面向甚大规模集成电路的时延驱动布局方法

吴为民[1], 洪先龙[1], 蔡懿慈[1], 顾 钧[2]
(1. 清华大学计算机科学与技术系, 北京 100084; 2. 香港科技大学计算机科学系, 香港)

摘 要: 本文针对甚大规模集成电路的时延驱动布局问题提出了一个新的解决途径, 其策略是将结群技术应用于二次规划布局过程中. 结群的作用是可大幅度地降低布局部件的数量. 本文设计了一个高效的结群算法 CARGO, 其优点是具有全局最优性并且运行速度很快. 采用了一个基于路径的时延驱动二次规划布局算法对结群后的电路完成布局过程. 由于二次规划布局算法能够在很短时间内寻找到全局最优解, 故本文的算法更有希望彻底解决甚大规模电路的布局问题. 在一组 MCNC 标准测试电路上对算法进行了测试, 得到了满意的结果.

关键词: 结群; 二次规划; 时延驱动; 布局
中图分类号: TP302 文献标识码: A 文章编号: 0372-2112 (2001) 08-1018-05

Timing-Driven Placement Algorithm for Very Large Integrated Circuits

WU Wei-min[1], HONG Xian-long[1], Cai Yi-ci[1], GU Jun[2]
(1. *Dept. of Computer Science & Technology, Tsinghua University, Beijing, 100084, China;*
2. *Dept of Computer Science, Hong Kong University of Science and Technology, China*)

Abstract: A novel approach for timing-driven placement of very large integrated circuits is presented. Our strategy is to apply clustering technique to a quadratic placement procedure. By clustering, the number of components for placement is reduced considerably. We design a high-efficiency clustering algorithm, named CARGO, which has global optimality and runs very fast. We use a path-based timing-driven quadratic placement algorithm to complete the placement of the condensed circuit. Because quadratic based placement algorithm can mathematically find global optima in very short time, our new approach might be more promising for solving the problem thoroughly. We have tested our algorithm on a set of MCNC circuits and obtained satisfactory results.

Key words: clustering; quadratic programming; timing-driven; placement

1 引言

在当前的深亚微米甚至超深亚微米工艺下, 集成电路集成度的不断提高使现今的布局算法越来越难以胜任. 一方面, 在单个芯片上需布局的单元数量急剧增大, 使现有算法不能在合理的时间内完成布局. 另一方面, 特征尺寸的不断减小使得单元之间的连线时延成为决定芯片性能的主要因素, 因而, 时延驱动的布局更为重要. 将时延作为附加约束引入到布局过程中无疑会进一步降低布局进程.

为解决甚大规模电路的布局问题, 最常采用的策略是"分而治之". 这种策略一般采取划分或结群两种途径之一. 对于划分途径, 其思想是通过对芯片进行连续的划分过程, 使单元不断地限制在更小的区域内[1,2]. 划分技术一般依赖于一个启发式的二划分例程, 如 KL[3] 或 FM[4] 算法. 然而, KL/FM 算法易于陷入局部最优解, 尤其对于大规模电路, 从而影响布局质量. 对于结群途径, 其思想是通过将结合紧密的单元结合在一起, 作为一个整体参与布局, 从而减小原始电路的规模. Mallela 和 Grover 首先将结群技术应用于模拟退火布局算法中[5]. 他们在结群的前后采用了不同的模拟退火布局过程, 得到了很好的结果. 后来, Sun 及 Tsay 等人也分别提出了采用结群技术和模拟退火布局算法相结合的分层布局方法, 并取得了较好的结果[6,7]. 虽然结群技术和模拟退火布局技术的结合是成功的, 但我们认为, 模拟退火算法由于其内在的随机搜索本质, 将在未来面临很大的困难.

