第 19 卷第 1 期　　半 导 体 学 报　　Vol. 19, No. 1
1998 年 1 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Jan., 1998

功耗和时延双重驱动的 VLSI 布局算法*

孔天明　洪先龙　乔长阁

(清华大学计算机科学与技术系　北京　100084)

摘要　针对超大规模的门阵列和标准单元电路, 本文提出一种功耗和时延双重驱动的 VLSI 布局算法. 以往发表的布局算法中, 很少能够同时处理功耗和时延的双重约束. 在以往的时延驱动布局算法中, 仅有一个算法[3]能够处理超大规模的电路; 该算法尚存在以下问题: 1) 其基本思想只能处理组合电路; 2) 延迟模型过于简单, 因而不适合深亚微米工艺; 3) 该算法不是基于全路径的. 我们的算法克服了这些问题, 能够精确地控制最长路径延迟, 同时保证优秀的布局质量和功耗的均匀分布. 而且, 对于超大规模的电路, 我们的算法是同类算法中最快的.

EEACC: 2570

1　引言

在布图设计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最近才被列入优化目标的. 如果关键路径上的延迟太大, 芯片将不能正常工作. 因此, 近年来, 针对时延驱动的布局算法进行了广泛的研究.

以往的时延驱动布局算法可以大致分为基于线网和基于路径[1~3]的两类. 基于线网的算法通过分别控制线网的长度来控制信号路径的延迟. 一种较为常用的策略是赋予松弛量较小的线网较高的权重, 这样时延驱动的布局问题就转化为一个无约束的优化问题. 这类算法本质上是启发式的, 布局质量较差. 基于路径的算法给出了问题的准确描述, 因为集成电路中的时延问题本质上是面向路径的. Srinivasan 等[1]提出一种基于拉格朗日松弛法的方法. 基于这一方法, 每次只有一个很小子集的路径上的延迟约束是起作用的, 从而避免了路径集无限制膨胀的问题. 延迟约束可由线性不等式集合来表示. 由于绝对值函数的存在, 得到的拉格朗日问题是非光滑的. Hamada 等[2]提出的算法也是基于拉格朗日松弛法的, 其采用了非线性的延迟模型, 将问题模拟成一个非线性阻态网络, 所用的求解方法是用线性的阻态网络逼近、模拟非线性阻态网络.

所有这些基于路径的算法存在着一个共同的问题: 只能处理小电路; 因此基于路径的算法在工业界没有得到广泛应用. 最近 Sechen 等[3]提出一个基于分级思想的布局算法处理大

* 国家攻关经费及高等学校博士学科点专项科研基金资助项目
孔天明　男, 1971 年出生, 博士生, 专业: IC CAD. Email: kongtm@tiger.cs.tsinghua.edu.cn
洪先龙　男, 1940 年出生, 教授, 专业: IC CAD
1996-11-18 收到, 1997-02-04 定稿

规模电路. 他们第一次给出了包含超过 20, 000 个单元的电路的时延驱动的布局结果. 但是该算法尚存在着一些严重的问题: 1) 其基本思想要求电路是组合的, 也就是说, 电路中不能有寄存器元件. 因此, 对时序电路必须进行特殊处理: 增加一个寄存器将使得关键路径的数目迅速增长; 2) 所用的延迟模型是最简单的线性模型, 因而不能适合于目前的深亚微米工艺; 3) 该算法不是基于全路径的. 文中[3]宣称: 某些路径是逻辑上不可行的, 因而是虚假的, 应排除在外. 但是, 对于超大规模的电路来说, 预计哪些路径会是虚假的或关键的本身就是一个极为复杂的问题.

