蚁群与粒子群混合的 FPGA 布局算法

赵 军,贾智平
ZHAO Jun,JIA Zhi-ping

山东大学 计算机科学与技术学院,济南 250101
Department of Computer Science and Technology,Shandong University,Jinan 250101,China
E-mail:zhaojun_sdu@yahoo.com.cn

ZHAO Jun,JIA Zhi-ping.Mixed ant colony and particle swarm FPGA placement algorithm.Computer Engineering and Applications,2009,45(18):70-71.

Abstract: Placement is a key issue in FPGA CAD flow.A mixed ant colony and particle swarm algorithm is used for the placement of FPGA.This method is utilized to place a set of Microelectronics Center of North Carolina(MCNC) benchmark circuits, and this paper presents a comparison with Simulated Annealing algorithm(SA),mixed Genetic and Simulated Annealing algorithm (GASA) and Ant Colony algorithm (ACO).The experimental results show that the placement method can achieve performance in terms of placement cost and routing channel density.

Key words: Field Programmable Gate Array(FPGA) placement;particle swarm;ant colony

摘 要:FPGA 布局在自动化设计中起到了十分关键的作用。将粒子群蚁群混合算法应用于 FPGA 布局问题,针对 MCNC 基准电路进行布局实验,并与模拟退火算法(SA),模拟退火与遗传混合算法(GASA)及蚁群算法(ACO)等进行了对比。结果表明该布局方法具有较好的性能。

关键词:现场可编程门阵列布局;粒子群;蚁群
DOI:10.3778/j.issn.1002-8331.2009.18.022 **文章编号:**1002-8331(2009)18-0070-02 **文献标识码:**A **中图分类号:**TP301.6

1 引言

布局在 FPGA 自动化设计过程中起到了十分关键的作用,因为布局质量的高低直接影响到了布线及 FPGA 的整体性能。由于布线资源占用了 FPGA 约 70%~80%的芯片面积和约 50%~60%的信号时延, 而一个好的布局算法能够减少布线拥挤并最小程度减少布线资源的占用, 因此在工艺条件一定的情况下,布局算法对 FPGA 的设计起着至关重要的作用。尽管 FPGA 的设计过程越来越简单,但 FPGA 布局问题仍然是 NP-hard 问题。近年来,各国学者提出许多相关的算法,例如,模拟退火算法(SA)[1],遗传算法(GA)[2],模拟退火与遗传混合算法(GASA)[3],蚁群算法(ACO)[4],粒子群算法(PSO)[5-6]及模拟退火改进算法(GSA)[7]等。

蚁群算法和粒子群算法是 20 世纪 90 年代才提出的一种新型的智能算法。蚁群算法由意大利学者 M.Dorigo 于 1992 年首次提出,并用该方法解决了一系列组合优化问题。蚁群算法具有正反馈、分布式计算及与某种启发式算法相结合的特点。正反馈有利于该算法更快地发现较好解,分布式计算有利于实现并行计算,而与启发式算法相结合使得该算法易于发现更好的解。文献[3]第一次将蚁群算法应用于 FPGA 布局问题中,取得比较理想的效果。但是蚁群算法也存在一些缺点,如收敛速度慢、易陷于局部最优等。

粒子群算法最早是由 Kenney 与 Eberhart 于 1995 年提出的,源于对鸟群捕食的行为研究,同遗传算法类似,是一种基于迭代的优化工具。文献[4-5]第一次将粒子群算法应用于 FPGA 布局问题中, 结果显示该算法在处理布局问题时具有很大潜力。与遗传算法相比,PSO 算法优点在于流程简单易实现,算法参数简洁,无需复杂的调整。

将蚁群与粒子群混合算法应用于 FPGA 布局问题,即在蚁群布局算法中引入粒子群优化算法思想,称为粒子群蚁群布局算法(PSAC)。核心思想是让蚂蚁具有“粒子”的特性,首先随机产生若干解生成信息素分布,然后由蚁群算法根据更新的信息素找出较优的解,再由粒子群优化算法进行交叉、变异操作,得到更有效的解。