自从二次规划布局算法诞生以来, 受到日益广泛的重视, 成为当前的主流布局算法. 比较著名的有 RITUAL[8]、Gordian[9] 及 Prime[10] 等系统. 这种算法可在较短的时间内得到全局最优解. 此外, 一些物理限制, 如时延、功耗等可自然地作为布局约束条件加入到问题描述中. 由于二次规划布局算法的这些优点, 我们认为将其与结群技术结合更合理, 更利于解决甚大规模集成电路的布局问题. CASH 是在这方面的首次尝试[11], 但 CASH 未考虑时延, 而且当结群尺寸设得较大时, 结群时间过长.

在深亚微米时代, 互连线时延超过了门时延, 成为决定芯片性能的主要因素. 因此, 非时延驱动的布局算法实际上是无

收稿日期: 2000-08-14; 修回日期: 2000-11-20
基金项目: 国家自然科学基金(No. 69776027); 973 国家关键项目(No. G1998030411)

意义的. 目前的时延驱动布局算法或者是基于线网的[12, 13], 或者是基于路径的[8, 10]. 由于集成电路中的时延问题本质上是面向路径的, 因此基于路径的布局算法对问题的描述和解决更为直接和准确.

本文将结群技术应用于基于路径的时延驱动二次规划布局过程中. 整个布局过程由结群、总体布局、改善布局及详细布局四个步骤组成. 由于前面提到的二次规划布局算法的优点, 这种结合更有希望彻底解决问题. 然而, 这种应用的可行性在很大程度上依赖于两个事实: 一个是结群时间必须足够小, 使得其与布局时间相比可忽略不计; 二是布局质量不应因总体布局自变量数的减少而过度下降.

2　电路结群

首先将每个单元看作一个群, 然后不断地选取连接最紧密的单元对, 并将相应的群合并. 结群过程分为两个阶段: (1) 建立单元连接信息库. (2) 群合并. 通过预先建立单元连接信息库, 使得不仅每次都能够选取最紧密的单元对进行结群, 并且能够避免在结群过程中大量的冗余计算. 因此, 将本文的结群算法命名为 CARGO(Clustering Algorithm Relies on Global Optimization).

2.1　建立单元连接信息库

设 C 为单元集合, N 为线网集合. 对于一个单元对 (c_i, c_j), 两个单元之间的连接度计算如下:

$$CON_1(c_i, c_j) = \sum_{n_k \in n_{i,j}} w_k \cdot R_k \qquad (1)$$

其中 $n_{i,j}$ 是连接单元 c_i 和 c_j 线网集合. w_k 线是线网 n_k 的权值, 此值是与线网长度相关的因素. R_k 表示是线网 n_k 的驱动管脚的电阻, 此值是与时延直接相关的因素(见式 9).

在计算完所有有连接关系的单元对的连接度之后, 首先将这些单元对按连接度大小分为 H 组. 这样, 所有单元对可被存于一个二维数组 $BUCK[m][n]$ 中, 其中 m 是组索引, n 是单元对在组内的索引. 对于任一单元对 (c_i, c_j), 其组索引可以下式求得:

$$m(c_i, c_j) = \lfloor (C_{max} - C_{min})/H \rfloor \qquad (2)$$

其中 C_{max} 和 C_{min} 分别是所有单元对连接度的最大和最小值.

然后, 对每个组内的单元对进行排序, 确定每个单元对在组内的索引. 排序的准则是根据内- 外连接度, 计算公式如下:

$$CON_2(c_i, c_j) = \frac{CON_1(c_i, c_j)}{\sum_{con(c_i, s_k) \neq 0} CON_1(c_i, c_k) + \sum_{con(c_j, s_k) \neq 0} CON_1(c_j, c_k)} \qquad (3)$$

