第 25 卷第 5 期　　半 导 体 学 报　　Vol. 25, No. 5
2004 年 5 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　May, 2004

一种使用遗传算法在高层次综合中完成互连优化的方法

王 磊　粟雅娟　魏少军

(清华大学微电子学研究所, 北京　100084)

摘要: 提出一种使用遗传算法在高层次综合中完成互连优化的方法. 相比同类的研究, 该方法的主要优势在于提出一种新颖的编码方法, 并设计了相应的遗传算子, 避免了在计算过程中不可行解的产生. 实验数据证明了该算法的有效性.

关键词: 深亚微米; 互连; 高层次综合; 资源分配; 遗传算法

EEACC: 1130B; 1265A

中图分类号: TN47　　**文献标识码**: A　　**文章编号**: 0253-4177(2004)05-0607-06

1 引言

高层次综合的任务是将算法的行为级描述映射成寄存器传输级的电路结构. 通过高层次综合, 设计者可以有效地搜索设计空间, 获得优化的电路设计. 在过去的十几年中, 高层次综合得到了广泛的研究[1,2], 同时, 研究机构也推出了很多高层次综合系统[3]. 另一方面, 主流工艺特征尺寸已经达到 0.18μm, 在不久的将来, 0.10μm 以下的生产线将趋于成熟并应用于集成电路的生产制造. 在新工艺条件下, 互连延时将大于逻辑功能单元的延时, 成为主导系统性能的关键因素[4], 前期设计中互连的优化成为非常重要的研究课题. 寄存器传输级电路中, 互连可以分成资源内部的互连和资源之间的互连, 影响系统性能的互连通常是运算资源和存储资源之间的互连, 高层次综合中资源分配确定了运算资源和存储资源之间的互连网络, 因此, 资源分配将极大影响电路的性能.

高层次综合中很多子步骤(包括资源分配)已经被证明是 NP 问题, 为了获得合理的算法时间复杂度, 人们普遍采用启发式的方法. 近年来, 出现了很多采取全局优化算法完成高层次综合的研究. 遗传算法(genetic algorithm)是其中一种有效的方法, 在电子设计自动化领域中获得了广泛的应用, 如器件综合[5]、布局布线等[6]. 本文提出一种使用遗传算法在高层次综合中优化互连的方法, 该方法主要的贡献在于提出了一种新颖的遗传编码方法, 并设计了相应的遗传算子, 避免了在进化过程中不可行解的产生. 实验数据证明了该算法的有效性.

2 相关研究

在许多研究工作中, 资源分配被分成运算资源分配、存储资源分配和互连网络分配三个子步骤. Rim 在他的研究中指出了同时完成多个分配任务的必要性[7], 如图 1 所示. 可以看出对同一数据流图的两种分配导致互连复杂度截然不同的结果, 而在功能单元分配和存储单元分配的过程中, 则无法区分两种分配策略的优劣. Rim 给出了一种统一完成各个资源分配任务的 ILP(整数线性规划)描述, 并使用一种启发式的算法求解. 启发式算法是一种局部

王　磊　男, 1976 年出生, 博士研究生, 研究领域是深亚微米集成电路设计方法学和数字系统的高层次综合方法.

粟雅娟　女, 1975 年出生, 博士研究生, 研究领域是系统级设计方法和 SOC 低功耗设计.

魏少军　男, 1958 年出生, 教授, 博士生导师, 研究领域是电子设计自动化方法和通信专用集成电路设计.

2003-05-07 收到, 2003-07-14 定稿

优化的方法,很难获得全局优解.

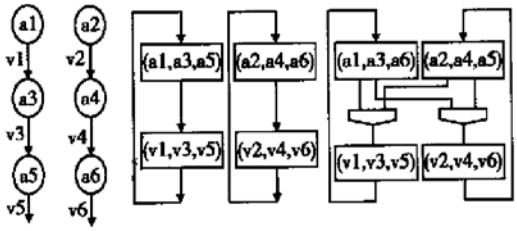

图 1 文献[7]中的例子

Fig. 1 Example in Ref. [7]

遗传算法是一种统计方法,它模拟自然界进化的规律,完成解空间的搜索.遗传算法与具体的问题关联性很小,而且是一种全局优化方法,因而使用遗传算法实现高层次综合近年来成为研究的热点,遗传算法的基本结构如下所示[8]:

```
Genetic Algorithm
    t< = 0;
    initialize P(t);
    evaluate P(t);
    while (Not termination-condition) do{
        t< = t+ 1;
        select P(t) from P(t- 1);
        alter P(t);
        evaluate P(t)
    }
```

文献[9]提出了一种"问题空间"(problem space)的遗传算法,原始的综合任务被变换成一种问题空间的遗传算法,在问题空间内求解之后,变换为原始的综合任务的解.文献[10]介绍了一种相对编码方法,不同于直接对控制步的编码,针对算子相对顺序进行编码,减小了每个染色体编码所需要的位数.文献[11]采取统一的调度和分配的编码,通过一种修补策略,对不可行解进行修正,使其成为可行解,从而实现全局寻优.

