集成电路的模块生成与选择算法

郎荣玲, 戴冠中
(西北工业大学, 陕西西安 710072)

摘 要: 借鉴软件设计中的思想, 采用模块化技术是提高大规模集成电路的设计能力和系统芯片开发效率的重要手段. 文章首先对现有的模块生成算法进行了全面的分析, 在此基础上提出了一新的模块生成算法, 此算法可生成一个电路系统的顶点数小于 m 的所有模块, 并且对电路系统以及模块的结构没有限制. 本文还提出了一个模块选择算法, 此算法可以在满足一定要求的前提下选择一部分模块覆盖整个电路, 同时还对算法进行了实验分析.

关键词: 模块; 模块化技术; 规则性; 控制数据流图
中图分类号: TP302 文献标识码: A 文章编号: 0372-2112 (2005) 11-1955-04

Algorithms for Template Generation and Selection in Integrated Circuits

LANG Rong-ling, DAI Guan-zhong
(*Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China*)

Abstract: Modularity is an important procedure for improving the design methodology and the design ability of VLSI. Modularity has the merits for predigesting design, shortening the period of design, reducing the design cost, reusing the design, partitioning software and hardware, and so on. The existing algorithms for settling the problem of template generation are analyzed. A regularity extraction algorithm is presented, which can generate all the templates of a circuit. The algorithm is independent on the structure of circuits and modules. An algorithm is also put forward which can select part of the templates to cover the circuit under some situations, and the experiment results of the algorithms are also given.

Key words: template; modularity; regularity; control data flow graph

1 引言

模块化技术是现在软硬件设计的主导思想. 将一个系统的设计划分成一系列已定义的模块有助于进行协同设计, 从而减轻设计难度、提高设计效率. 模块库中的每个模块都得到了功能上的验证, 可以在模块建立时检查其属性的正确性, 并且提供了在设计初期中的面积、速度、功耗等更精确的估计, 因而使整个系统的设计更加简单. 库单元的建立不仅继承了以前设计工作的成果, 而且建立了将来电路设计的基础, 有利于设计再利用, 减少当前或后续项目中开发的精力. 近年来, 对 VLSI 设计中的模块化问题的研究已经变得越来越重要, 对其深入研究有着重要的意义和价值.

模块化技术包含模块的生成、模块的选择、模块的调度与分配、模块布局、布线等一系列重要问题, 本文主要研究了模块的生成和选择问题.

2 研究现状

目前已有一些算法可以生成电路的模块. 在文献[1~3]中, 作者提出了一些模块生成算法, 这些算法均是先选择某一顶点作为一个模块, 然后在此模块内不断加入其余的顶点形成新的模块. 这些算法对模块的形式没有限制, 但是此方法最大的缺点是所生成的模块形式依赖于起始模块的选择. 文献[4]中提出了通过收缩出现频数最多的边的方法生成模块. 文献中提出的模块生成算法的局限性在于生成的模块与选择哪条边收缩有关. 文献[5]中的算法可以生成树形并且是单输出的模块, 文献[6]的算法可以生成对输出没有限制, 但要求单输入的模块. 但是在实际的电路中往往也需用到多输入以及多输出的模块, 因此文献[5, 6]中的算法不满足实际的要求. 文献[7, 8]中提出了一模块生成算法, 生成了一个系统所需要的所有模块, 此算法利用三个条件来判断一个顶点 u 是否能和 i 个顶点的子图形成 $i+1$ 个顶点的子图: 第一个顶点的序号最小; 在新加入的顶点与原来的子图构成的新的子图中, 新加入顶点距第一个顶点的距离不小于子图中其余顶点距第一个顶点的距离; 距第一个顶点的距离相等的顶点中, 序号小的顶点排在前面. 因为每增加一个顶点后, 图中各个顶点之间的距离就有可能改变, 因此在判断第二个条件时需要多次计算同一对顶点之间的距离, 增加了算法的复杂性, 特别是当处理有圈图时, 此问题变得更为复杂.

本文提出的模块生成算法可以生成任意给定电路的顶点数小于某个整数 m 的所有模块, 并且利用三个限定条件保证了相同顶点的子图不会重复出现, 降低了在对模块分类时问题的复杂性. 算法在实现过程中利用有序顶点集存储模块, 不仅保证了模块与顶点集一一对应, 而且节省了存储空间.

