. 计算每一个缓冲器的独立可行区域;
3. 计算每一个缓冲器的候选 tile 集合;
4. 构造二分图 $G(V, E)$, $V = V_1 \cup V_2$ 其中 $V_1$ 中节点表示缓冲器, $V_2$ 中节点表示 tile, $E = \{(v_1, v_2), v_1 \in V_1, v_2 \in V_2, v_1$ 可以插入 $v_2\}$.
5. 从 $G$ 构造 $s$-$t$ 图;
6. 计算从 $s$ 到 $t$ 的最大流, 确定每一个缓冲器的位置.

第 1 步, 每一个空白区都被划分成小的 tile, 每一个 tile 都是和缓冲器尺寸相当的矩形, 缓冲器可以放置在这些 tile 中. 在第 3 步, 对每一个缓冲器 $b$, 通过 3.1 所述的方法计算该缓冲器所有可能的位置; 因此, 所有可能的缓冲器分配也就计算出来了, 基于这些可能的分配, 可以构造二分图 $G$; $G$ 的每一条边都表示某缓冲器到某 tile 的一种可能分配. $G$ 可以按如下方法构造:

$G(V, E)$;

$V = V_1 \cup V_2$, 其中 $V_1$ 中节点表示缓冲器, $V_2$ 中节点表示 tile;

$E = \{(v_1, v_2), v_1 \in V_1, v_2 \in V_2,, v_1$ 可以插入 $v_2\}$.

在第 5 步中, 为了尽可能多的插入缓冲器, 在二分图 $G$ 的基础上构造 $s$-$t$ 图以计算最大势匹配. 我们把所有的边定向为从 $V_1$ 到 $V_2$; 加入源点 $s$ 及从该源点 $s$ 到 $V_1$ 中所有顶点的有向边, 同时加入吸纳点 $t$ 及从 $V_2$ 中所有顶点到 $t$ 的有向边; 图中的每条边容量均设定为 1. 图 5 给出了示例. 最大势匹配可以通过计算从 $s$ 到 $t$ 的最大流来获得.

### 3.3　优化算法

优化算法通过对布局中的空白区进行重新分布, 改善缓冲器规划的结果, 最

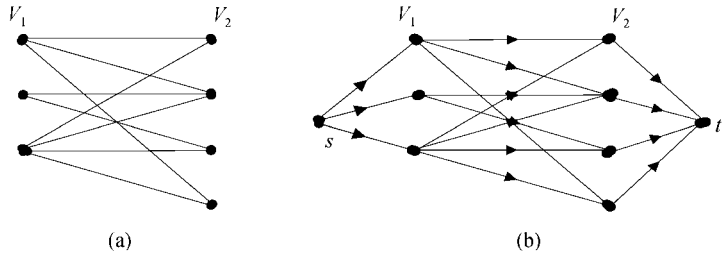



图 5 $s$-$t$ 图的构造
(a) 二分图 $G$; (b) $s$-$t$ 图

大化满足时延约束的线网数. 算法 2 给出了优化算法的步骤.

布局中新的空白区分布可以通过如下两种方法生成. 一种方法是随机选取两个与模块相关联的空白区, 改变选定的空白区在相应模块周围的分布. 在这里采取移动空白区至关联电路模块的另一侧的方式. 如图 3(b)中的 $e_1$ 从模块 $e$ 的上方移到了下方. 另一种方法是选取与某模块相关联的空白区, 改变选定空白区在其关联模块周围的分布. 在这里采用将空白区划分为两部分的方式, 并使得电路模块位于这两部分空白区的中间. 如图 3(b)中空白区 $a_1$ 被分成了 $a_{11}$ 和 $a_{22}$. 新的空白区分布产生后, 需要更新在移动的电路模块上有引脚的线网的信息, 其中包括线网长度、缓冲器数目以及每一个缓冲器的独立可行区域. 在第 5 步中, 如果新的空白区分布不被接受, 我们将恢复空白区分布、及相关线网信息至前一状态.

最后, 重复 3 到 5 步几百次不等.

算法 **2** 优化算法

1. 计算布局中的所有空白区, 并将它们和电路模块相关联;
2. 运行缓冲器规划算法(算法 1)计算满足目标时延的线网数目, 记为 $N_{\text{old}}$;
3. 生成新的空白区分布, 并更新相关信息;
4. 运行缓冲器规划算法计算满足时延约束的线网数, 记为 $N_{\text{new}}$.
5. 如果 $N_{\text{old}} < N_{\text{new}}$, 新的空白区分布将被接受, $N_{\text{old}} = N_{\text{new}}$; 否则恢复至上一次的空白区分布;
6. 在给定的次数内重复步骤 3 到 5.

在空白区重分布的过程中, 布局的拓扑结构和总面积保持不变. 每产生新的空白区分布, 算法 1 用来计算满足目标时延的线网数.