将遗传算法应用到高层次综合中,需要解决的一个重要问题就是避免在进化的过程中产生不可行解.例如,调度不能违反数据的相关约束,而相同控制步的算子不能分配到同一个硬件资源.所有这些,表现为施加于遗传算法的约束.上述讨论的修补策略和问题空间的映射虽然避免了产生不可行解,但是不能保证遗传过程中解的多样性,因为不同的不可行解有可能被变换或修正成同一可行解,这种现象在可行解稀疏分布于解空间时尤其显著.同时,修补策略将消耗大量的 CPU 时间,降低算法的效率.

3 算法实现

我们选择 SDFG(scheduled data flow graph)作为算法的输入,寄存器传输级电路结构作为输出.

$G(V,E)$ 表示了 SDFG,V 是算子的集合,E 是数据传输集合.对于 V 中每一个元素 v,有两个重要的属性:type(类型)和 active control step(激活控制步),激活控制步表示的是执行算子的第一个时钟周期,数据的传递发生在目标算子的激活控制步.如果 $e_1=\{s_1,u_1\}$,$e_2=(s_2,u_2)$ 满足:

$$\begin{cases} s_1.\text{type}=s_2.\text{type} \\ u_1.\text{type}=u_2.\text{type} \\ u_1.\text{active control step}\neq u_2.\text{active control step} \end{cases}$$

那么,分配 s_1 和 s_2 到同一资源的同时,分配 u_1 和 u_2 到同一资源将有利于优化互连,因为两次数据传输可以分时复用物理连线.我们的分配目标就是获得实现原始 SDFG 的连线复杂度最小的寄存器传输级结构.

输入:SDFG $G(V,E)$ 属性 type 和 active control step 已初始化.

输出:寄存器传输级数据通道,即分配 V 中每一个元素到相应的硬件资源.

优化目标:最优的连线结构

$$\text{Minimize:}\quad \sum_{i=1}^{M}\sum_{j=1}^{M} x_{i,j}$$

其中

$$\begin{cases} x_{i,j}=1, & \text{Res}_i\times\text{Res}_j\cap E\neq\phi \\ x_{i,j}=0, & \text{otherwise} \end{cases}$$

M 表示所有资源的个数,Res_i 表示分配到资源 i 的算子集合.$x_{i,j}$ 表示资源 i 和资源 j 之间是否存在物理互连.

编码方法是遗传算法中的一个重要课题.不适当的编码将造成在使用遗传算子(杂交、变异等)之后,产生大量的不可行解.我们使用图 5 所示的 SDFG 为例来说明我们的方法.假设实现此 SDFG 的寄存器传输级数据通道包括加法器、乘法器和寄存器,相应的资源数目分别是{2, 1, 4}.

染色体由一系列的基因构成,每一个基因表示了 SDFG 中的一个算子.我们将这些算子根据它们类型属性分成若干个集合,每个集合中的算子按照

他们的激活控制步递增排序, 对于激活控制步相同的算子, 我们按照生命周期递减的排序. 生命周期定义为执行算子所需要的时钟周期数, 生命周期等于 1 的算子称为单周期算子, 大于 1 的算子称为多周期算子. 图 2 表示了示例数据流图中各算子的激活控制步和生命周期.