收稿日期: 2005-01-14; 修回日期: 2005-06-16

3 模型建立

设 C 表示一个电路的 CDFG, $G'(V, E)$ 为此电路的有向图表示, 其中 $V = P_G \cup N_G$, N_G 表示操作与控制顶点集合, $P_G = I_G \cup O_G$, 其中 I_G 和 O_G 分别为输入端和输出端的顶点集合, G 为 G' 的以 N_G 为顶点集的导出子图.

定义 **1** $G'(V, E)$ 为某电路 C 的有向图表示, 其中 $V = P_G \cup N_G$, 若电路 $S(P_S \cup N_S, E_S)$ 满足下列条件:

(1) $N_S \subset N_G$, 且以 N_S 为顶点集的 S 的导出子图与以 N_S 为顶点集的 G' 的导出子图相同;

(2) 对于 $\forall u \in I_S$, 在 V 中必存在一顶点 v 满足 $v \notin N_S$, 并且有信号从顶点 v 输出, 作为 N_S 中某个顶点的输入;

(3) 对于 $\forall w \in O_S$, 在 V 中必存在一顶点 t 满足 $t \notin N_S$, 并且有信号从 N_S 中的某个顶点的输出, 顶点 t 为此信号的输入; 则称 S 为 C 的子电路.

定义 **2** 设 C 表示一个电路, C 的模块是指 C 中的子电路, 与此模块结构相同的子电路均称为此模块的实例.

定义 **3** 设 $G(V, E)$ 为某电路 C 的有向图表示, 其中 $V = I_G \cup O_G \cup N_G$, I_G、O_G 分别表示电路的输入、输出顶点集合, 若集合 $C(G) = \{C_1, C_2, \cdots, C_n\}$ 满足:

(1) $C_i = (V_i, E_i)$, $V_i = I_i \cup O_i \cup N_i$, $i = 1, 2, \cdots, N$ 为电路 C 的子电路.

(2) $V \subset V_1 \cup V_2 \cup \ldots \cup V_n$.

(3) 任何一个子电路 $C_i = (V_i, E_i)$ 的输入一定是某个子电路的输出或电路 C 的输入, 即若 $v \in I_i$, 则 $v \in O_1 \cup O_2 \cup \ldots \cup O_n \cup I_G$.

(4) 任何一个子电路 $C_i = (V_i, E_i)$ 的输出一定是某个子电路的输入或电路 C 的输出, 即若 $v \in O_i$, 则 $v \in I_1 \cup I_2 \cup \ldots \cup I_n \cup O_G$.

则称 $C(G) = \{C_1, C_2, \cdots, C_n\}$ 为电路 C 的一个覆盖.

模块生成问题就是指给定电路 C, 生成此电路的一系列模块 $T = \{T_1, T_2, \cdots, T_l\}$, 以及每类模块的所有实例. 模块化技术的最终目的是降低设计成本、缩短设计周期、减小芯片的面积、减少延迟、降低功耗等. 因此在设计过程中并不是模块库中的所有模块都可以用于构造整个系统, 需要在这些模块中选择一些能够优化上述指标的模块. 模块选择问题就是指给定电路 C 以及模块集合 $T = \{T_1, T_2, \cdots, T_l\}$ 和每类模块的实例, 在 T 中找到电路的一个覆盖 $C(G) = \{T_{i_1}, T_{i_2}, \cdots, T_{i_n}\} \subseteq T$, 使得目标函数 $f(C(G)) = \sum_{j=1}^{n} g(T_{i_j})$ 取得最大值, 其中 $f(C(G))$ 为覆盖 $C(G)$ 的函数, $g(T_{i_j})$ 衡量模块 T_{i_j} 对系统的贡献的函数.