## 4 实验结果

缓冲器规划和优化算法已经在 SUN Ultra-SPARC III (750M)的机器上用 C 语

言实现. 在本节中, 我们给出一些实验结果的细节. 互连线和缓冲器参数已经在第一节中介绍. 文中用到的这些参数值(见表 1)都是基于 0.18 μs 工艺[19].

表 1　本文用到的参数值

| | | |
|---|---|---|
| $R$ | 单位长度线电阻(Ω/μm) | 0.075 |
| $C$ | 单位长度线电容(fF/μm) | 0.118 |
| $T_b$ | 缓冲器固有时延(ps) | 36.4 |
| $C_b$ | 缓冲器输入电容(fF) | 23.4 |
| $R_b$ | 缓冲器输出电阻(Ω) | 180 |
| $R_d$ | 驱动输出电阻(Ω) | 180 |
| $C_l$ | 负载电容(fF) | 23.4 |

本文主要关注两端(单源端, 单漏端)线网的缓冲器规划问题; 所有的多端线网都被拆分为从源到各个漏端的两端线网. 由于在测试基准文件中没有给出信号的方向, 所以我们选取线网中的某个引脚作为源端, 其余的引脚均为漏端, 然后再对多端线网进行拆分. 所有的电源线和地线将不予考虑. 实验中用到的所有 MCNC 的测试基准电路[20]的初始布局是从文献[16]获得的.

由于在MCNC测试基准电路文件中不包含任何时延信息, 两端线网的目标时延采用类似文献[3]中的方法进行赋值. 所有长度小于关键长度$l_{\min}$[21]的两端线网都不予考虑, 因为在这些线网中插入缓冲器无法达到减少时延的目的. 我们计算每一个线网在最优的缓冲器插入[21]下的最优时延$T_{\text{opt}}$, 然后将该线网的目标延时随机赋值为$T_{\text{opt}}$的 1.05 到 1.20 倍[3,4]. 因为我们生成了我们自己的布局和时延约束, 所以直接利用实验结果比较所提出的方法和文献[3~6]中的方法是不合理的, 我们仅在自己实验结果的基础上说明所述方法的有效性, 但同时给出了文献[3]中的部分结果以供参考.

在表 3 中, 我们提供了缓冲器规划及优化算法在 5 个 MCNC 测试基准电路上的实验结果. 这些测试基准电路的详细信息见表 2.

表 3 中的数据包括满足时延约束的线网数, 插入的缓冲器数目, 及优化算法的改善结果和 CPU 时间. “ Buffer Planning ”列给出了在初始的空白区分布下, 运行缓冲器规划(算法 1)算法的实验结果, “ Optimization ”列是优化(算法 2)算法的结果. 子列“ met ”, “ buffers ”和“ time ”分别表示满足时延约束的线网数, 插入的缓冲器数和消耗的 CPU 时间. 优化算法对满足时延约束的线网数的改善及相应的比例分别在“ $N_{\text{imp}}$ ”和“ $R_{\text{imp}}$ ”中给出.

表 3 所给出的实验结果表明, 优化算法可以增加满足时延约束的线网数, 同时保持布局的总面积不变. 例如测试电路 Xerox, 在初始的空白区分布下满足时延约束的线网数是 275, 优化之后满足时延约束的线网数增加到 315; 满足时延约束的线网数增加了 14.5%. 对给出的 5 个电路, 满足时延约束的线网数的增长

比例平均达到 9%. 实验结果表明我们的算法是非常有效的. 由于进行空白区重分布而进行迭代的原因, 算法的运行时间要长于文献[3]和[5].

表 2 MCNC 测试基准电路信息

| 电路 | 模块数 | 线网数 | 两端线网数 |
|---|---|---|---|
| Apte | 9 | 97 | 172 |
| Xerox | 10 | 203 | 455 |
| Hp | 11 | 83 | 226 |
| Ami33 | 33 | 123 | 363 |
| Ami49 | 49 | 408 | 545 |

表 3 缓冲器规划算法(算法 1)和优化算法(算法 2)的实验结果

| 电路 | Buffer Planning | | | Optimization | | | $N_{imp}$ | $R_{imp}$ | 文献[3] | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | met | buffers | time (s) | met | buffers | time (s) | | | met | buffers | time (s) |
| Apte | 89 | 83 | 0.16 | 100 | 104 | 28.6 | 11 | 12.4% | 102 | 185 | 0.23 |
| Xerox | 275 | 152 | 0.1 | 315 | 182 | 8.7 | 40 | 14.5% | 260 | 399 | 0.53 |
| Hp | 129 | 179 | 0.25 | 139 | 182 | 25.1 | 10 | 7.8% | 131 | 280 | 0.48 |
| Ami33 | 235 | 162 | 0.08 | 249 | 178 | 7.1 | 14 | 5.9% | 305 | 667 | 1.63 |
| Ami49 | 437 | 236 | 0.51 | 457 | 253 | 49.1 | 20 | 4.6% | 412 | 946 | 3.25 |

## 5 结论

我们提出了基于空白区重分布的缓冲器规划算法, 以充分利用空白区插入缓冲器. 通过重分布布局中的附属空白区可以达到重分布空白区的目的, 同时保持布局的拓扑结构和总面积不变化. 实验结果说明我们的算法是有效的.