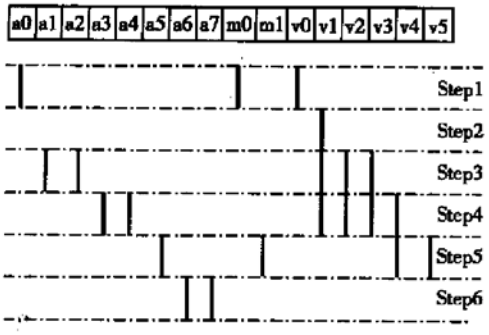

图 2　示例数据流图中各算子的激活控制步和生命周期

Fig. 2　Active steps and life spans for operators in Fig. 5

根据上述的构造方法, 一个染色体包含了 N 个基因集合, 每个基因集合又由一系列类型属性相同的有序算子组成. N 表示了实现相应 SDFG 的寄存器传输级结构中资源类型的数目. 例如, 表示图 5 数据流图的染色体如图 3 所示. 上述编码方法的优点在于不同的基因集合之间是无关联的, 进化过程中对父代解的变换在各个基因集合中完成, 能够有效地避免不可行解的产生. 同时, 编码方法可以表示可行解空间的任何一个实例, 是一种完备的编码方法. 由于不同控制步的算子可以复用相同的硬件资源, 初始化过程中的每一步随机产生一个控制步的分配结果, 对于多周期算子, 算子跨越多个周期, 算子的分配在其激活控制步完成, 由于后续的连续时钟周期中该算子依旧占用已分配的资源, 因此分配其他算子时将不使用被占用的资源.

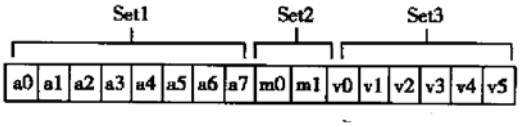

图 3　表示图 5 数据流图的染色体

Fig. 3　Chromosome presentation for Fig. 5

杂交过程组合两个父染色体的特性, 形成两个新的染色体. 应用杂交算子的目的是实现不同解之间的信息交换. 我们随机选择控制步 rstep, 对于每一个基因集合, 计算出杂交点, 使得杂交点之前的所有算子的激活控制步小于 rstep. 由于我们已经对所有的基因集合进行排序, 杂交点一定是存在的. 子代解 c_1 继承了父代解 p_1 杂交点之前的所有基因, 而在杂交点之后的基因, 尽量保持与父代解 p_2 相匹配. 所使用的方法将在下文中解释. 子代解 c_2 采用类似的方法构造. 对于图 5 的示例, 我们随机地选择控制步 4, 计算出各个基因集合的杂交点, 图 4 中阴影部分表示了杂交点之前的所有基因.

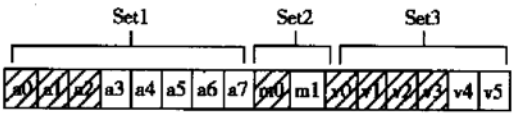

图 4　控制步为 4 的杂交点　阴影部分表示了杂交点之前的基因.

Fig. 4　Illustration for crossover point　Shaded operators′ active steps are earlier than crossover point when control step equals 4

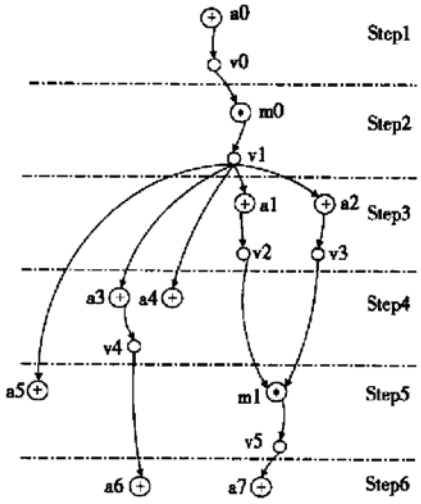

图 5　一个 SDFG 的例子

Fig. 5　An SDFG example

对于单周期算子, 子代解 c_1 可以直接继承父代解 p_2 杂交点之后的基因, 这是因为在不同的控制步中的算子可以使用相同的资源, 不会引起任何的资源冲突. 而对于多周期算子来说, 这样做将引起资源的冲突. 例如图 5, 我们假设父代解 p_1 中, v3 分配到寄存器 register0, v4 分配到寄存器 register1, 在父代解 p_2 中, v3 分配到寄存器 register1, v4 分配到寄存器 register0. 如果子代解 c_1 继承了 p_1 中 v3 的分

配结果, p_2 中 v4 的分配结果, 则 v3 和 v4 将均被分配到寄存器 register0, 而这两个变量在控制步 4 中同时有效, 很显然, c_1 成为一个不可行解. 下文的伪码表示中 Crossover 是我们设计的杂交算子. Keep-Compatible 是子代解中激活控制步大于等于参数 rstep 的基因保持与父代解匹配的算法. 完成杂交过程之后, 所产生的两个子代解均为可行解.