2.2　群合并

假设单元数量为 N^c, 要求的群数量为 N^{clu}, 群内单元数量限制为 S_{max}. 包含单元 c_i 的群设为 $Clu(c_i)$, 群内已有的单元数量为 $S(Clu(c_i))$. 群合并算法的伪码表示可见图 1.

```
Algorithm: Cluster Combining
Input: Original netlist PN = (C, N), BUCK, S_max.
Output: A clustered netlist CN = (C_c, N_c).
Begin
  Create an initial cluster netlist, with each cluster
  has only one cell;
  Set index = 0;
  for (i = H to 0)
    if (index = = N^c - N^clu) break;
    for (each cell pair (C_i, C_j) at the Buck height i)
      if Clu(C_i) = Clu(C_j) or S(Clu(C_i) + S(Clu(C_j)) > S_max
        continue;
      else
        merge Clu(C_i) and Clu(C_j) together;
      index + + ;
      if (index = = N^c - N^clu) break;
    endfor;
  endfor;
  Create N_c from N;
  return CN = (C_c, N_c)
```

图 1　群合并算法

在理想情况下, 群合并的次数为 $N^m = N^c - N^{clu}$. 但在大多数情况下, 在群合并次数达到 N^m 之前已找不到可合并的单元对. 若任一单元连接最多 L 个其它单元, 则计算所有单元对连接度的计算复杂度为 $O(N^c \cdot L)$. 对各组内单元对排序的计算复杂度为 $O(H^v N_g \log_2 N_g)$, 其中 H^v 是所有包含至少一个单元对的组的数量, N_g 是组内最大单元对数量. L 一般较小, 连接过多单元的大线网可略去.

3　总体布局

3.1　问题描述

二次规划的目标函数可以矩阵形式表示如下:

$$L(x, y) = (1/2)(x^T Q x + y^T Q y) + c^T x + d^T y \qquad (4)$$

其中 $Q = [q_{ij}]$, 并且

$$q_{ij} = \begin{cases} \sum_k a_{ik}, & \text{if } i = j \\ -a_{ij}, & \text{otherwise} \end{cases}$$

及 $c = [c_1, \cdots, c_m]$, 并且

$$c_i = \sum_{m < k < m+n} a_{ik} x_k, \text{ and } a_{ij} = \sum_{i,j \in n,\, n \in N} w_n$$

其中矢量 x 和 y 表示各群的坐标, m 是群的数量, n 是 PAD 的数量. d 的表达式与 c 类似. w_n 表示线网 n 的权值, 目前采用下式计算:

$$w_n = 2/|T_n| \cdot |T_n - 1| \qquad (5)$$

其中 T_n 是线网 n 的端点数.

为使各群均匀地分布在芯片上, 需要加入分布约束. 在第 l 划分层次, 芯片被划分成 4^l 个区域, 相应的所有群也根据其坐标划分成 4^l 个集合. 这样, 每个群集合根据其坐标位置被分配到一个区域中. 假设 s_j 是一个区域, 包含 $|s_j|$ 个群, 区域的中心坐标为 (r_j^x, r_j^y), 则可对此区域施加如下两个线性的分布约束:

$$\frac{1}{|S_j|} \sum_{i \in S_j} x_i = r_j^x, \ \frac{1}{|S_j|} \sum_{i \in S_j} y_i = r_j^y \qquad (6)$$

假设 CRI 是关键路径集合. 若 π_{se} 是其中任何一条关键路径, 始于单元 s, 终止于单元 e, T_{se} 是其时延上限, 则时延约束表示为:

$$d(\pi_{se}) \leq T_{se}, \ \pi_{se} \in CRI \qquad (7)$$

综上所述, 时延驱动的布局问题可表示为

$$\text{Min}\{L(x, y) \mid A^{(l)} X = u^{(l)}, \ d(\pi_{se}) \leq T_{se}, \ \pi_{se} \in CRI\} \qquad (8)$$

3.2 时延模型

采用与文献[8]相同的时延模型. 线网 i 的时延采用下式计算:

$$d_i = R_i\left(C_i^{\text{net}} + C_i^{\text{load}}\right) \tag{9}$$

$$C_i^{\text{net}} = \sum_{j\in[i1,\cdots,ik]} \left(C_h|x_i - x_j| + C_v|y_i - y_j|\right) \tag{10}$$

$$R_i = \sum_{j\in[i1,\cdots,ik]} \left(R_h|x_i - x_j| + R_v|y_i - y_j|\right) \tag{11}$$

其中 C_i^{load} 是线网 i 的驱动电容, C_i^{net} 是线网 i 的连线电容, C_h 和 R_h 分别表示单位长度水平连线的电容和电阻, C_v 和 R_v 分别表示单位长度垂直连线的电容和电阻.