4 模块生成算法

算法的思路是首先给每个顶点一个唯一的序号, 然后在 i 个顶点的子图的基础上加入一个顶点生成 $i+1$ 个顶点的子图, 进而生成含有 $i+1$ 个顶点的模块. 某个 $i+1$ 个顶点的子图可以在不同的 i 个顶点的子图的基础上生成, 因此如果不加任何限制, 会有许多完全一样的子图重复出现, 相同的 $i+1$ 个顶点的子图最多可能会重复出现 $(i+1)!$ 次, 造成对 CDFG 分类时, 需要判断是否同构的图的数量大量增加, 算法的复杂性提高. 因此需要加入限定条件使得相同 $i+1$ 个顶点的子图只出现一次. 可以利用有序顶点集存储导出子图, 使得相同的 $i+1$ 个顶点的排序是唯一的, 从而保证相同顶点的子图不会重复出现, 减少了需要判断是否同构的子图的数量.

定义 **4** 设 G 为有向图, $S \subset G$ 为图 G 的某导出子图, 将 S 的顶点集记为有序的顶点集, 此顶点集的第一个顶点称为此顶点集或此子图的起点.

定义 **5** 设 G 为有向图, G 的每个顶点有一个唯一的序号, $\tilde{G}$ 为 G 的基础图, D 为实数矩阵, 矩阵元素

$$d_{ij} = \begin{cases} 0 & , \ i \leqslant j \\ d_{\tilde{G}}(v_i, v_j) & , \ i > j \end{cases}$$

其中 $d_{\tilde{G}}(v_i, v_j)$ 为顶点 v_i 和 v_j 在图 $\tilde{G}$ 中的距离, i, j 分别为顶点 v_i 和 v_j 的序号, 称矩阵 D 为图 G 的有序距离矩阵.

由定义 5 可以看出, 有向图的有序距离矩阵为三角形矩阵. 目前已有一些算法可以建立距离矩阵, 例如利用著名的 Dijkstra 求最短路算法[9], 可以求得 $\tilde{G}$ 中任意两点之间的距离, 从而可以建立有序距离矩阵. 若 $\tilde{G}$ 中有 n 个顶点, 建立有序距离矩阵的计算复杂度为 $O(n^3)$.

定理 **1** 设 G 为无向图 S 为 G 的 n 个顶点的连通的导出子图, 将 S 的顶点按下述准则排序,

(1) 序号最小的顶点为起点;

(2) 设起点为顶点 u, 若顶点 v 和 w 满足 $d_G(v, u) > d_G(w, u)$, 则在排序中顶点 v 位于顶点 w 之后;

(3) 若 $d_G(v, u) = d_G(w, u)$, 并且顶点 v 的序号大于顶点 w 的序号, 则在排序中顶点 v 位于顶点 w 之后;

则 S 中顶点的顺序是唯一的.

证明: 现利用反证法证明定理的结论成立. 设 S 为 G 的连通的导出子图, 假设按上述 3 个条件将顶点排序后, 得到两个不同的有序序列, 设这两个序列分别为 I_1 和 I_2.

根据排序的准则, I_1 和 I_2 中的起点相同. 因为在 I_1 和 I_2 中顶点的排序不同, 因此至少存在一对顶点 v_i 与 v_j 在 I_1 和 I_2 中的先后顺序不同. 下面分两种情况讨论 v_i 与 v_j:

(1) v_i 与 v_j 距起点的距离不同. 不妨设 v_i 距起点的距离大于 v_j 距起点的距离, 根据排序准则无论在 I_1 还是 I_2 中, 顶点 v_i 均位于顶点 v_j 之后.

(2) v_i 与 v_j 距起点的距离相同. 不妨设 v_i 的序号大于 v_j 的序号, 根据排序准则无论在 I_1 还是 I_2 中, 顶点 v_i 均位于顶点 v_j 之后.

因此无论怎样 v_i 与 v_j 在 I_1 和 I_2 中的先后顺序都会相同, 因此假设不成立, 所以定理的结论成立.

证毕.

