第 36 卷 第 1 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.1 Computer Engineering January 2010
·博士论文· 文章编号：1000—3428(2010)01—0004—03 文献标识码：A 中图分类号：N945

基于相似结构自动提取的 SoC 划分方法

韩睦华，刘雷波，魏少军
(清华大学微电子学研究所清华信息科学与技术国家实验室，北京 100084)

摘 要：提取应用描述中的相似运算结构并使用相似结构划分系统可以有效实现片上系统划分。提出一种基于生长的相似结构自动提取方法，其中的单模板匹配算法实现任意结构模板的提取，多模板生成算法采用模板和子图同步生长的方法。实验结果表明，该方法适用于包含扇出和汇聚结构的模板提取，计算时间与传统方法相比可减少 30%~70%。

关键词：片上系统；划分；相似性

SoC Partition Method Based on Automatic Extraction of Similar Structure

HAN Mu-hua, LIU Lei-bo, WEI Shao-jun
(Tsinghua National Laboratory for Information Science and Technology, Institute of Microelectronics, Tsinghua University, Beijing 100084)

【Abstract】Extracting similar structure and using the structure to cover the system is an effective method for the partition of System-on-Chip(SoC). This paper presents an automatic extraction method of similar structure based on growth, of which Single Template Matching Algorithm(STMA) extracts any kinds of structure of templates, and Multiple Templates Generation Algorithms(MTGAs) uses the incremental and synchronous growth method for all the templates and sub-graphs. Experimental results show that the template containing fan-in and fan-out branches can be supported by the new method, while computing time is reduced by 30%~70% compared with traditional methods.

【Key words】System-on-Chip(SoC); partition; similarity

在多媒体应用的描述中，往往含有大量的相似运算结构，基于相似性的划分方法通过提取描述中的相似结构并将相似结构分配和调度到相应硬件模块中以实现片上系统(System-on-Chip, SoC)的划分[1-4]。相似结构的自动提取是该方法的基础和重点，通常分为手动[2]和自动提取[1,3-4]2 种方法。手动提取方法依靠设计者的经验，因此，不能保证获得最优的划分结果。自动提取方法根据应用自身的特点提取相似结构，扩大了设计搜索空间，因此，可以得到较优的解。本文针对基于相似性的系统划分问题，提出了一种基于生长的相似结构自动提取方法，包括单模板匹配方法和多模板结构提取方法，可以降低子图搜索的冗余操作，减少计算时间。

1 基于相似结构自动提取的SoC划分方法

应用通常用有向无环图来表示：$G(V,E)$，其中，V 表示运算节点集合；E 表示节点之间的数据传递和控制关系集合。描述中的相似运算结构称为模板，表示为 $T(A,C)$。使用模板 T 划分任务图 G，可以得到相匹配的子图集合 $\{S_1^T,S_2^T,\cdots,S_n^T \mid S_i^T \subseteq G\}$。如果将模板 T_i 综合成硬件 Res^{T_i}，那么系统的划分过程可以描述为：提取任务图中的模板 $\{T_i \mid i=1,2,\cdots,L\}$ 和匹配子图集合 $\{\{S_1^{T_i},S_2^{T_i},\cdots,S_n^{T_i}\} \mid T_i, i=1,2,\cdots,L\}$，并寻找任务图 G 的一组子图覆盖 S_t，将覆盖中的子图划分到模板资源 $\{Res^{T_i} \mid T_i, i=1,2,\cdots,L\}$ 中并调度。基于相似结构提取的划分流程如图 1 所示，分为相似结构提取和优化覆盖搜索 2 个过程。相似结构的提取获得应用描述中的各种结构信息，包括各种类型的模板、模板在描述中匹配的子图信息等。优化覆盖搜索选取一组不重叠的子图，并且覆盖整个任务描述，最后通过调度和配置实现应用的功能。

