有向无环图的高效归约算法

侯 睿 武继刚
(天津工业大学计算机科学与软件学院 天津 300387)
(中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

摘 要 将一个应用程序部署到给定的片上网络上执行时,需要将应用程序中的每一个子任务都指派给片上网络中的一个节点执行。该问题一般被建模成一组子任务作为顶点的有向无环图,任务在片上网络上的部署过程就等同于一个有向无环图的顶点向一个片上网络拓扑映射的过程。而随着应用程序和片上网络规模的增大,计算一个最优的映射方案是典型的难解问题。为了加速有向无环图到片上网络拓扑的映射过程,提出了有向无环图的归约算法,使归约后的图中的顶点数量尽可能地与给定片上网络中的节点数量相同。提出的图归约算法可以有效地识别出所有可归约子图,这些可归约子图可被归约为单一顶点。新算法的适用范围从嵌套图扩展到了任意图,并且拥有与原算法相同的复杂度量级。还提出了一种并行化的算法思想来加速可归约子图的搜索过程。

关键词 片上网络,有向无环图,图归约,可归约子图
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.017

Efficient Reduction Algorithms for Directed Acyclic Graph

HOU Rui WU Ji-gang
(School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)
(State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract When deploying an application to a given network-on-chip (NoC) to execute, each task of the application should be assigned to a tile in the NoC seperately first. The application is generally modeled as a directed acyclic graph (DAG) in which tasks are repersented as vertices, and the deployment process is equivalent to mapping a DAG to the topology of a NoC. With the increasing scale of applications and NoCs, finding out an optimal mapping scheme is a typical intractable problem. To accelerate the process of mapping DAGs to a given NoC system, this paper proposed an efficient reduction algorithm for DAG, so that the number of vertices in the DAG after reduction is close to the number of tiles in the NoC system. The proposed algorithm can effectively identify all reducible subgraph, which can be reduced to a single vertex. The new algorithm extends the applicable scope from nested graphs to arbitrary DAGs, with the same level of computational complexity compared to the original algorithm. This paper also presented a parallelized algorithm which can shorten the process of seraching reducible subgraph.

Keywords Network-on-chip (NoC), Directed acyclic graph, Graph reduction, Reducible subgraph

1 引言

随着半导体工业的蓬勃发展,计算单元、逻辑控制单元、内存、开关,甚至路由器组成的整个处理系统(processing system)已经可以被集成到一个单一芯片内。目前,一个片上互联系统(on-chip interconnection system)已经能将上千个处理器集成在一起[1,2]。相应地,片上互联系统的复杂性也急剧膨胀。对于一个采用50纳米制造工艺的芯片来说,贯穿芯片的传播延迟已在6到10个时钟周期之间[3]。如此大的传播延迟导致系统维持运算单元间的计算同步异常复杂。

作为片上互联系统的一个典型架构,片上网络(Network-on-Chip,NoC)侧重于组件的高度模块化,从而实现组件的高度可重用性,并具有极好的可扩展性[4,5]。片上网络的常用结构为mesh网络。而在mesh网络中的每一个节点(tile)都由一个处理器(processor)和一个路由器(router)组成。其中,处理器都是通过和它直连的路由器与它的4个邻居相连。由于每个处理器都能运行在各自独立的时钟域(clock domain)内,而且不同的处理器可以通过和它直连的路由器进行相互之间的异步数据通讯,设计者可以将精力集中于提高片上网络的性能和扩展片上网络的功能上。当一个新应用要部署在一个以mesh为连接拓扑的片上网络系统运行时,它需要先将自身包含的子任务(task)映射到片上网络中的节点上才能

到稿日期:2014-07-05 返修日期:2014-10-15 本文受国家自然科学基金(61173032),国家自然科学基金天元青年基金(11326211,11326198),计算机体系结构国家重点实验室开放课题(CARCH201303)资助。
侯 睿(1989-),男,硕士生,主要研究方向为网络可靠性、高性能计算,E-mail:asrhou@gmail.com;武继刚(1963-),男,博士,教授,主要研究方向为高性能算法设计、组合优化算法设计与分析、并行分布计算,E-mail:asjgwu@gmail.com。