3.3 求解方法

式(8)的二次规划问题可采用拉格朗日松弛法解决. 时延驱动或时延约束的引入首先提出一个问题: 以何种方式将时延加入到算法中? 著名的时延驱动布局系统 RITUAL 采取的方法是: 将时延以与分布约束类似的方式加入到方程中. 由于时延约束带有绝对值函数, 需要在布局过程中根据各相关单元的位置将其变为线性方程. 由于关键路径数及所涉及到的单元数(包括关键路径上的单元及其所驱动的所有单元)很庞大, 所以实现起来非常繁琐费时.

为避开绝对值函数带来的困难, 采取将时延约束加到矩阵 $\boldsymbol{Q}$ 中办法. 由于 $\boldsymbol{Q}$ 的各项就是表示相应单元对之间的紧密程度. 故将时延因素直接引入到 $\boldsymbol{Q}$ 是很自然的方法. 这样, 原始问题可转化为如下的拉格朗日问题:

$$\max_{\lambda_D,\lambda_R\geq 0}\min_{x,y} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{Q}\boldsymbol{y} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{y} + \lambda_{R,x}^T(\boldsymbol{e}^T\boldsymbol{x} - \boldsymbol{r}_x) + \lambda_{R,y}^T(\boldsymbol{e}^T\boldsymbol{y} - \boldsymbol{r}_y) \tag{12}$$

其中, $\boldsymbol{Q} = \boldsymbol{Q} + \lambda_D(d - d_{\max})$, λ_R 和 λ_D 为拉格朗日乘子.

对于固定的 λ_R 和 λ_D, 可得到式(12)的解如下:

$$\boldsymbol{x}^{(k+1)} = -\boldsymbol{Q}^{-1}[\lambda_{R,x}^{(k)}\boldsymbol{e}^T + \boldsymbol{c}] \tag{13a}$$

$$\boldsymbol{y}^{(k+1)} = -\boldsymbol{Q}^{-1}[\lambda_{R,y}^{(k)}\boldsymbol{e}^T + \boldsymbol{d}] \tag{13b}$$

在前后迭代之间, λ_R, λ_D 采用下式修改:

$$\lambda_{R,x}^{(k+1)} = \max\{0,\ \lambda_{R,x}^{(k)} + t_x^{(k)}((\boldsymbol{e}^T\boldsymbol{x}^{(k)} - \boldsymbol{r}_x))\} \tag{14a}$$

$$\lambda_{R,y}^{(k+1)} = \max\{0,\ \lambda_{R,y}^{(k)} + t_y^{(k)}((\boldsymbol{e}^T\boldsymbol{x}^{(k)} - \boldsymbol{r}_y))\} \tag{14b}$$

$$\lambda_D^{(k+1)} = \max\{0,\ \lambda_D^{(k)} + t_D^{(k)}(-slack)\} \tag{14c}$$

其中 $t_x^{(k)}$, $t_y^{(k)}$ 和 $t_D^{(k)}$ 是标量步长. $slack = d - d_{\max}$ 是相应路径的实际时延减去时延上限的差值.

由于在求解过程中每次迭代只有一部分约束起作用, 故需经常作时延验证(timing verification), 以便发现新的关键路径集合及其 $slack$, 用来修改拉格朗日乘子 $\lambda_D^{(k)}$. 时延验证方法是根据各单元位置坐标及驱动线网的驱动电容、电阻, 互连线的电容、电阻以及时钟周期, 计算出各线网的实际信号到达时间和要求的信号到达时间, 两者的差就是 $slack$. 若某线网的 $slack$ 为负数, 就意味着相应的单元不满足时延约束, 此线网或单元是关键线网或关键单元, 任何通过关键线网的路径都是关键路径. 时延验证应在单元层次上进行, 然后再将结果映射到相应的群层次上. 这样做的原因是群层次的线网可能存"环", 不能做时延验证.