与性能比较, 低功耗优化是一个更新的目标. 只是最近随着无线通讯和计算的急剧增长, 低功耗优化才显得越来越重要. 在低功耗设计中, 主要有两点需要考虑: 1) 如何在面积和功耗之间作出选择. 对于大规模高速电路, 我们必须降低总的功耗以避免热可靠性问题. 而且降低功耗还有更显著的作用: Keutzer 等[4]指出总功耗降低 20% 可使得电路从陶瓷封装改为塑料封装, 从而使得封装成本大幅度降低. 我们知道, CMOS 的动态功耗计算公式为: $\rho_{\text{gate}} = A_{\text{gate}} \times C_{\text{gate}} \times f \times V_{\text{dd}}^2$, 式中 ρ_{gate}是门的动态功耗; A_{gate}是门的开关概率; f 是时钟频率; V_{dd}是供电电压. 显然, 降低供电电压是降低功耗的有效办法. 为了补偿由此带来的速度损失, 必须采用一些特殊的电路技术[5], 因而占用了更多的芯片面积. 因此, 我们必须在面积和功耗之间作出权衡选择; 2) 其次, 如何使功耗均匀分布. Chao 等[6]指出: 功耗的非均匀分布可能会导致局部过热, 从而引发可靠性问题. 而且, 某些对温度敏感的电路(如模拟电路中影响输出的重要参数-放大系数 β 就随温度变化而变化)也希望功耗能够均匀分布. 因此, 在布局过程中, 我们必须将功耗密度列入优化的目标之中.

以往很少有算法能够同时考虑时延和功耗优化[6, 7]. 在本文中, 我们提出一个基于全路径的时延和功耗双重驱动的 VLSI 布局算法, 该算法已成功应用在超大规模的电路布局中. 本文组织如下: 第 2 节中给出延迟模型; 问题定义和算法分别在第 3、4、5 节给出; 最后在第 6 节中给出实验结果.

2　延迟模型

电路的时钟周期是由具有最大延迟的路径(或关键路径)决定的. 在物理布图阶段, 我们仅关心这样的路径: 起始于电路的原始输入或寄存器的输出端, 终止于电路的原始输出或寄存器的输入端. 路径延迟定义为路径上所有单元到单元延迟的总和.

两个单元之间的延迟由如下部分组成: 内部延迟, 扇出和负载电容的充放电延迟, 分布 RC 参数引起的延迟. 有迹象表明: 在深亚微米工艺条件下, 连线延迟, 尤其是分布参数引起的延迟, 将成为路径延迟的主要组成部分.

我们对文献[2]中的延迟模型进行了扩充使之适用于多端线网. 假设 n 是任一线网, v_s 是驱动单元, v_i是一个负载单元, 我们定义单元 v_s到单元 v_i 的延迟如下:

$$d(v_s, v_i) = d_{bs} + \beta R_s C_L + \alpha\left(r_h c_h \sum_{j\in n} |x_j - x_s|^2 + r_v c_v \sum_{j\in n} |y_j - y_s|^2\right) + \beta R_s \sum_{j\in n}\left(c_h |x_j - x_s| + c_v |y_j - y_s|\right) + \beta \sum_{j\in n} C_j \left(r_h |x_j - x_s| + r_v |y_j - y_s|\right)$$

式中　(x_s, y_s)、(x_j, y_j) 分别表示单元 v_s 和单元 v_j 的坐标; 常数 d_{bs}是驱动单元 v_s 的内部延迟; C_j 是单元 v_j 的输入负载电容; R_s 是驱动单元的输出驱动电阻; C_L 是驱动单元的电容负

载 常数 c_h 和 c_v 分别是水平层和垂直层单位长度连线的电容; 常数 r_h 和 r_v 分别是水平层和垂直层单位长度连线的电阻 当水平层和垂直层都使用金属连线时, 电阻 r_h, r_v 和电容 c_h, c_v 的典型值分别是 0.05Ω/μm 和 2.0×10^{-4}pF/μm. 系数 α 和 β 常取的值为: 对于 90%的阈值, $\alpha=1.02$, $\beta=2.21$; 对于 70%的阈值, $\alpha=0.59$, $\beta=1.21$; 对于 62% 的阈值, $\alpha=0.5$, $\beta=1.0$ 延迟函数的一个重要特性是: 尽管该函数是非光滑的, 但却是凸的和分段光滑的

