NoC映射问题中的列举路径分配算法

岳培培[1]，刘 建[1]，SHEIKH Anjum[1,2]，陈 杰[1]

(1. 中国科学院微电子研究所 北京 海淀区 100029; 2. COMSATS Institute of Information Technology Islamabad Pakistan)

【摘要】映射和路径分配是片上网络在编译过程中两个相辅相成的重要步骤，对系统的通信功耗影响很大。该文针对片上网络映射过程中现有路径分配法寻径不充分的问题，提出了一种基于列举的路径分配算法。该算法通过列举各通信流的所有合法路径，对路径的各种组合方式进行充分搜索。同时将路径分配算法应用到禁忌搜索映射算法中，并对映射算法做了改进，以适应路径分配算法。仿真结果表明，基于列举的路径分配算法提高了满足约束的路径被搜索到的概率，优化了映射算法的结果。

关 键 词 映射; 片上网络; 路径分配; 禁忌搜索
中图分类号 TN47 文献标识码 A

Enumeration-Based Path Allocation Algorithm in NoC Mapping

YUE Pei-pei[1]，LIU Jian[1]，SHEIKH Anjum[1,2]，CHEN Jie[1]

(1. Institute of Microelectronics, Chinese Academy of Sciences Haidian Beijing 100029;
2. COMSATS Institute of Information Technology Islamabad Pakistan)

Abstract Two important steps, namely mapping and path allocation, are tightly bounded with each other in current network on chip (NoC) compiler technology, and have a large impact on the power consumed during communication. A novel algorithm is proposed for path allocation based on an enumerations scheme which enumerates legal paths of traffic, to search the routing paths combination in the NoC mapping process. The proposed algorithm is embedded to a tabu search mapping algorithm which is modified to adapt the behavior of path allocation. The simulation results show that the probability of finding the correct paths is increased within the bandwidth constraints and the mapping algorithm is optimized.

Key words mapping; network on chip; path allocation; tabu search

规则二维网孔结构的片上网络[1-2] (networks on chip，NoC)以其拓扑排列规整、易于布局布线的优势成为NoC研究领域中比较常用的一种结构。开发这种结构的NoC，需要把应用任务分配给适合的IP，然后把IP映射到块中，并为IP之间的通信流分配路径。如果一段连线分配了过多的通信量，将会引起严重的拥塞，造成实时系统的任务无法在时限内完成。因此，需要把路径分配嵌入到映射过程中综合考虑。

现有的映射算法在路径分配时大部分采用XY路由，如文献[3]的分支限界算法、文献[4]的两步遗传算法、文献[5]的NMAP算法等，但XY路由在实际问题中常会出现热点附近通信拥塞的情况。文献[6]采用了多条路径来降低系统的带宽要求，但数据包到达目的节点后需要进行复杂的包排序操作。文献[7]使用的路径分配算法，使路径分配过程既有灵活性，又不需增加额外资源，但该算法存在路径搜索不充分的问题。本文在文献[7]的路径分配算法基础上提出了一种基于列举的路径分配方法，能够对路径进行充分搜索。本文还将该路径分配算法应用到禁忌搜索映射算法中，并针对该路径分配算法对禁忌搜索映射算法做了部分修改，以提高性能时间比。

1 映射和路径分配问题描述

1.1 映射

映射就是把IP和拓扑中的块一一对应，同时要满足某些限制，如带宽限制。映射的优化程度由通信功耗衡量。

映射问题通常被描述为[6-7]：应用任务用任务图描述，表示为有向图TG(IP,*T*)，如图1所示(对应于表

收稿日期：2007－06－06；修回日期：2007－09－16
基金项目：国家自然科学基金(60425413)
作者简介：岳培培(1981－)，女，博士生，主要从事NoC架构方面的研究.