一个需要注意的问题是在修改 $\boldsymbol{Q}$ 时应保持其对称性. 通过修改 $\boldsymbol{Q}$ 矩阵, 加重关键路径上的线网权值, 使得在下次优化过程中关键线网所驱动的单元更趋近于驱动单元. 这样, 就使得在上一小节的时延模型下, 电流通过线网的时延能够逐步降低.

若 $I_{\max}$ 表示设定的每一层次的最大迭代次数, $H_{\max}$ 表示设定的最大层次数, 则求解过程的具体步骤可见图 2.

(1) 置初始划分层次 $H = 0$, 拉格朗日乘子 $\lambda_{R,x} = \lambda_{R,y} = \lambda_D = 0$;
(2) 设初始关键路径集为空;
(3) 对层次 H 施加分布约束, 设置迭代次数 $I = 0$;
(4) 对式(13a)和(13b)用共轭梯度法求解;
(5) 对电路进行时延验证, 确定新的关键路径集;
(6) 依式(14a)、(14b)及(14c)修改拉格朗日乘子, $\lambda_{R,x}$, $\lambda_{R,y}$ 及 λ_D;
(7) 若 $I > I_{\max}$, 或者已满足分布约束, 且 $H < H_{\max}$, 则置 $H = H + 1$, 然后转步骤 3;
(8) 若已满足时延约束, 则结束算法;
(9) 置 $I = I + 1$, 然后转步骤 4;

图 2 总体布局的求解过程

4 改善布局

总体布局后, 同一群内的各单元处于同一位置. 可以通过解散群进一步提高布局质量和减小单元之间的重叠. 改善布局通过两个步骤完成: (1) 改善时延. (2) 改善线长.

改善时延优先进行. 首先标记所有单元为 FREE, 然后对电路做时延验证, 并标记所有关键单元为 FIXED. 接着对于每个标记为 FIXED 的单元 c_i, 通过将所有 c_i 驱动的 FREE 单元向 c_i 移动来降低 $slack$. 比如, 若 c_j 是标记为 FREE 且被 c_i 驱动, 则可通过下式移动 c_j:

$$x_j^{k+1} = x_j^k + \alpha_t \cdot (x_i^k - x_j^k) \tag{15}$$

其中 $0 < \alpha_t < 1$. 以上的例程应迭代进行多次以进一步改善关键路径时延和减小单元之间重叠.

在改善线长步骤, 首先要做的是对电路进行时延验证, 并标记所有关键单元为 FIXED. 对于每个被标记为 FREE 的单元 c_i, 采用下式计算其最佳位置(具体计算方法见文[11]):

$$x_i^g = \left(\sum_{n\in N, i\neq j} w_n \cdot x_j\right)/(n-1) \tag{16}$$

然后, 采用下式将 c_i 向此最佳位置移动:

$$x_i^{k+1} = x_i^k + \alpha_w \cdot (x_i^g - x_i^k) \tag{17}$$

其中 $0 < \alpha_w < 1$. 以上例程应迭代进行多次以进一步改善线网总长和减小单元之间重叠.

5 详细布局

详细布局算法继承改善布局算法的结果, 并将各单元无重叠地分配到单元行上. 本文采用的详细布局算法结合了一系列的局部优化策略, 并通过线网加权的方法处理时延约束问题. 也就是说, 在每次优化迭代完成后, 需要进行时延验证并依次动态修改线网权值, 并结合采用了若干种修改线网权值的方法. 详细布局方法的具体描述可见文献[14].