3 问题定义

给定如下的信号路径 p:

$$(v_1, v_2)\Rightarrow(v_2, v_3)\Rightarrow\cdots\Rightarrow(v_{m-1}, v_m)$$

如下的约束必须满足, 电路才可能正常工作: $\sum_{i=1}^{m-1} d(v_i, v_{i+1})\leq T$, 其中 T 是时钟周期 这样, 功耗和时延驱动的布局问题可以归结为约束优化问题: 在时延约束下, 实现连线总长和总功耗的最小化, 同时保证功耗的均匀分布, 即

$$\begin{aligned}&\min && \sum L_n+\kappa\sum\rho_n\\ &\text{subject to:} && \sum_{(v_i, v_{i+1})\in p} d(v_i, v_{i+1})\leq T,\quad \forall p\in P\end{aligned}$$

式中 $L_n=w_n\sum_{(i,j)}[(x_i-x_j)^2+(y_i-y_j)^2]$, ρ_n 分别是线网 n 的长度的二次估计式和动态功耗; κ 是反映线长和功耗的权重关系的因子; P 是所有的信号路径的集合, w_n 是目标函数中线网 n 的加权因子:

类似于分层布局, 我们用下面的步骤将功耗均匀分布的概念转化为一系列线性约束方程: 在第 l 个优化层, 芯片被划分为 m 个区域(我们用 $R_i^{(l)}$ 表示这 m 个区域的集合), 每个集合中包含一部分单元 每个单元的中心 (u_i, v_i) 对应两个线性约束:

$$\mathrm{pcx}_i=u_i,\quad \mathrm{pcy}_i=v_i$$

其含义是: 每个区域中单元的"功耗中心"对应于区域中心, 式中

$$\mathrm{pcx}_i=\sum_{j\in R_i^{(l)}}P_{wj}x_j\Big/\sum_{j\in R_i^{(l)}}P_{wj},\qquad \mathrm{pcy}_i=\sum_{j\in R_i^{(l)}}P_{wj}y_j\Big/\sum_{j\in R_i^{(l)}}P_{wj}$$

其中 $R_i^{(l)}$ 表示区域 i 中的单元集合, P_{wj} 表示单元 j 的功耗 功耗的均匀分布是通过划分区域时, 使每个区域中的单元的功耗总和大致相当来保证的

注意到: 线长和动态功耗具有不同的物理意义, κ 的含义不清楚 由 $\rho_n=A_n\times C_n\times f\times V_{dd}^2$, 而 A_n、V_{dd}^2、f 在布局阶段均为常数, 因此我们用 $\kappa\times\sum A_nC_n$ 来代替 $\kappa\times\sum\rho_n$, C_n 用如下公式估计: $C_n=c_h\sum_{i\in n}|x_i-x_s|+c_v\sum_{i\in n}|y_i-y_s|$

对每条路径 p, 我们定义一个松弛量变量 S_p, 使得延迟约束不等式成为等式:

$$\sum_{(v_i, v_{i+1})\in p} d(v_i, v_{i+1})+S_p=T$$

采用拉格朗日松弛法, 原问题可以转化为下列无约束的优化问题:

$$\max_{\lambda\geq0}\min_{X, Y} Q(X, Y, \lambda, \xi, \zeta)$$

$$\begin{aligned}Q(X, Y, \lambda, \xi, \zeta)=&\sum w_n\sum_{(i,j)}[(x_i-x_j)^2+(y_i-y_j)^2]\\&+\sum_{p\in P}\lambda_p\Big[\Big(\sum_{(v_i, v_{i+1})\in p} d(v_i, v_{i+1})+S_p-T\Big]\end{aligned}$$

$$+ \sum_{i\in R^{(l)}} \xi_i(\mathrm{pcx}_i - u_i) + \sum_{i\in R^{(l)}} \zeta_i(\mathrm{pcy}_i - v_i) + \kappa\sum_{i\in N} A_i C_i$$

式中　N 表示线网集合; λ_p 表示路径 p 对应约束方程的拉格朗日因子, ξ、ζ 分别表示对应约束方程 $\mathrm{pcx}_i = u_i$, $\mathrm{pcy}_i = v_i$ 的拉格朗日因子. 由 Kuhn-Tucker 条件, 我们有下面的定理: 令 λ_P^*, S_P^*, ξ_i^*, ζ_i^* 表示在最优点对应的值.