2 FPGA 布局问题描述

FPGA 布局过程是将电路网表中的单元,包括逻辑单元及输入输出单元,布置到 FPGA 芯片指定的位置中去,如图 1 所示。

FPGA 布局问题一般数学描述如下[8],给定一个具有 n 个模型(单元)的集合 $M=\{m_1,m_2,\cdots,m_n\}$和一个具有 r 个端口的集合 $S=\{s_1,s_2,\cdots,s_n\}$,M 中的模型代表 CLBs 或 IOBs,将模型与端口关联起来,一个模型 $m_i\in M$ 具有端口集 $S_{m_i}(S_{m_i}\subseteq S)$。也

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.90718032)。
作者简介:赵军(1979-),男,研究生,主研领域为 FPGA、嵌入式系统;贾智平(1964-),男,教授,主研领域为分布式网络、嵌入式系统、实时系统。
收稿日期:2008-05-09 **修回日期:**2008-07-25

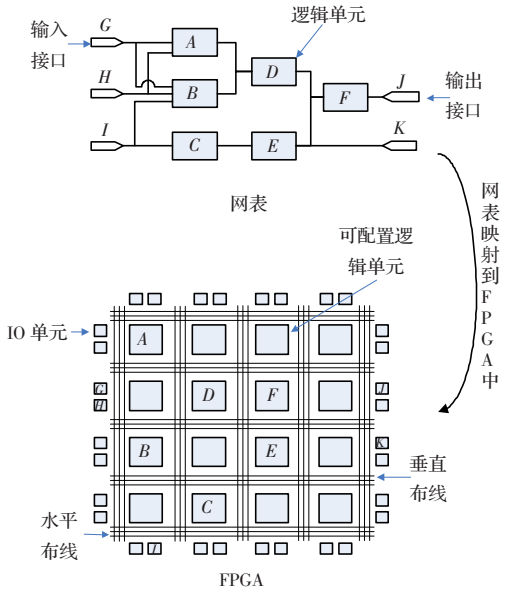

图 1　FPGA 的布局过程

就是说，对每个端口信号 $s_i \in S$，总有模型集 $M_{s_i}=\{m_j | s_i \in S_{m_j}\}$中。同时，给定一个位置集合 $L=\{l_1, l_2, \cdots, l_p\}$，其中 $p \geqslant |M|$。集合 L 对应于 FPGA 芯片的二维阵列，位置 $l_j \in L$ 在阵列中表示为下标为(x_j, y_j)对应的点。二维阵列的外围位置用于布置网表中的 IOBs，而 CLBs 则布局在阵列内部。FPGA 布局问题可描述为如何将模型 $m_i \in M$ 布置到一个 FPGA 二维阵列对应的唯一的位置 $l_j \in L$ 中，而使得电路的资源占用率和信号时延满足要求。

3　蚁群与粒子群混合的 FPGA 布局算法

3.1　适应度函数

布局问题的代价估计采用研究中常用的 BB-cost[1,9]，函数如下：

$$F=\sum_{i=1}^{n} q(i)\cdot\left[\frac{bb_x(i)}{C_{av,x}}+\frac{bb_y(i)}{C_{av,y}}\right] \tag{1}$$

其中，对每个网络(net)，$bb_x(i)$、$bb_y(i)$表示限制盒(Bounding-box)所占的水平和垂直长度；$C_{av,x}$、$C_{av,y}$表示限制盒在水平和垂直方向上的平均沟道容量；$q(i)$是补偿因子，用于补偿随着端口递增而导致网线的非线性增加。

3.2　交叉变异策略

蚁群算法利用正反馈原理和某种启发式算法的有机结合，容易出现早熟现象以及易陷入局部最优解，本文采用交叉变异策略对蚁群算法每次遍历得到的解进行优化。

当蚂蚁完成一次遍历后，设 j 只蚂蚁得到的解集 $C_0(j)$为$\{l_2, l_3, l_4, l_1, l_9, l_7, l_5, l_8, l_6\}$，参考解集 C_{best}为$\{l_5, l_4, l_8, l_1, l_2, l_3, l_7, l_9, l_6\}$。