6 实验结果及分析

本文的算法采用 C 语言实现并在一台 SUN Enterprise450 工作站上运行. 测试电路采用一组 MCNC 标准测试实例. 这里给出并分析其中最大的三个电路 C7, s13207, avq 的测试结果.

测试电路的参数如表 1 所示.

表 1　电路参数

电路	单元数	管脚数	线网数	单元行数
C7	2150	315	2465	16
S13207	4267	1490	5757	24
Avq	21854	64	22183	65

首先通过分析其中最大电路 avq 的测试结果, 对结群性能进行考察. 见表 2. 由表中可得出以下结论: (1) 结群时间基本上不随结群尺寸的改变而变化. 这是因为结群的时间主要耗费在建立单元连接库的过程中, 而这个过程与结群尺寸没有关系. 由于 CARGO 算法的特点, 结群尺寸仅对单元合并过程有轻微影响. (2) 详细布局时间由于其不依赖于初始结果的特点, 基本上也不随结群尺寸变化. (3) 随着结群尺寸的增大, 群数量相应减小, 总体布局时间也相应减小. 但是, 群数量和总体布局时间并不是随结群尺寸的增加而成比例地变化. 这是因为随结群尺寸的增大, 总会出这样一些不能合并的单元对. 它们所在群的单元数量均不超过设定的群尺寸, 但它们之中任何一个单元对的单元之间或者不存在任何连接关系, 或者它们合并后产生的新群所包含的单元数大于设定的群尺寸. 因此, 一般来说, 随着结群尺寸的增大, 群数量及总体布局时间的改善相对来说越来越缓慢. (4) 结群时间远小于总体布局时间. 随着结群规模呈 3, 5, 7 的变化, 结群时间分别仅占总布局时间的 0.12%, 0.17%, 0.19%. 可见 CARGO 是完全满足在运行时间上的要求的. (5) 改善布局时间随结群尺寸的增大而增大. 因为较大的结群尺寸意味着总体布局后的重叠性也较大, 需要更多的迭代次数进行改善布局. (6) 布局结果并没有因为结群而过度变坏. 从 avq 的结果来看, 总线长最大增加 5.6%, 最大时延最大增加 2.6%. 而且, 随着结群规模增大, 总线长和最大时延并不一定单调增加. 这首先是得益于结群算法的全局优化性, 即结为一群的单元都是连接最紧密的单元. 其次, 在改善布局阶段的局部优化过程也大大减小了结群的影响. 一般来看, 线长增加得大, 则时延增加得小. 反之亦然.

实际上, 以上对 avq 的分析结果同样适用于其它几个 MCNC 电路, 具有一定的普遍性.

表 2　在 avq 上的结群性能

结群性能	结群尺寸			
	1	3	5	7
群数量	21854	8645	5253	3938
结群时间(s)	0	3.5	3.4	3.3
总体布局时间(s)	4600	2501	1873	1584
改善布局时间(s)	0	21	27	31
详细布局时间(s)	129	129	129	130
总布局时间(s)	4729	2716	2032	1748
总线长(m)	10.5	10.7	11.1	10.8
时延(ns)	83.1	85.3	81.3	84.2

最后, 在相同的布局条件下, 将本文的布局结果与 RITUAL 的结果做一下比较. 见表 3. 这里用结群算法 CARGO 命名本文的布局算法. 结群尺寸是适当选择的. 可以看出, CARGO 在 C7, s13207 及 avq 上的布局结果都远远好于 RITUAL 的结果. 从总线长来看, CARGO 在 C7 上改善了 15%, 在 s13207 上改善了 5.1%. 从最大时延来看, CARGO 在 C7 上改善了 27%, 在 s13207 上改善了 35%. 可见, CARGO 在最大时延方面的改善更显著. 从布局时间来看, CARGO 的运行时间在 C7 上仅是 RITUAL 运行时间的 40%, 在 s13207 上仅是 RITUAL 运行时间的 22%. RITUAL 在 avq 上不能在合理的时间内得到解, 而 CARGO 仅在约 34 分钟内得到解.