互补松弛定理　在最优解, (1) $\lambda_P^* \times S_P^* = 0$; (2) $\xi_i^* \times (\mathrm{pcx}_i^* - u_i) = 0$; (3) $\zeta_i^* \times (\mathrm{pcy}_i^* - v_i) = 0$.

定理表明: (1) 路径可以分为两类: 关键路径($\lambda_p^* > 0$)和非关键路径($\lambda_p^* = 0$). 在优化过程中, 我们可以忽略非关键路径. (2) 线性约束必须以等式得到满足, 故而不能忽略. 但这类约束的数目是很少的.

我们用 $Q(X, Y)$ 表示当其它变量(λ, ξ, ζ)固定时的拉格朗日问题. 这样, 我们需要求解拉格朗日问题: $\min_{X,Y} Q(X, Y)$.

4　求解拉格朗日问题

我们的关键问题是求解拉格朗日问题. 函数 $Q(X, Y)$ 是凸的和分段光滑的, 我们采用局部搜索的方法寻找局部极小解. 由于函数是凸的, 局部极小解就是全局极小解. 该算法是一个叠代算法. 在每个循环, 我们通过固定其它变量将函数视为单变量函数, 从而找到其极小点.

每个单变量函数具有如下形式:

$$g(x) = 1/2ax^2 + bx + \sum_{i=1}^{n} l_i |x - x_i|$$

其中　a, b, l_i, x_i 均为常数, 且 $a > 0$, $l_i > 0$.

不失一般性, 假设 $x_1 < x_2 < \cdots < x_n$, 则 $g(x)$ 的微分为:

当 $x \in (x_i, x_{i+1})$, $g'(x) = ax + b + 2\sum_{j=1}^{i} l_j - \sum_{j=1}^{n} l_j$;

对于 x_i 诸点, 定义: $g'^-(x_i) = \lim_{x\to x_i^-} g'(x)$, $g'^+(x_i) = \lim_{x\to x_i^+} g'(x)$;

从优化的角度出发, 我们有下面的定理:

存在性定理: 1. 如果存在区间 $[x_i, x_{i+1})$ 使得 $g'^-(x_i) \le 0$, $g'^-(x_{i+1}) > 0$, 则极小点位于该区间. 2. 如果这类区间不存在, 则: 2.1) 如果 $g'^-(x_1) > 0$, 则极小点位于 $[0, x_1)$; 2.2) 如果 $g'^-(x_n) \le 0$, 则极小点位于 $[x_n, +\infty)$. 3. 有且仅有一个极小点.

证明: 如果不考虑 x_i 诸点, 则函数 $g'(x)$ 是严格增函数.

1. 如果存在区间 $[x_i, x_{i+1})$ 使得 $g'^-(x_i) \le 0$, $g'^-(x_{i+1}) > 0$; 则根据微分的定义, 我们有:

1.1) 当 $x < x_i$, 且 $x \ne x_j$, $j = 1, \ldots, i-1$ 时, $g'(x) < g'^-(x_i) \le 0$;

当 $j = 1, \ldots, i-1$ 时, $g'^-(x_j) < g'^-(x_i) \le 0$, $g'^+(x_j) < g'^-(x_i) \le 0$;

因此 $g(x) > g(x_i)$;

1.2) 当 $x > x_{i+1}$, 且 $x \ne x_j$, $j = i+1, \ldots n$ 时, $g'(x) > g'^-(x_{i+1}) > 0$;

当 $j = i+1, \ldots, n$ 时, $g'^-(x_j) \ge (g'^-(x_{i+1}) > 0$, $g'^+(x_j) > g'^-(x_{i+1}) > 0$;

从而 $g(x) > g(x_{i+1})$;

注意到: x_{i+1} 不可能是极小点; 因此极小点位于区间 $[x_i, x_{i+1})$.