1中的任务图TG2)。

图中的每个顶点IP$_i$代表一个IP，每条有向边T_{ij}代表由IP$_i$到IP$_j$的通信流。每条通信流T_{ij}有两个参数：Vol(T_{ij})为该通信流的流量；Bw(T_{ij})为该通信流的带宽需求，该带宽需求由实时应用的时限(deadline)和通信流的流量确定，对于映射问题该参数为已知量。

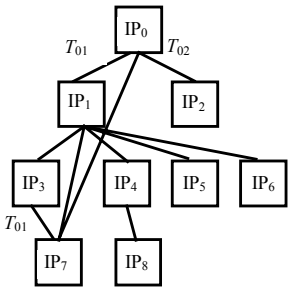


图1　TG(IP,T)

NoC的硬件结构由有向图AG(tile, path)来表示，图中的每个节点tile$_i$都是一个块，可以放置一个IP。每条有向边path$_{ij}$表示连接tile$_i$到tile$_j$的路径。路径由一系列连线(link)组成，而由哪些连线组成是由路由规则和路径分配算法确定的。本文只允许最小化路径。该结构的参数——Bw_constrain是由硬件结构所限定的每条连线所能承受的最大带宽。

在文献[7]的通信能量计算公式中，每位数据的通信能量和跳数成正比。本文还引入了数据在网络接口中的能量消耗，得到的每bit数据通信能量为：

$$E_{\text{bit}}^{i,j}=2E_{\text{NI}}+2E_{\text{NI\_SW}}+(n_{\text{hops}}+1)E_{\text{SW}}+n_{\text{hops}}E_{\text{link}}=(2E_{\text{NI}}+2E_{\text{NI\_SW}}+E_{\text{SW}})+n_{\text{hops}}(E_{\text{SW}}+E_{\text{link}}) \quad (1)$$

式中　E_{NI}表示数据在网络接口(NI)中打包或解包所消耗的能量；$E_{\text{NI_SW}}$是网络接口和路由器(switch)之间的连线消耗的能量；E_{SW}是数据在路由途中经过的路由器消耗的能量；E_{link}是数据在经过路由器间的连线上消耗的能量；n_{hops}表示数据经过的连线数，即曼哈顿距离：

$$n_{\text{hops}}=|x_{m(i)}-x_{m(j)}|+|y_{m(i)}-y_{m(j)}| \quad (2)$$

式中　下标$m(i)$表示IP$_i$到块的映射；$x_{m(i)}$、$y_{m(i)}$表示IP$_i$映射到块的坐标。

每条通信流的能量消耗表示为：

$$E(T_{ij})=\text{Vol}(T_{ij})E_{\text{bit}}^{i,j} \quad (3)$$

由此，对应于该映射的通信能量总和为：

$$E=\sum_{T_{ij}\in\text{TG(IP},T)}\text{Vol}(T_{ij})E_{\text{bit}}^{i,j}=\sum_{T_{ij}\in\text{TG(IP},T)}\text{Vol}(T_{ij})(2E_{\text{NI}}+2E_{\text{NI\_SW}}+E_{\text{SW}})+\sum_{T_{ij}\in\text{TG(IP},T)}(\text{Vol}(T_{ij})n_{\text{hops}})(E_{\text{SW}}+E_{\text{link}}) \quad (4)$$

式中只有 $\sum_{T_{ij}\in\text{TG(IP},T)}\text{Vol}(T_{ij})n_{\text{hops}}$ 由映射控制，其余各项对于固定硬件结构均为常数。

映射算法的目标就是得到一个映射m，使得该映射的通信能量总和最小，即使：

$$\sum_{T_{ij}\in\text{TG(IP},T)}\text{Vol}(T_{ij})(|x_{m(i)}-x_{m(j)}|+|y_{m(i)}-y_{m(j)}|) \quad (5)$$

达到最小值。同时每条连线l_k都要满足带宽要求：

$$\sum_{T_{ij}\in\text{TG(IP},T)}\text{Bw}(T_{ij})f(l_k,\text{path}_{m(i),m(j)})<\text{Bw\_constrain} \quad (6)$$