变异过程按照变异概率随机改变染色体的一个或者多个基因进行变异. 应用变异算子的目的是提高子代的多样性. 我们随机选择一个基因进行变异, 为了使新的染色体仍然表示一个可行解, 变异算子如 Mutation 所示, 根据变异基因所对应的控制步, 我们计算出变异基因所属的基因集合的变异点, 与杂交算子类似, 该基因集合中变异点之后的基因尽可能的保持与父代解相同.

```
Crossover (p1, p2, rstep, c1, c2)
Input p1, p2, rstep;
Output: c1, c2;
  Compute crossover point cpi based on rstep for
  every gene sets;
  For every gene set i: {
    c1[i: 1-> cpi] < = p1[i: 1-> cpi];
    c2[i: 1-> cpi] < = p2[i: 1-> cpi];
    if i corresponds to uni-cycle resource{
        c1[i: cpi+ 1-> end] < = p2[i: cpi+ 1->
        end];
        c2[i: cpi+ 1-> end] < = p1[i: cpi+ 1->
        end];
    }
    if i corresponds to multi-cycle resource{
        KeepCompatible (c1, p2, rstep, i);
        KeepCompatible (c2, p1, rstep, i);
    }
  }

Mutation(c, p, t);
Input p, t; (t present the gene for mutation)
Output c;
  s presents gene set which includes gene ready for
  mutation;
  rstep < = gene t's active control step;
  Computer mutation point mp for gene set s based
  on rstep;
  For every gene set i except s: {
    c[i: 1-> end] < = p[i: 1-> end];
  }
  c[s: 1-> mp] < = p[s: 1-> mp];
  Construct a feasible allocation results for genes
  whose active control step equals rstep;
  KeepCompatible (c, p, rstep+ 1, s);

KeepCompatible (c, p, rstep, i):
Input c, p, rstep, i;
Output c;
  For s= rstep to total control steps: {
    For every t in index sets of genes whose active
    step equals s: {
        c[i: t] < := p[i: t] if no confliction,
        otherwise add t to unprocessed list;
    }
    For every t in unprocessed list: {
        c[i: t] < = random select a free resource;
    }
```

4 实验结果

4.1 设计示例

对于图 5 中的例子, 我们使用上述算法, 得到图 6 所示的寄存器传输级数据通道. 分配到同一功能单元的算子在大部分情况下读/写相同的寄存器, 这点由存储单元的分配(变量到寄存器的分配)保证. 图 6 表示的是一种优化的结构.

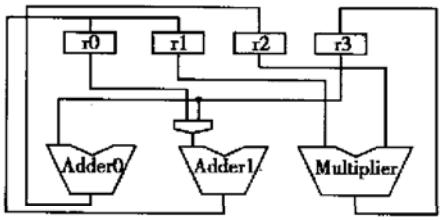

图 6 实现图 5 数据流图的的数据通道

Fig. 6 Synthesized datapath for Fig. 5

4.2 基准电路的实验结果

为了与文献[12]中的方法比较, 我们选择了 EWF 和 DCT 作为例子, 并采用了文献[12]中的硬

件配置(功能单元数目、寄存数目). EWF 和 DCT 是两种被广泛采用的基准电路,用来比较和评价综合算法. 表 1 列出了实验的结果,在所有的情形下我们的算法优于文献[12]的算法.

表 1　实验数据

Table 1　Experiment results

	Resource			Wire count	
	+	×	Reg	Ours	Ref. [12]
EWF	2	1	11	23	28
	2	2	11	25	31
	3	3	11	31	37
DCT	2	2	15	50	53
	3	3	15	52	54
	4	4	15	55	60

4.3　参数取值对实验结果的影响

表 2 列出了上述实验中所使用的参数. 遗传算法中参数的选择直接影响实验的结果. 通过实验发现,算法对杂交概率(crossover possibility)和变异概率(mutation possibility)不敏感,而一代中个体的数目(population)以及进化代数目(generation)对结果产生较大的影响. 以 EWF 的一种硬件配置为例,我们讨论这两个参数对实验结果的影响. 表 3 列出了 EWF 对应 SDFG 的信息以及示例的硬件资源配置.