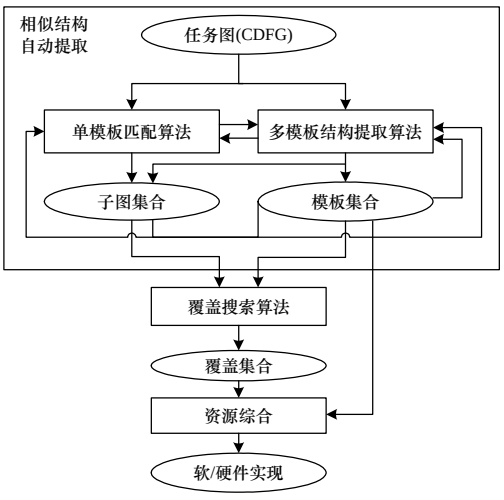

图 1 基于相似结构提取的系统划分方法

2 单模板匹配算法

单模板匹配用于在任务图中识别并提取与特定模板结构相同的子图。本文的单模板匹配算法(Single Template Matching Algorithm, STMA)可以实现包含扇出和汇聚结构子

基金项目：国家自然科学基金资助项目(60506007, 60676012)
作者简介：韩睦华(1976－)，男，博士研究生，主研方向：超大规模集成电路设计，软/硬件协同设计；刘雷波，副教授、博士；魏少军，教授、博士
收稿日期：2009-05-09 **E-mail：**hanmuhua@139.com

图的识别。算法的思想是通过图的变换，将任意结构的子图转换为树状结构的描述，如图 2 所示，具体包括：

(1)有向边变换。通过改变边的方向和属性，将汇聚分枝上的节点变换成扇出分枝上的节点。如图 2 中 S^2 变化得到 $S^{2'}$：由节点 3－>4－>5 组成的汇聚分枝变换成 5－>4－>3 构成的扇出分枝，并且重新定义边(5,4)和(4,3)的属性。边的属性为 0 表示原始边；为 1 表示变换后的边。同样，S^3 通过有向边变换得到图 $S^{3'}$，而图 S^1 无汇聚分支，不需要变换。

(2)节点复制。图 2 中 S^4 的节点 3 同时处于扇出分枝和汇聚分枝上，在这种情况下，变换边(3,4)的同时，需要将节点 3 复制到新的扇出分枝 4－>3'上。同时增加节点 3'和节点 1 的隐性父子关系。

(3)搜索方法。从首节点出发，按照边的方向搜索图中的节点，当遇到汇聚节点时，折回逆向搜索其他汇聚分枝中的节点，直到遇到已访问过的节点。如果访问的节点同时处于某已访问节点的扇出分枝中，则复制该节点，并对搜索路径上的边做变换。采用分枝定界的方法，遍历图中的所有节点。

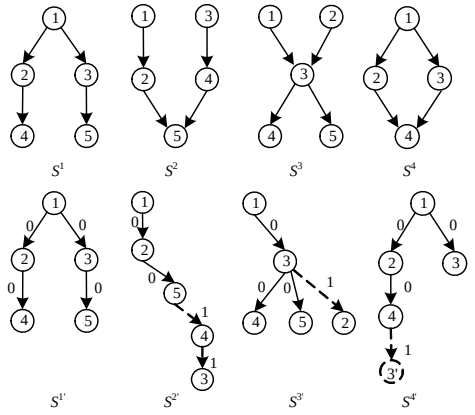

图 2　模板结构与子图变换

单模板匹配算法的输入是任务图 $G(V,E)$、模板 $T(A,C)$；输出是与 T 匹配的子图集合 $S^T=\{S_1^T,S_2^T,\cdots,S_n^T \mid S_i^T \subseteq G\}$。算法描述如下：