2 定理的另外两种情况显然成立

3 $g(x)$ 为凸函数, 有且仅有一个极小点

根据该定理, 我们得到“单变量优化”算法和求解拉格朗日问题的算法

算法 1: 单变量优化

对数组 x_i 按递增排序;

```
lindex = - 1; rindex = 0; b- = Σ_{i=1}^{n} l_i; acc = 0;
while(l) { /* scan to find the index */
    if (rindex > = n) break;
    if (a * x_rindex + 2 * acc + b> 0) break;
    acc + = l_rindex; lindex + + ; rindex + + ;
  }
slv = - (b + 2 * acc) / a;
if (rindex < n && rindex > 0 && slv < x_lindex) slv = x_lindex;
return slv;
```

算法 2: 求解拉格朗日

对每个单元

1). 为其 x 坐标变量建立数据(a, b, l_i, x_i);

2). 调用算法“单变量优化”得到其最优值;

3). 为其 y 坐标变量建立数据(a, b, l_i, x_i);

4). 调用算法“单变量优化”得到其最优值;

关于算法的收敛性, 我们有下面的定理:

收敛性定理 : 在算法 2 的每个叠代步 2) 和 4) 中, 函数 $Q(X, Y)$ 严格递减

5 POTIF: 功耗和时延双重驱动布局

结合以上诸算法, 我们得到算法“POTIF: 功耗和时延双重驱动布局”

算法 3 (POTIF): 功耗和时延双重驱动布局

对每一优化层次 {

1). 为当前层建立必要的初始数据;

2). 初始化变量(所有拉格朗日乘子设为 0);

3). 从 1 到 k /* k: 常数, 用来限制叠代步数 */

4). 求解拉格朗日问题;

5). 如果达到要求精度或最大循环步数, 转 7);

6). 更新拉格朗日乘子, 转 3);

7). 将每个区域划分为四个区域, 使每个所含单元的功耗之和大致相等;

}

根据设计要求, 用线性分配方法使单元按行排列;

每行长度和功耗均匀化;

在全局优化之后, 单元必须按行排列. 我们采用一种简单有效的线性分配算法将单元分配到单元行中去. 在分配过程中, 每行对应有一个约束方程: $|pr_i - \overline{pr}|$; 式中 pr_i, $\overline{pr}$ 分别表示第 i 行单元的总功耗和每行的平均功耗. 我们采用拉格朗日乘子法求解. 对于功耗较多的单元行, 分配一个具较大功耗的单元将具有较大的费用.

6　实验结果

以上算法全部用 C 语言实现. 所有的实验都在 Sun Sparc20/50 工作站上完成. 我们共测试了 5 个电路. 表 1 中列出了电路的特征参数. 表 2～6 列出了实验结果.

表 1　电路特征

电路	# iopads	# cell	# net	# row
电路 C2	373	590	963	9
电路 C5	301	1586	1887	16
电路 C7	315	2150	2465	16
电路 s13207	1490	4267	5757	24
电路 avq. small	64	21854	22183	65

表 2　C2 结果

比较准则	Ritual	POTIF
线长	4.94×10^5	4.59×10^5
延迟	21.11	21.98
功耗	210.36	228.10
均匀度	12.0%	7.4%
CPU 时间/s	109	300

表 3　C5 结果

比较准则	Ritual	POTIF
线长	1.16×10^6	1.05×10^6
延迟	34.79	32.97
功耗	922.09	828.03
均匀度	9.1%	6.7%
CPU 时间/s	481.4	889.1

表 4　C7 结果

比较准则	Ritual	POTIF
线长	1.97×10^6	1.81×10^6
延迟	46.68	45.85
功耗	1110.97	1009.8
均匀度	6.5%	6.4%
CPU 时间/s	340.4	993

表 5　s13207 结果

比较准则	Ritual	POTIF
线长	6.86×10^6	6.30×10^6
延迟	47.51	39.76
功耗	3391.49	2756.22
均匀度	10.2%	6.9%
CPU 时间/s	1501.7	2122.4