表 2　遗传算法的参数

Table 2　Parameters in experiments

Population	200
Generation	150
Crossover possibility	0.8
Mutation possibility	0.2

表 3　EWF 实例的 SDFG 统计和硬件配置

Table 3　SDFG statistic data and resource configuration in exploring relations between parameters and results

SDFG 统计		硬件配置	
Add	26	Adder	2
Multiply	8	Multiplier	1
Variable	30	Register	11

实验中我们取 population 和 generation 从 50 到 250. 由于遗传算法是一种基于统计的方法,为了消除随机性的影响,每对参数均进行 10 组实验取平均值. 图 7 的 x 和 y 坐标分别是 population 和 generation, z 坐标为目标函数. 可以看出,增加 population 和 generation 有助于提高算法获得最优解的能力,代价是需要更长的计算时间. 实际应用中,通常根据问题的规模选取合理的参数.

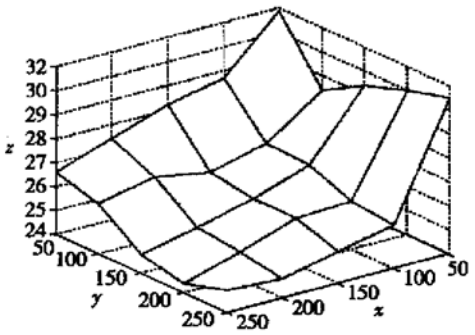

图 7　参数对实验结果的影响

Fig. 7　Experiment results in different parameters

5　结论

本文讨论了在高层次综合中同时完成各个资源分配任务的重要性,并提出一种使用遗传算法在高层次综合中优化互连的方法,相比同类研究,本文的主要贡献在于提出一种新颖的编码方法并设计了相应的遗传算子,避免了计算过程中不可行解的产生. 实验数据证明了该算法的有效性.

参考文献

[1] Gajski D. High-level synthesis: introduction to chip and system design. Kluwer Academic, 1992

[2] Lin Y. Recent developments in high-level synthesis. ACM Trans Design Automation of Electronic Systems, 1997, 2(1): 2

[3] Walker R A, Camposano R. A survey of high level synthesis systems. Kluwer Academic, 1991

[4] Cong J, Pan D Z. Interconnect estimation and planning for deep submicron designs. IEEE Design Automation Conference, 1999: 507

[5] Xie Xiaofeng, Li Zhao, Ruan Jun, et al. Realization of device synthesis using genetic algorithms. Chinese Journal of Semiconductors, 2002, 23: 95(in Chinese) [谢小锋, 李钊, 阮骏, 等. 应用遗传算法实现 MOS 器件综合. 半导体学报, 2002, 23: 95]

[6] Wang Xiaogang, Yao Linsheng, Gan Junren. VLSI floorplanning method based on genetic algorithms. Chinese Journal of Semiconductors, 2002, 23: 330(in Chinese) [王小港, 姚林声, 甘骏人. 基于遗传算法的 VLSI 布局规划方法. 半导体学报, 2002, 23: 330]

[7] Rim M, Mujumda A. Optimal and heuristic algorithms for

solving the binding problem. IEEE Trans VLSI Syst, 1994, 2 (2): 211

[8] Michalewicz Z. Genetic algorithms+ data structures= evolution programs. Springer Verlag, 1994

[9] Dhodhi M, Hielscher F. Datapath synthesis using a problem space gnetic algorithm. IEEE Trans ICCAD, 1995, 14: 934

[10] Torbey E, Knight J. High-level synthesis of digital circuits using genetic algorithms. Proc of the IEEE Conference on Evolutionary Computation, 1998

[11] Ohmori K. High-level synthesis using a genetic algorithm. Electronics & Communications in Japan, 2000

[12] Choi K, Levitan S P. A flexible datapath allocation method for architectural synthesis. ACM Trans Design Automation of Electronic Systems, 1999

Unified Resource Binding for Interconnect Reduction with Genetic Algorithm

Wang Lei, Su Yajuan and Wei Shaojun

(*Institute of Microelectronics, Tsinghua University, Beijing* 100084, *China*)

Abstract: A unified resource binding is proposed for interconnect reduction with genetic algorithm. The main contributions are that a novel coding method and corresponding genetic operators are put forward to avoid the generation of infeasible solutions. Experimental results show the efficiency of the algorithm.

Key words: deep submicron; interconnect; high level synthesis; resource binding; genetic algorithm

EEACC: 1130B; 1265A

Article ID: 0253-4177(2004)05-0607-06

Wang Lei male, was born in 1976, PhD candidate. His research interests are high level synthesis for digital systems.

Su Yajuan female, was born in 1975, PhD candidate. Her research interests are system level low power design.

Wei Shaojun male, was born in 1958, professor. His research interests are EDA methodology and communication ASIC design.

Received 7 May 2003, revised manuscript received 14 July 2003