1. while(G(V,E)中有未被访问过的节点 v_i)
2. { if(节点 v_i 和模板 T(A,C)中的首节点 a_1 类型不同)转步骤 14;
3. if(节点 v_i 子节点集合和 a_1 的子节点集合相容)将节点对(v_i,a_1)压入堆栈 FILO, flag=1;
else 转步骤 14;
4. while(堆栈 FILO 不为空)
5. { 取出堆栈 FILO 中的一对节点(v_j,a_m);
if(a_m 的子节点未被访问，且有多于一个的父节点)
if(a_m 有未被访问过的父节点){a_m 未被访问过的父节点变换为 a_m 的子节点，并且修改它们之间的边的属性为 1;}
else {复制此节点为 a_m 的子节 a_m'，设定 a_m' 与 a_m 的父节点的隐性连接关系，增加连接边到 a_m，边属性为 2;}
else 转步骤 14;
6. 生成节点 v_j 和 a_m 的子节点的排列 $P(v_j)$和 $P(a_m)$;
7. while($P(v_j)$和 $P(a_m)$中有没访问过的组合)
8. {if((p_w=$\{v_{t1},v_{t2},\ldots,v_{tr}\}$和 p_q=$\{v_{s1},v_{s2},\ldots,v_{sr}\}$节点属性一致，其中，$p_w \subseteq P(v_j)$，$p_q \subseteq P(a_m)$)&&(节点 v_j 和 a_m 至 p_w 和 p_q 中节点的边属性一致))
9. {if(所有节点对$(v_{t1},v_{s1}),(v_{t2},v_{s2}),\cdots,(v_{tr},v_{sr})$ 子节点相容)
for(k=1,k≤r,k++) { if (v_{sk} 有子节点) 则将节点对(v_{tk},v_{sk})压入堆栈 FILO; } }
10. else flag=2;
11. 取下一对组合;}
12. if (flag==2) 清空 FILO，转步骤 14;
13. 取 FILO 中的下一个节点; }
14. 取 G(V,E)中的下一个节点 v_j; }

3　多模板生成算法

基于相似结构提取的划分方法中，通过搜索并选取多个模板划分描述，最终得到多个子图组成的覆盖。多模板生成算法为覆盖搜索提供各种可能的模板结构以及描述中与它们匹配的子图集合。由于模板和子图的数量往往很大，因此需要采取优化策略降低算法的复杂度，减少搜索过程中的冗余操作。

3.1　算法原理

同步生长算法通过子图逐步生长的方法搜索模板和匹配子图，即当前的模板和子图是通过前一级的模板和子图以增加相邻节点的方式获得的。该算法避免使用计算复杂度较高的单模板匹配算法来获得匹配子图，并且模板和子图的同步生长可以减少冗余的模板和子图搜索操作。结合第 2 节的 STMA，可以实现任意结构模板的生成和子图提取。

(1)同步生长

如果 2 个任务图匹配(同构)，那么它们必然存在相互匹配的子图，或者说，相匹配的 2 个任务图可由比它们规模小并且相匹配的一对子图通过生长的方式获得。

定理　给定任务图 $G(V,E)$、模板 $T_1(A_1,C_1)$ 和 $T_2(A_2,C_2)$。$G(V,E)$ 中与模板 $T_1(A_1,C_1)$ 匹配的子图集合为 $S_1=\{S_1^{T_1},S_2^{T_1},\cdots,S_{n_1}^{T_1}\}$，与模板 $T_2(A_2,C_2)$ 匹配的子图集合为 $S_2=\{S_1^{T_2},S_2^{T_2},\cdots,S_{n_2}^{T_2}\}$。如果模板 $T_1 \subseteq T_2$，则对于任意 $S_i^{T_2} \in S_2$，存在 $S_j^{T_1} \in S_1$ 满足 $S_j^{T_1} \subseteq S_i^{T_2} \subseteq G$。

证明：根据上述定理，给定 $S_i^{T_2} \in S_2$，由于集合 S_2 与 T_2 匹配，因此 $S_i^{T_2}$ 与 T_2 同构。又因为 $T_1 \subseteq T_2$，所以 $S_i^{T_2}$ 存在子图 S'，使得 S' 与 T_1 同构。因为 S_1 与 T_1 匹配，所以 $S' \in S_1$，即 S_1 中存在 $S_j^{T_1}$ 满足 $S_j^{T_1} \subseteq S_i^{T_2} \subseteq G$。

该定理说明，如果 T_1 是 T_2 的子图，那么 T_2 对应的匹配子图集合 S_2 一定可以通过 S_1 获得。因此，从节点数量为 1 的模板开始，通过递推关系依次可以得到节点数量为 2, 3, …, n, n+1,…的模板以及它们对应的匹配子图集合。由于子图匹配的计算复杂度较高，而生长过程是复杂度为线性关系的运算，因此本文方法的计算代价更小。