式中　$f(l_k,\text{path}_{m(i),m(j)})=\begin{cases}0 & l_k\notin\text{path}_{m(i),m(j)}\\ 1 & l_k\in\text{path}_{m(i),m(j)}\end{cases}$

1.2　路径分配

路径分配需要考虑免死锁问题，为了避免使用面积需求大的虚通道免死锁方法，可以在路径分配过程中使用一定的路由规则[8-9]来对可用路径进行限制。将符合路由规则的路径称为合法路径。对于某个映射，如果为每条通信流都分配了合法路径以后，所有连线都没有超出硬件系统的带宽限制，则该映射称为有效映射，所对应的合法路径的组合称为有效路径组合。

文献[7]中使用了一种最小化免死锁路径分配算法(在本文中称为单步法)。在单步法中，每步按照指定规则判断方向并前进一步，然后再判断。该方法能够以较快的速度进行路径分配，但缺点是可能错过合法路径分配，如图2所示。假设结构限制的最大带宽是10，tile$_3$→tile$_4$、tile$_0$→tile$_1$各有一条已分配的通信流，带宽分别是8和6。若一个带宽要求是3的通信流从tile$_0$出发到tile$_4$，可以选择的路径有tile$_0$→tile$_3$→tile$_4$或tile$_0$→tile$_1$→tile$_4$。根据单步法，在第一步时会选择已经占用带宽小的tile$_0$→tile$_3$，第二步只能选择tile$_3$→tile$_4$。但由于该连线的带宽已经占用了8，加入该通信流后超出了最大带宽限制，所以该映射被判定为无效。而从tile$_0$→tile$_1$→tile$_4$可以找到满足带宽限制的路径。可见，单步法可能会导致有效映射被判定为无效，错过某些优化的映射。

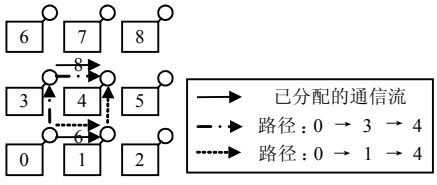


图2　单步法错过的最优路径分配

2 基于列举的路径分配算法

2.1 算法介绍

本文提出的一种基于列举的路径分配算法(简称列举法)，能够对路径进行充分搜索。在给定映射下，把通信流按照灵活性分类：只有一条合法路径存在的通信流称为固定通信流，存在两条及以上合法路径的通信流称为灵活通信流。首先分配所有固定通信流的路径，然后按照带宽需求递减的顺序来分配灵活通信流。在为灵活通信流分配路径时，列举出每个灵活通信流的所有合法路径，对这些路径的组合方式进行遍历式的搜索，寻找有效路径组合。若所有的路径组合方式搜索完毕仍没有找到有效组合，判定该映射无效。算法的伪代码如下：

(1) for each T_{ij} in current mapping{

(2) if no flexibility, add Bw(T_{ij}) to every $l_k \in \text{path}_{m(i),m(j)}$ in bw_table

(3) else{

(4) add T_{ij} to flex_traffic_table

(5) record all legal $\text{path}_{m(i),m(j)}$ to path_table(T_{ij})}}

(6) if already exceed Bw_constrain, exit

(7) repeat{

(8) for each T_{ij} in flex_traffic_table{

(9) choose cur_path $\in$ path_table(T_{ij})

(10) add Bw(T_{ij}) to $l_k \in$ cur_path in bw_table}

(11) if not exceed Bw_constrain，record the paths, exit

(12) if already try every combination of paths, report invalid mapping and exit}

其中，bw_table是一个带宽存储表，记录结构中每条连线的已占用带宽。当为某个通信流选定一条路径后，将表中该路径经过的每条连线都加上该通信流的带宽值。分配完所有通信流的路径后，检查bw_table就可以得到该路径组合是否超出带宽限制。

flex_traffic_table保存该映射中的灵活通信流。每条灵活通信流T_{ij}对应一个path_table(T_{ij})，用来存储该通信流的所有合法路径。按照步骤8、9，从每个path_table中各取出一条路径得到路径组合。