由于我们的缓冲器规划算法(算法 1)是嵌入在优化过程中的, 所以该算法很容易加以扩展以处理额外的约束, 例如拥挤度约束和噪音约束. 为了获取近似最优解, 应用先进的搜索策略是必要的, 例如模拟退火等. 由于费时的原因, 在文中采用了一种贪心算法. 虽然实验结果显示贪心算法是有效的, 但提出更快的缓冲器规划算法以应用更好的搜索策略仍然是必需的. 此外在文中没有考虑非单调的曼哈顿路径情况, 这些我们将在未来的工作中作进一步研究.

致谢 在本文的工作过程中, 清华大学的王一博硕士和 UCLA 的孔天明博士提供了很多帮助, 在此一并致谢.

## 参 考 文 献

1 Cong J, He L, Koh C K, et al. Performance optimization of VLSI interconnect layout. Integration, the VLSI Journal, vol. 21, Nov. 1996, 1~94

2 Cong J. Challenges and opportunities for design innovations in nanometer technologies. Frontiers in Semiconductor Research: A collection of SRC Working Papers, Semiconductor Research Corporation, http://www.src.org/prg_mgmt/frontier.dgw, 1997

3 Cong J, Kong T, Pan D Z. Buffer block planning for interconnect-driven floorplanning. in: Proc Int Conf on Computer Aided Design, Nov. 1999, 358~363

4 Cong J, Kong T, Pan D Z. Buffer block planning for interconnect planning and prediction. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Volume: 9 Issue: 6 , Dec. 2001, 929~937

5 Sarkar P, Koh C K. Routability-driven repeater block planning for interconnect-centric floorplanning. Intl Symp Physical Design, 2000, 186~191

6 Sarkar P, Koh C K. Routability-driven repeater block planning for interconnect-centric floorplanning. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Volume: 20 Issue: 5, May 2001, 660~671

7 Tang X, Wong D F. Planning buffer locations by network flows. Intl Symp Physical Design, 2000, 180~185

8 Tang X, Wong D F. Network flow based buffer planning. Integration, the VLSI Journal, Vol 30, issue 2, October 2001, 143~155

9 Dragan F F, Kahng A B, Mandoiu I, et al. Provably good global buffering using an available buffer block plan. IEEE/ACM ICCAD, 2000, 104~109

10 Dragan F F, Kahng A B, Mandoiu I, et al. Provably good global buffering by multiterminal multicommodity flow approximation. ASP-DAC, 2001, 120~125

11 Alpert C J, Hu J, Sapatnekar S S, et al. A practical methodology for early buffer and wire resource allocation. DAC, 2001

12 Alpert C J, Hu J, Sapatnekar S S, et al. A practical methodology for early buffer and wire resource allocation. Computer-aided Design of Integrated Circuits and Systems, Vol 22, No.5, May 2003

13 Sham C W, Young F Y. Routability driven floorplanner with buffer block planning. ISPD, 2002

14 Sham C W, Young F Y. Routability-driven floorplanner with buffer block planning. Computer-aided Design of Integrated Circuits and Systems, IEEE Trans On, Vol 22, No.4, April 2003

15 Ragiq F, Jeske M C, Yang H H, et al. Integrated floorplanning with buffer/channel insertion for bus-based microprocessor designs. ISPD, 2002

16 Hong X L, Huang G, et al. Corner Block List: An effective and efficient topological representation of non-slicing floorplan. IEEE/ACM ICCAD'2000

17 洪先龙, 马昱春, 董社勤, 等. 角模块序列布图表示及基于角模块序列表示的边界约束布图规划算法. 中国科学, E缉, 2002, 32(3): 409~418[摘要]　[PDF]

18 Dong S, Zhou S, Hong X L, et al. An Optimum Placement Search Algorithm Based on Extended Corner Block List. Journal of Computer Science and Technology, 2002, Vol 17, No.6 Nov

19 Semiconductor Industry Association, National Technology Roadmap for Semiconductors, 1997

20 Collaborative Benchmarking Laboratory, North Carolina State University, http://www.cbl.ncsu.edu/CBL Docs/lys92.html: LayoutSynth'92 Benchmark Information

21 Alpert C J, Devgan A. Wire segmenting for improved buffer insertion. in: Proc. Design Automation Conf, June 1997, 588~593