(2)子图生长的禁忌与匹配预判断

如果得到一个含有 n 节点的模板 T_i^n 以及对应的子图集合 $S^{T_i^n}$，那么 $S^{T_i^n}$ 最多可以由 n 种含有 n-1 个节点的子图集合 $\{S^{T_j^{n-1}}\}_{j=1,2,\cdots,n}$ 通过生长方式获得。不失一般性，如果 $S^{T_i^n}$ 由 $S^{T_w^{n-1}}$ 通过生长的方式得到，那么生长 $S^{T_j^{n-1}} -> S^{T_i^n}, j \neq w$ 是重复的。为了消除这类冗余生长操作，本文引入了生长禁忌表，用来标识子图生长时被禁忌的生长方向。在子图生长搜索时，只搜索未被禁忌的生长。为了进一步减少调用匹配操作，使用了以下特征量进行预判断：子图中各类节点的数量，子图的节点面积之和，子图的关键路径长度。

3.2　算法流程

多模板生成算法(Multiple Templates Generation Algorithms, MTGAs)的输入是应用的 CDFG 描述 $G(V,E)$ 以及模板约束；输出是满足约束的模板集合以及每个模板对应的子图集合。算法流程如图 3 所示。模板和子图生长过程的核

心算法是模板与子图搜索算法(Templates and Sub-graphs Searching Algorithm, TSSA)。算法根据 *n*-1 节点的模板和子图通过生长得到 *n* 节点的模板和子图集合。TSSA 在生长过程中建立并搜索子图禁忌表，同时调用匹配预处理与 STMA。经 *n*-1 次迭代，算法得到所有 *n* 个节点的模板和匹配的子图集合。

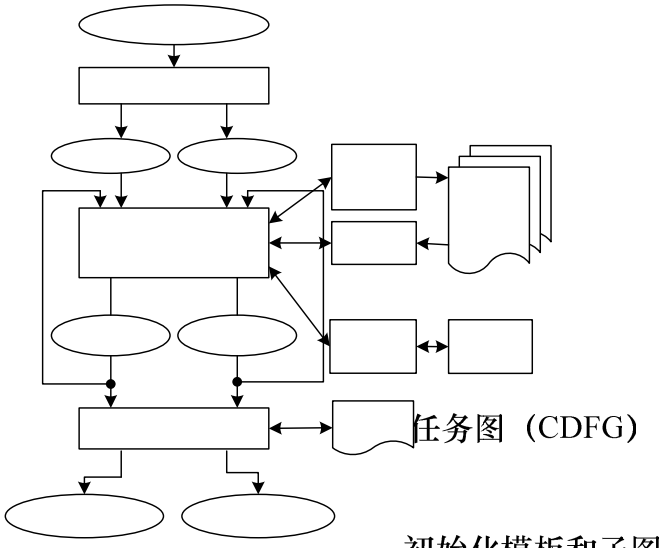

图 3　多模板生成算法

4　实验与分析

STMA 和 MTGAs 算法使用 C++语言编写，运行的平台是 Intel CPU 1.8 GHz，内存为 512 MB。实验应用以运算为主的应用，包括 AR 滤波器(28 节点)、EWF 滤波器(34 节点)、DCT (48 节点)等。

4.1　单模版匹配算法

STMA 的提取结果如表 1 所示。可以看出，该算法可以正确识别和提取含有扇出和汇聚结构的模板。当模板节点数量约为 12 个时，提取所有子图的计算时间通常在 10 s 以内。

表 1　模板类型与子图提取结果

任务图	模板种类	模板节点数量	扇出分枝	汇聚分枝	匹配子图数量
AR	T1	3	否	是	8
	T2	3	是	否	4
	T3	5	是	是	2
	T4	7	否	是	2
EWF	T1	4	是	是	2
	T2	4	是	是	2
	T3	5	是	是	4