表 3　与 RITUAL 的性能比较(各项的单位与表 2 的相同)

电路	RITUAL			CARGO		
	总线长	时延	CPU 时间	总线长	时延	CPU 时间
C7	1.97	46.7	340	1.67	34.3	135
s13207	6.86	47.5	1501	6.51	30.8	334
Avq	–	–	–	11.1	81.3	2032

7　结论

实验结果证明本文提出的面向甚大规模集成电路的时延驱动布局算法是成功的. 也就是说, 通过结群方式缩小电路规模后, 仍能够得到相当好的布局质量, 同时使布局时间大大降低. 由于对于甚大规模电路来说, 总体布局过程需消耗比改善布局和详细布局更多的 CPU 时间, 故本文的算法有望成为解决甚大规模布局问题的有效途径.

参考文献:

[1] P R Suaris, G Kedem. An algorithm for quadrisection and its application to standard cell placement [J]. IEEE Trans. Circuit Syst., 1988, 35: 294–303.

[2] A E Dunlop, B W Kernighan. A procedure for placement of standard cell VLSI circuits [J]. IEEE Trans. on CAD, 1985, 4(1): 92–98.

[3] B W Kernighan, S Lin. An efficient heuristic procedure and its partitioning graphs [J]. Bell Syst. Tech. J., 1970: 291–307.

[4] C M Fiduccia, R M Matheyses. A linear-time heuristic for improving network partitions [A]. 19th Design Automation Conference [C], 1982: 175–181.

[5] S Mallela, L K Grover. Clustering based simulated annealing for standard cell placement [A]. Proc. 25th DAC [C], 1988: 313–317.

[6] W J Sun, C Sechen. Efficient and effective placement for very large circuits [J]. IEEE Trans. on CAD, 1995, 14(3): 7349–7359.

[7] Y W Tsay, Y L Lin. A row-based cell placement method that utilizes circuit structural properties [J]. IEEE Trans CAD, 1995, 14(3): 393–397.

[8] A Srinivasan, K Chaudhary, E S Kuh. RITUAL: a performance driven placement algorithm [J]. IEEE Trans. Circuit Syst., 19912, 39(11): 825–840.

[9] J M Kleinhans, G Sigl, F M Johannes, K J Antreich. GODIAN: VLSI placement by quadratic programming and slicing optimization [J]. IEEE Trans. CAD, 1991, 10(3): 356–365.

基于 DDS 的宽带雷达信号产生技术研究

费元春[1], 陈世伟[2], 米 红[1]
(1. 北京理工大学, 北京 100081; 2. 装备部指挥技术学院, 北京 101416)

摘 要: 本文研究了基于 DDS 的宽带雷达信号产生的几项关键技术: 包括提高 DDS 工作频率上限, 展宽带宽、改善频谱、建立雷达波形库、通过 DSP 实现对 DDS 输出波形的控制. 提出了一种宽带捷变雷达信号产生的方案. 并给出了产生 0~ 25MHz、200~ 650MHz、1200~ 1650MHz 宽带雷达复杂信号的研究结果. 相关技术已在一些工程中应用, 获得满意结果.