2.2 实验结果

以一组固定映射下找到有效映射的次数为性能指标，对两种路径分配算法进行实验对比。使用软件TGFF[10]随机产生十个任务图，其中五个是3×3结构，五个是4×4结构，并为每个任务图随机产生10 000个映射。然后，分别用单步法和列举法对各映射进行路径分配。路由规则都采用奇偶转弯模型(OE)[9]。用两种方法分别得到的有效映射的数量如表1所示。

从表1可以看出，使用列举法能找到的有效映射相比于单步法更多。即某些映射本来是有效的，但使用单步法不能找到有效的路径组合，从而被错误的判为无效，而使用列举法就能准确判断。

表1 不同路径分配算法在固定映射下找到的有效映射数量

	任务图	OE单步法	OE列举法
3×3结构	TG1	30	78
	TG2	24	37
	TG3	150	693
	TG4	100	130
	TG5	933	1 560
4×4结构	TG6	9 400	9 951
	TG7	2 293	6 075
	TG8	3	38
	TG9	3 504	7 484
	TG10	138	605

3 列举法在禁忌搜索映射法中的应用

3.1 禁忌搜索映射算法

从禁忌搜索算法在求解旅行商问题中的应用[11]，得到二维网孔结构NoC的禁忌搜索映射算法：

(1) cur_map = greedy_map, best_cost = cost(cur_map)

(2) repeat M times{

(4) repeat N times{

(5) random generate neighbor_map, new_cost = cost (neighbor_map)

(6) if neigher_map better than candidate, candidate= neighbor_map

(7) if new_cost<best_cost, best_cost=new_cost}

(8) cur_map = candidate, clear candidate

(9) update tabu_list}

步骤1中贪婪映射(greedy_map的方法参照文献[3]，每次选取与已映射的IP通信量最大的未映射IP，计算理想X和Y位置，并寻找与该位置最近的未映射块放置该IP，重复该过程直到所有IP映射完毕。

步骤1、5代价计算cost(map)：因为式(5)和通信功耗成线性关系，所以用式(5)来度量映射的代价。在无效映射时，即不满足式(6)时，cost = ∞ 。

步骤5邻域映射(neighbor_map)的产生：在当前映射(cur_map)中随机挑选两个IP，互换其位置。

步骤6候选映射(candidate)的更新：邻域映射首先要经过禁忌检查，若该项在禁忌表中存在，该邻域映射是被禁忌的。但由于现有的最优代价(best_cost)的邻域映射不受禁忌影响。候选映射的选择规则是：优于best_cost的邻域映射>不被禁忌的代价最低邻域映射>被禁忌的代价最低邻域映射。

步骤9禁忌表(tabu_list): 禁忌表中每项都是互换的两个IP号，每次用候选映射更新当前映射后，用候选映射互换的IP号代替表中时间最久的项。

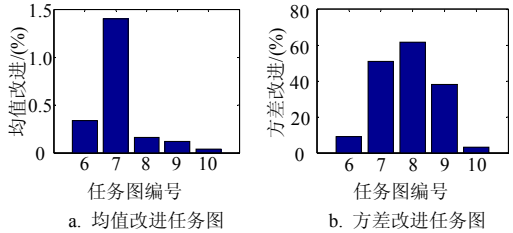
3.2　列举法加入禁忌搜索映射算法

将列举法加入禁忌搜索映射算法。考虑到列举法消耗的时间较长，对禁忌搜索映射算法做如下修改来减少路径分配算法的调用次数，而不影响最终结果：首先使用式(5)得到该映射的临时代价，使用临时代价来评价该邻域映射是否需要替换候选映射，若不需替换，那就不调用路径分配算法，因为该邻域映射的代价无论等于临时代价还是∞，都不会替换候选映射，对结果也没有影响。若临时代价计算后发现需要替换候选映射，再调用路径分配算法。若路径分配成功，替换候选映射；若不成功，该邻域映射 cost = ∞，重新判定是否替换候选映射。