算法 1 给出了生成所有顶点数小于 m 的模块的算法. 设 T_j 表示含有 j $(j = 1, 2, \cdots, m)$ 个顶点的所有模块的集合, $S_{i,j}$, $i = 1, 2, \cdots, n, j = 1, 2, \cdots, m$ 表示 G 中以顶点 v_i 为起点的 j 个顶点的所有连通的导出子图的集合, 其中 n 为 G 的顶点数. 从 step6~step12 为在 j 个顶点的子图的基础上生成 $j+1$ 个顶

点的子图的过程, 在生成过程中首先利用函数 1(*a*) 验证所选顶点是否满足前面提出的 3 个条件. Step10 step11 为生成子图所对应的模块且将模块分类的过程. 在一个子图的基础上不一定只能生成一个 $j+1$ 个顶点的子图, step12 保证了对子图的顶点邻域内的所有顶点进行搜索. 因为当某子图起点的序号大于 $n-j$ 时, 不可能在此子图的基础上生成 $j+1$ 个顶点的子图, 因此在 step13 将子图起点的序号与 $n-j$ 进行了比较.

算法 **1**　生成所有顶点数小于 ***m*** 的模块

输入　图 G, 整数 m,

输出　模块以及每类模块的实例和频数.

Step1　给 G 的每个顶点一个唯一的序号.

Step2　生成所有一个顶点的模块, $j=1$;

Step3　$i=1$, $T_{j+1}=\Phi$;

Step4　$S_{i,j+1}=\Phi$;

Step5　取 $S\in S_{i,j}$;

Step6　$v\in N_S$, 设顶点 v 的序号为 $S.N(v)$;

Step7　调用函数 1(*a*), 若 *condition*= 1, 则执行 step8; 否则执行 step12;

Step8　S 与顶点 v 构成一个新的导出子图 S', S' 的顶点集为 S 的顶点集与 v, 并且 S 中顶点的顺序不变, 将顶点 v 作为 S' 的顶点集的最后一个顶点;

Step9　$S_{i,j+1}=S_{i,j+1}\cup\{S'\}$;

Step10　按定义 1 的方法生成 S' 所对应的模块, 记为 C.

Step11　判断在 T_{j+1} 是否存在与 C 同构的模块, 若存在, 此类模块的频数加 1; 若不存在, 增加一类新的模块, 即 $T_{j+1}=T_{j+1}\cup C$;

Step12:　$N_S=N_S-\{v\}$, 若 $N_S=\Phi$, 执行 step13; 否则执行 step6;

Step13:　$i=i+1$, 若 $i\leqslant n-j$ 执行 step4; 否则执行 step14;

Step14:　$j=j+1$, 若 $j\leqslant m$ 执行 step3; 否则结束.

结束

函数　**1(*a*)**　验证顶点是否满足条件

输入　顶点 v, 子图 S

输出 *condition* 的值

Step1　若 $S.N(v)>i$, 执行 step2; 若否 *condition*= 0;

Step2　在有序矩阵 D 中, 若 $d(v,v_i)\geqslant d(w,v_i)$, 执行 step3; 若否 *condition*= 0; 其中 w 为 S 内任意一个顶点.

Step3　若 $d(v,v_i)=d(w,w_i)$, 执行 step4; 若否 *condition*= 1;

Step4: 若 $S.N(v)>S.N(w)$, *condition*= 1; 否则 *condition*= 0.

结束.

在算法 1 中考虑了每个子图的顶点邻域中的所有顶点, 因此算法 1 可以生成所有顶点数小于 k 的连通子图, 进而生成所有顶点数小于 k 的连通模块. 由算法的 step11 可知, 不同模块的结构不同, 并且根据定理 1 可知相同顶点构成的模块不会重复出现. 因此给定一个电路和一个整数 k, 算法 1 可以生成所有顶点数小于 k 的模块, 并且不同模块的结构不同, 相同顶点构成的模块不会重复出现.

5　模块选择算法

一般情况下, 在设计过程中所有的优化指标不可能同时达到最优[10~13], 根据选择的模块不同, 造成的优化结果的侧重点可能不同, 因此在进行模块选择时要在这些需优化的指标之间进行协调. 为了节省设计成本以及后期的布局、调度工作的顺利进行, 在进行模块选择时, 应尽量要求覆盖 $C(G)$ 中模块的种类和总的数量越少越好, 因此应尽量使得模块不相交. 因此可以仿照文献[4, 7, 8]中的方法, 利用最大独立集与最小覆盖的关系选择模块.