表 6　avq. small 结果

比较准则	POTIF*	POTIF
线长	1.49×10^7	1.42×10^7
延迟	103.37	104.11
功耗	3284.1	3038.7
均匀度	11.7%	9.0%
CPU 时间/s	10012.8	11410.2

在这些表中, "线长"是按"单树干 Steiner 树"模型估计的连线总长, "延迟"指延迟最长路径上的延迟, "功耗"指总的动态功耗, "均匀度"表示功耗分布的均匀度. 我们用如下公式估计"均匀度": $\sum |pr_i - \overline{pr}|/(\overline{pr}\lambda)$, 式中 λ 表示单元行的数目.

我们的比较对象是 Ritual[6] 系统. 在所有的测试例子中, POTIF 都得到了更好的布局质量(较小的"线长")和更为均匀的功耗分布. 除了电路 C2 以外, POTIF 得到的延迟(电路 s13207 中提高 16%)和总功耗(电路 s13207 中提高 19%)目标都比较好. 考虑到 Ritual 是一个时延驱动的算法, 延迟的下降量是令人惊讶的.

由于 Ritual[6] 不能处理超大规模的电路, 如 avq. small, 我们只能与自身比较. 表 6 中, POTIF* 栏表示不考虑功耗优化时的布局结果. 从表中可以看到, 当考虑功耗优化时, 总功

耗较小(7.4%),分布更为均匀,而延迟的增加量只有 1.0%.有趣的是:"线长"也减小了.

值得指出的是:在前 4 个实验中,POTIF 所需的 CPU 时间比 Ritual 所需的时间要多.但在大规模的电路上,我们的算法远远超过了 Ritual,而且只要求很少的系统资源.在计算电路 avq.small 时,POTIF 只需 20M 内存,而文献[3]中则需超过 60M 的内存.对于电路 avq.small,在所有已发表的算法中,我们的算法速度是最快的.

参 考 文 献

[1] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm for Small Cell ICs", Proc. ICCAD, November 1991, 48~51.

[2] T. Hamada, C. K. Cheng and P. M. Chau, "Prime: A Timing-Driven Placement Tool using A Piecewise Linear Resistive Network Approach", Proc. 30th DAC, 1993, 531~536.

[3] W. Swartz and C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits", Proc. 32nd DAC, 1995, 211~215.

[4] K. Keutzer and P. Vanbekbergen, "The Impact of CAD on the Design of Low Power Digital Circuits", IEEE Symp. on Low Power Electronics, 1994, 42~45.

[5] A. P. Chandrakasan, S. Sheng and R. W. Bodersen, "Low-Power CMOS Digital Design", IEEE J. Solid-State Circuits, 1992, **27**(4): 473~484.

[6] K. Y. Chao and D. F. Wong, "Low Power Consideration in Floorplan Design", Proc. IWLPD, 1994, 45~50.

[7] V. Hirendu and M. Pedram, "PCUBE: A Performance Driven Placement Algorithm for Lower Power Designs", Proc. Euro-DAC, 1993, 72~77.

POTIF: Efficient Power and Timing Driven Placement Algorithm

Kong Tianming, Hong Xianlong and Qiao Changge

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Received 18 November 1996, revised manuscript received 4 February 1997

Abstract A new efficient power and timing driven placement algorithm for very large scale small cell-based ICs has been reported. Previously, very few approaches tend to handle power and timing issues simultaneously. Among previous approaches to timing driven placement, only one algorithm can handle large scale circuits, while it has three important problems: 1) the basic idea can only handle circuits whose timing graph is a DAG (Directed Acyclic Graph); 2) the delay model is rather simple, thus not appropriate for deep submicron circuits; 3) it is not an all-path based timing-driven placement algorithm. Our algorithm can accurately control the longest path delay for very large scale circuits, while yielding excellent placement qualities and guaranteeing evenness of power distribution. Besides, our algorithm is the fastest for large scale circuits ever reported.

EEACC: 2570