该方法大大减少了路径分配算法的调用次数。经统计，在本文的列举法实验结果中，只有5.4%的邻域映射需要调用路径分配算法。

3.3　实验结果

本文计算了4×4结构的任务图在禁忌搜索映射算法下的功耗，其中调用的路径分配算法分别采用单步法和列举法，使用OE路由规则。每个任务图都用两种路径分配法分别运行100次。从图3可以看出，加入列举法的映射算法的均值相对于单步法有了一定的改进，目方差改进比较多。而且，在使用单步法寻找任务图8的最佳映射的时候，在100次仿真中有两次没有找到有效映射(这两次的结果没有加入到均值和方差的计算中)。而使用列举法总能找到有效映射。实验结果表明：相对于单步法，使用列举法能得到更优而且更稳定的结果。



a. 均值改进任务图　　b. 方差改进任务图

图3　列举法相对于单步法对映射算法的均值和方差改进

但使用列举法的映射算法计算花费的时间比较长。实验采用主频为3 GHz的奔腾4双核CPU进行计算，系统内存为1 GB。算法使用C语言编写。在图3任务图的映射中，采用单步法的计算时间一般在1 s以内；列举法的计算时间基本也都在几秒以内，只有少数情况下(<5%)，需要几分钟至几十分钟。但因为映射算法工作在编译阶段，每个应用只需一次计算，所以花费的时间是可以接受的。

4　结　论

本文提出了一种适用于NoC映射问题的列举路径分配算法，寻找符合约束的路径组合。通过对随机映射的试验，该算法找到有效映射的概率大于文献[7]中使用的单步法。将该算法嵌入到一个改进的禁忌搜索映射算法中，也优化了映射算法的结果。

但该算法的不足之处是计算时间相对比较长。在IP比较多的时候(采用5×5的拓扑进行试验)，该算法的时间增长比较快，还需寻求新的改进方案。

参　考　文　献

[1] DALLY W J, TOWLES B. Route packets, not wires: on-chip interconnection networks[C]//DAC 2001. Las Vegas: ACM, 2001:684-689.

[2] KUMAR S, JANTSCH A, MILLBERG M, et al. A network on chip architecture and design methodology[C]// ISVLSI'02. Pittsburgh: IEEE, 2002: 105-112.

[3] HU J, MARCULESCU R. Energy-aware mapping for tile-based NoC architectures under performance constraints[C]// ASP-DAC 2003. Japan: ACM, 2003: 233-239.

[4] LEI T, KUMAR S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture[C]// DSD'03. Turkey: IEEE, 2003: 180-187.

[5] ASCIA G, CATANIA V, PALESI M. Multi-objective mapping for mesh-based NoC architectures[C]// CODES+ISSS'04. Stockholm: ACM, 2004: 182-187.

[6] MURALI S, MICHELI G D. Bandwidth-constrained mapping of cores onto NoC architectures[C]//DATE'04. Paris, France: IEEE, 2004: 896-901.

[7] HU J, MARCULESCU R. Energy- and performance-aware mapping for regular NoC architectures[J]. IEEE TCAD, 2005, 24(4): 551-562.

[8] GLASS C J, NI L M. The turn model for adaptive routing[C]//19th ISCA. Queensland: ACM, 1992: 278-287.

[9] CHIU G. The odd-even turn model for adaptive routing[J]. Parallel and Distributed Systems, 2000, 11(7): 729-738.

[10] DICK R、Task Graphs For Free(Version 2.0)[DB/OL]. [2007-06-06]. http://ziyang.ece.northwestern.edu/tgff.

[11] 贺　一, 刘光远. 禁忌搜索算法求解旅行商问题研究[J]. 西南师范大学学报(自然科学版), 2006, 27(3): 341-345.

编　辑　张　俊