则 $C_0(j)$与 C_{best}交叉策略为：在 C_{best}中随机选择一个交叉区域，将该交叉区域加入到 $C_0(j)$对应的位置，然后删除 $C_0(j)$中已在交叉区域出现的元素。如交叉区域为 l_1, l_2, l_3, l_7，交叉后，$C'_0(j)$为 $l_4, l_1, l_2, l_3, l_7, l_9, l_5, l_8, l_6$。

变异策略为：随机选择 $C_0(j)$中的第 i 个元素，在 $C_0(j)$中交换第 i 个元素和第 $i-1$ 个元素，其余不变，得到新的解集。例如，随机选择元素 l_1，变异后为：$l_2, l_3, l_1, l_4, l_9, l_7, l_5, l_8, l_6$。

3.3　PSAC 算法流程

首先把布局问题转化为蚁群网络，这里把布局过程看作 n 个阶段，然后每个阶段分配一个单元到 FPGA 中，具体步骤如下：

(1)$t=0$(t 为迭代步数)，初始化参数。

(2)将各蚂蚁的初始出发点置于禁忌表 $Tabu_k$ 中，对每只蚂蚁 $k(k=1,2,\cdots,n)$，按概率 P_{ij}^k 移至下一顶点 j，将顶点 j 置于 $Tabu_k$ 中。

$$P_{ij}^k=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}{\sum\limits_{l\in allowed_k}\tau_{il}^{\alpha}(t)\cdot\eta_{il}^{\beta}(t)} & j\in allowed_k\\ 0 & \text{otherwise}\end{cases} \tag{2}$$

其中 $allowed_k=\{0,1,\cdots,n-1\}-Tabu_k$。

(3)一次遍历完成后，每只蚂蚁根据当前解集计算适应值 $F(j)$，根据当前适应值与以前遍历的适应值比较得出个体极值 F_{pbest}，并记录个体极值对应的个体极值解集 C_{pbest}，根据各个粒子的个体极值 F_{pbest}，找出全局极值 F_{gbest}和全局极值对应的全局极值解集 C_{gbest}。

(4)对每只蚂蚁进行如下操作，第 j 只蚂蚁解集 $C_0(j)$与 C_{gbest}交叉得到 $C_1'(j)$，$C_1'(j)$与 C_{pbest}交叉得到 $C_1'(j)$，$C_1''(j)$以一定的概率变异到 $C_1(j)$，根据当前位置计算新的适应值 $F'(j)$，若新的目标函数变好，接受新值；否则就拒绝，第 j 个粒子解集仍 $C_1(j)$然为 $C_0(j)$，重新找出各只蚂蚁的个体极值 F_{pbest}和个体极值解集 C_{pbest}，找出全局极值 F_{gbest}和全局极值解集 C_{gbest}。

(5)按式(3)更新信息素。

$$\tau_{ij}(t+1)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t,t+1) \tag{3}$$

$$\Delta\tau_{ij}(t,t+1)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t,t+1) \tag{4}$$

$$\Delta\tau_{ij}^{k}(t,t+1)=\begin{cases}Q/C(j) & \text{第 } k \text{ 只经过路径}(i,j)\\ 0 & \text{否则}\end{cases} \tag{5}$$

(6)$t=t+1$。

(7)若 $t<$预定的迭代次数且无退化行为(即找到的都是相同的解)，则转(2)。

输出目前最好解。

4　实验结果及分析

利用目前流行的 FPGA 布局布线工具 VPR，对几种 MCNC 基准电路的实验结果与 SA、GASA 及 ACO 算法进行了对比。开发平台采用了 Intel Core2 双核处理器，操作系统采用了 Fedora7，采用 C 语言对 VPR 源程序中进行布局算法编程修改。表 1

表 1　全局布线所需轨迹数

基准电路	逻辑单元数	全局布线所需轨迹数				
		SA	GA	GASA	ACO	PSAC
9symml	70	5	5	5	5	5
alu2	143	6	6	6	6	6
apex7	77	5	5	5	5	5
e64	274	8	8	8	8	8
example2	120	5	5	5	5	5
k2	358	9	10	9	9	9
terml	54	5	5	5	5	5
too-large	148	7	7	7	7	7
vda	208	8	8	8	8	8
Total	1 452	58	59	58	58	58

(下转 114 页)