4.2　多模板生成算法

MTGAs 对 4 个~9 个节点的模板和子图进行提取，如图 4~图 6 及表 2 所示。

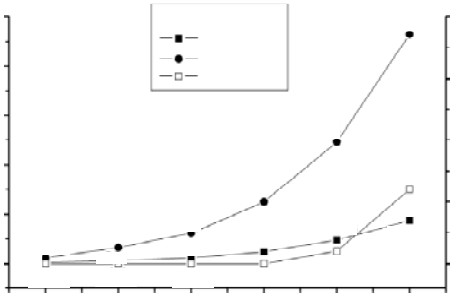

图 4　AR 算法的多模板生成及子图提取

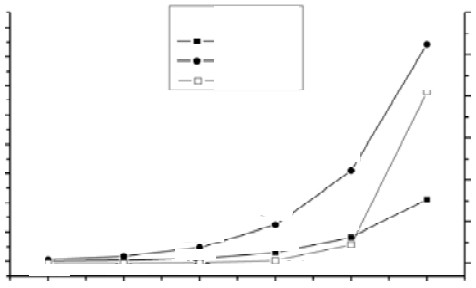

图 5　EWF 的多模板生成及子图提取

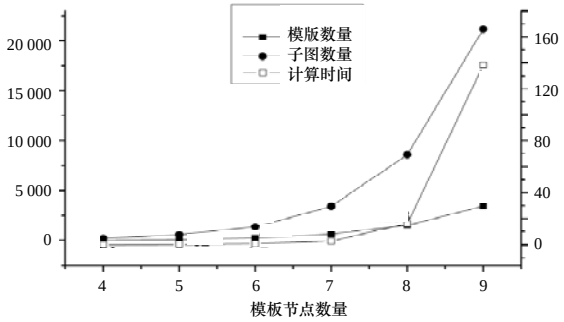

图 6　DCT 的多模板生成及子图提取

表 2　实验结果

模板节点数量	模板数量			子图数量			计算时间/s		
	AR	EWF	DCT	AR	EWF	DCT	AR	EWF	DCT
4	8	33	30	46	291	237	0.0	0.0	0.0
5	29	124	82	230	1 457	560	0.0	0.2	0.1
6	58	457	212	250	2 366	1 371	0.0	1.2	0.7
7	120	1 461	539	496	5 996	3 439	0.0	6.1	2.7
8	236	4 127	1 293	996	15 472	8 602	0.2	41.7	16.1
9	437	10 608	3 036	1 888	37 083	21 187	1.2	410.7	138.2

对于同一个任务图，模板和子图的数量随模板节点规模增长得很快，约为指数关系。根据表 2，EWF 9 个节点的模板数量达 10 608 个，子图数量更是达到了 37 083 个，因此，必须采用优化策略。采用生长禁忌技术后，提取模板和子图的计算时间与模板的数量有关，而不取决于子图数量。更多实验结果显示，对于节点数量为 20~100 的任务图，提取最大节点数量为 9，计算时间是可接受的(1 s~10 min)。

图 7 是采用背包算法的模板和子图生成[1]与采用子图生长禁忌技术和不采用禁忌技术的 MTGAs 算法的结果对比。

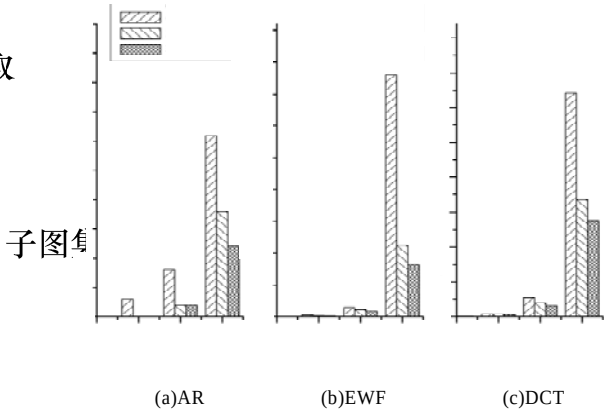

图 7　禁忌与非禁忌生长的计算时间

（下转第 14 页）