定义 **4**　模块关系图 $G_T(V_T, E_T)$ 是一无向图, 模块库中的每一个模块对应于顶点集 V_T 中的一个顶点, 若模块 T_1、T_2 有公共的顶点, 则它们对应的 V_T 中的顶点 v_1、v_2 之间有边. 并且将模块关系图中的顶点根据模块的类型分为一些子集, 每个子集在模块关系图中表示为复合顶点.

算法 **2**　模块选择算法

输入　电路, 以及模块库 T.

输出　所选取的模块集合 C.

Step1　建立关于模块集合 T 的模块关系图.

Step2　找出每个复合顶点的最大独立集.

Step3　计算每类模块的对系统的贡献函数值 $g(T_i)$, 选择具有最大函数值的模块以及此模块所对应的最大独立集加入 C.

Step4　在模块关系图中去掉 C 中以及与 C 中顶点相邻的顶点.

Step5　判断 C 中的模块是否满足覆盖了整个电路, 若是结束, 若否执行 step3.

结束.

6　实验分析

首先利用算法 1 生成了一些电路的顶点数≤2 的所有模块, 表 1 列出了相应的结果. 从表 1 可以看出, 模块的种类比模块的数量要小的多, 并且一个电路的规则性越强, 此电路包含的模块的种类越少. 表 1 还可以看出虽然 4 点快速傅里叶变换、8 点快速傅里叶变换、16 点快速傅里叶变换的模块的数量不同, 但是它们的模块的种类几乎是一样的. 这说明虽然有些电路的基本运算功能存在差别, 但是它们的 CDFG 的结构相似.

然后利用算法 2 在上述模块中进行选择, 选择一部分模块覆盖电路. 在此试验中定义 $g(T_i)=g(w,s,i,o)=\frac{w^{2.2}s^2}{i+o}$, 其中 T_i 表示一个模块, w 为模块所覆盖的顶点数, s 为该模块的最大独立集中模块的数量, i 表示模块的输入端的数目, o 表示模块的输出端的数目. 此函数综合考虑了模块的输入、输出结构、频数、以及所覆盖的顶点数, 其中 $w^{1.2}s$ 可以保证所用模块的数量以及模块的种类尽量少, $\frac{ws}{i+o}$ 保证在输入、输出结构满足一定限定要求的前提下, 优先选取覆盖顶点数最多的模块.

模块选择的结果如表 1 所示, 由表 1 可以看出, 虽然所生成的模块的数量很大, 但是利用少量模块就可以覆盖整个电路, 因此进行合理的选取后必然可以达到一定的优化效果.

表 **1** 生成和选择模块的结果

模块 / CDFG	顶点数≤2 的所有模块		所选模块	
	模块数量	模块种类	模块数量	模块种类
7 阶 FIR 滤波器	29	4	8	1
4 点快速傅里叶变换	88	16	24	9
8 点快速傅里叶变换	266	17	72	9
16 点快速傅里叶变换	800	17	218	9
4 层波形椭圆滤波器	62	20	19	15
离散余弦变换	89	23	26	16

在这个试验中利用算法 1 分析了功能不同电路中是否存在结构相同的子电路的问题. 首先利用算法 1 生成 FIR 滤波器、离散余弦变换、4 阶波形椭圆滤波器、快速傅里叶变换的所有 2 个顶点的模块, 并将模块按其在上述所有的 CDFG 中出现的频数排序, 然后计算排在前 13 位的模块覆盖 CDFG 的顶点数的百分比, 得到图 1. 在图 1 中, 纵坐标表示序号小于等于的所有模块覆盖整个 CDFG 的顶点数的百分比. 从图 1 可以看出当横坐标 5 时, 图中每条曲线上升的速度都比较快, 说明不同的电路的 CDFG 中不仅存在着结构相同的子电路, 而且这样的子电路的数目也是非常多的, 因此利用算法 1 建立的模块库对于设计重用有重要的意义.