关键词: 直接数字合成; 扩展频带; 改善频谱; 控制电路; 雷达波形库

1 引言

理论研究和实践表明, 雷达的性能及其雷达的四抗能力和雷达信号带宽与信号形式紧密相关. 例如为了提高测距精度和距离分辨力、对目标成像识别, 要求雷达发射的信号具有大的带宽、时宽乘积——宽脉冲内附加线性调频或非线性调频信号, 以扩展信号频带; 又如, 为了提高雷达对抗有源干扰的能力, 要求产生宽带捷变频信号, 以避开干扰频谱, 有效对抗瞄准式干扰等; 由此可见, 研究宽带雷达信号产生技术对雷达性能提高有重要意义. 就宽带捷变频雷达信号产生而言, 传统的方法是模拟技术. 近几年, 新的方法是用数字技术产生, 直接数字合成技术 (Direct Digital Synthesis, 简称 DDS) 便是其中一种新的方法. 由于 DDS 采用全数字结构, 使它具有极高的频率分辨率 (达 Hz、mHz)、极短的频率转换时间 (可达 ns 量级)、输出频率相对带宽很宽、具有任意波形输出能力和程控灵活等优点, 是传统的模拟信号产生技术无可比拟的. 但 DDS 是一种新技术, 目前输出频率还不高, 它的的全数字结构, 又带来了杂散电平和谐波电平高的缺陷, 因而目前在雷达中应用还很少. 本文的内容是研究 DDS 产生宽带雷达信号的几项关键技术, 并围绕工程应用的需要, 解决 DDS 频谱不纯和工作频率不高等问题. 结合作者近几年研究工作的体会, 提出了改善频谱、提高输出频率、扩展工作带宽的方法, 并给出 0~ 25MHz、200~ 650MHz、1200~ 1650MHz 宽带雷达复杂信号产生的方案与研究结果.

2 基于 DDS 的宽带雷达信号产生的关键技术

2.1 直接数字合成 (DDS) 技术

DDS 是随数字集成电路和微电子技术的发展出现的一种新的数字频率合成技术, 它从相位量化的概念出发进行合成. 其内部组成包含相位累加器、波形存储器、数模转换器、低通滤波器和参考时钟五部分. 在参考时钟的控制下, 相位累加器依频率控制字 K 产生拟合成信号数字化的线性相位取样值, 对波形存储器寻址, 使相位码转换为对应波形的幅度码, 经过数模转换器得到模拟阶梯波, 最后经低通滤波器得到所需频率的调幅、调频、调相波形.

设 f_c 为时钟频率, N 是相位累加器的字长, 则 DDS 输出信号的频率 f_{out} 及最小频率分辨率 Δf_{min}、最小相位分辨率 $\Delta\Phi_{min}$ 可表示为: $f_{out} = k \cdot f_c / 2^N$、$\Delta f_{min} = f_c / 2^N$、$\Delta\Phi_{min} = 2\pi / 2^N$.

(下转第 1027 页)

基金项目: 国防预研基金课题资助

[10] T Hamada, C K Cheng, P M Chau. Prime: a timing-driven placement tool using a piecewise linear resistive network approach [A]. Proc. 30th DAC [C], 1993: 531- 536.
[11] X L Hong, et al. CASH: A novel quadratic placement algorithm for very large standard cell layout design based on clustering [A]. Proc. 5th Int. Conf. On Solid-State and Integrated ocuit technology [C], 1998: 497- 501.
[12] A E Dunlop, V D Agrawal, et al. Chip layout optimization using critical path weighting [A]. Proc. 21th DAC [C], 1984: 133- 136.
[13] S Sutanthavibul, E Shragowize. An adaptive timing-driven layout for high speed VLSI [A]. Proc. Of 27th DAC [C], 1990: 90- 95.
[14] Bo Yao, et al. Timing-driven detailed placement for standard-cells based on lookup table delay model [A]. Proc. WCG-ICDA 2000 [C], 2000: 311, 315.

作者简介:

吴为民 男. 1966 生于吉林省辽源市. 1989 年毕业于吉林大学计算机科学系, 1992 年在哈尔滨工程大学获硕士学位, 1995 年在哈尔滨工业大学获博士学位. 1996 年至 2000 年分别在浙江大学和清华大学做博士后. 现为清华大学计算机科学与技术系副教授. 主要研究领域是超大规模集成电路布图设计.

洪先龙 男. 1940 年出生于中国浙江省. 1964 年毕业于清华大学数学力学系, 现任清华大学计算机科学与技术系教授, 博士生导师. 多年来从事 IC- CAD 的研究和教学工作, 主要研究领域是布图算法和 VLSI CAD 系统.