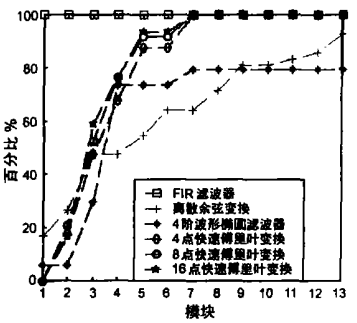

图 1 利用前 13 位模块覆盖 CDFG 的结果

7 结束语

文章从几个方面总结了模块化技术在 VLSI 自动化设计领域的重要作用, 并且全面介绍了现有文献中的模块生成算法, 深入的分析了已有算法的优点和局限性. 提出了一个新的模块生成算法, 此算法可以生成任意给定的电路系统的所有模块. 提出了一个模块选择算法, 此算法可以在满足一定限制条件的前提下, 在给定的模块库中选择出一部分模块覆盖整个电路, 从而达到一定的优化目的. 同时通过实验, 不仅验证了本文算法的实际可应用性, 而且充分说明了建立模块库的重要意义.

参考文献:

[1] Srinivasa R Arikati, Ravi Varadarajan. A signature based approach to regularity extraction[A]. In Proceedings of IEEE International Conference on Computer-Aided Design[C]. Washington, DC, USA, 1997. 542 – 545.

[2] Thomas Kutzschebauch. Efficient logic optimization using regularity extraction[A]. In Proceedings of the IEEE International Conference On Computer Design: VLSI In Computers & Processors[C]. San Jose, California, United States, 2000. 487– 493.

[3] D Sreenivasa Rao, Fadi J Kurdahi. On clustering for maximal regularity extraction[A]. IEEE Transactions on Computer-Aided Design[C]. Santa Clara, CA, 1993, 12(8): 1198– 1208.

[4] Ryan Kastner, Seda Ogrenci-Memik, Elaheh Bozorgzadeh, Majid Sarrafzadeh. Instruction generation for hybrid reconfigurable systems[A]. In Proceedings of IEEE International Conference on Computer-Aided Design[C]. New York, NY, USA, 2002, 7(4), 605– 627.

[5] Amit Chowdhary, Sudhakar Kale, Phani Saripella, Naresh Sehgal, Rajesh Gupta. A general approach for regularity extraction in datapath circuits [A]. In Proceedings of IEEE International Conference on Computer-Aided Design[C]. San Jose, California, United States, 1998. 332– 339.

[6] Marnix Arnold, Henk Corporaal. Automatic detection of recurring operation patterns[A]. In Proceedings of the seventh International Workshop on Hardware/software Codesign[C]. New York, NY, USA: ACM Press, 1999. 22– 26.

[7] Yuanqing Guo, Gerard J M Smit. A graph covering algorithm for a coarse grain reconfigurable system[A]. Conference of Languages, Compilers, and Tools for Embedded Systems[C]. New York, NY, USA: ACM Press, 2003. 199– 208.

[8] Michel A J Rosien, Yuanqing Guo, Gerard J M Smit, Thijs Krol. Mapping applications to FPGA tile[A]. In Proceeding of Design, Automation and Test in Europe[C]. Munich, Germany, 2003. 1124– 1125.

[9] J A Bondy, U S R Murty. Graph Theory with Applications[M]. Macmillan, London, 1976.

[10] M Corazao, M Khalaf, L Guerra, M Potkonjak, J Rabaey. Instruction set mapping for performance optimization[A]. Proceedings of the IEEE/ACM international conference on Computer-aided design[C]. Los Alamitos, CA, USA, 1993. 518– 521.

[11] F J Kurdahi, C Ramachandran. Evaluating Layout Area Tradeoffs for high level synthesis applications[A]. IEEE Trans on VLSI systems [C]. New York, NY, USA, 1993, 1(1): 46– 55.

[12] L Stok. Interconnect Optimization for Multiprocessor Architectures[A]. In Proc of the IEEE Int'l Conf on Computer Systems and Software[C]. Los Alamitos, CA, USA, 1990. 461– 465.

作者简介:

郎荣玲 1997 年毕业于西北工业大学应用数学系, 2002 年在西北工业大学应用数学系获得硕士学位, 目前在西北工业大学自动化学院攻读博士学位, 主要研究方向: 信息安全、VLSI 自动化设计、运筹学.

E-mail: ronglinglang@163.com; rllang@sohu.com.

戴冠中 1937 年生, 西北工业大学自动化学院教授, 博士生导师, 长期从事信息安全、计算机控制、现代控制理论及应用的研究.