开始执行。将一个任务映射到一个片上网络系统的过程一般分成两步完成。首先,任务要被细分成一组相互存在通讯的子任务,这些子任务和相互之间的通讯一般被建模成一个有向无环图(directed acyclic graph,DAG)。接下来,每一个子任务,即有向无环图中的顶点(vertex)被一一指派到 mesh 网络中的节点上,再根据有向无环图的依赖关系有序调度执行。

如果应用程序中的子任务被指派到片上网络中的固定节点,且在应用程序的整个执行过程中都不再变化,那么这种映射过程称为静态映射。而如果在程序开始运行以后,程序中的子任务仍然可以从片上网络中的一个节点被改派到另一个节点去执行,那么这种映射过程就称为动态映射,其运算过程贯穿整个应用程序执行周期。显然,动态映射可以给应用程序带来更好的运行性能,但同时也更复杂和困难。由于映射质量的好坏对程序的运行性能有极大的影响,已经有大量的研究着力于解决动态和静态映射问题。为了获得一个能源消耗最少且整个程序执行时间最短的最优映射,整个程序需要被划分成一组不能被继续划分的原子任务(atomic task)。而即使是相对简单的静态映射过程,在将一个包含 p 个原子任务的应用程序映射到一个基于 $m\times n$ 个节点的 mesh 网络时,其可行解也有 $(mn)^p$ 个。面对如此巨大的解空间,要找到一个最优静态映射是非常耗时的。另外,由于不同应用程序间的运行逻辑、所执行的任务都全然不同,使得不同应用程序在同一片上网络的映射方案是完全异构的,任何两个映射方案间几乎没有可重用之处。在对映射方案质量与映射方案求解时间这两者权衡之后,大部分研究者选择在合理的时间内,利用遗传算法[6,7]、分支定界算法[8,9]、整数规划的算法[10]或启发式算法[11]找到一个近似最优的映射方案。

除了在映射过程的第二步,通过降低所求问题的解的质量来减少寻找一个可行解的时间,通过合理减少应用程序对应的有向无环图中顶点的数量,即归约有向无环图,从而缩小解空间,也是一个加速寻找可行映射方案的途径。但是,有向无环图中的顶点数量并不是可以随意削减的,如果将一个子图归约成一个顶点时破坏了原图中节点间的依赖关系,进而破坏了程序的有序执行,将得不偿失。之前的研究者在类似问题上做了大量探索,但都有各自的局限[12-16],而文献[17]提出了一个能以灵活的粒度(flexible granularity)归约有向无环图的方法,该方法在文中用于解决硬件/软件划分问题。但是,该方法只能用于有严格层次关系的有向无环图,称之为嵌套图(nested graph),且该算法的时间复杂度达到了 $O(n^3)$。本文提出了一种新的有向无环图的归约方法,该方法适用于任意的有向无环图,而且该方法的时间复杂度不高于文献[17]的方法。

本文第 2 节介绍所研究问题的相关背景以及基本定义;第 3 节分析了有向无环图中可归约子图的一些性质;第 4 节给出所提出的在有向无环图中寻找所有可归约子图的算法的描述;第 5 节对 3 种算法做了实验比较和分析;最后总结全文。

2 问题背景及基本定义

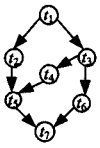
在一个应用程序中,任意两个子任务间的依赖关系是固定的,且只能由一方指向另一方。另外,如果应用程序中的多个子任务间的依赖关系构成了一个环,那么这些任务会一直循环执行,不会停止,相应地,这种情况会使得应用程序的执行无法停止。这显然既不利于片上网络的基础设施被多个应用程序所复用,也不符合一个应用程序会在一段有限的时间内停止执行的有限性约束。所以一个应用程序中的子任务不可能构成环。

在本文中,沿用之前的假设,将一个应用程序建模成一个有向无环图[18-23]。其中,有向无环图的顶点代表应用程序中的子任务,而有向无环图中的边用来指示子任务间的依赖关系。显然,有向无环图中没有重边,即任意两个子任务间不会有多重依赖关系。如果有,多重依赖关系也可以合并成一个依赖关系。

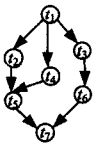
给定有向无环图 $G(V,E)$,其中 V 表示图中的顶点集合,$|V|=n$,每个顶点表示对应的应用程序中的一个子任务;E 表示图中有向边的集合,$|E|=m$,有向边的存在表示两个顶点所代表的子任务间有数据通讯。令 $V=\{t_0,t_1,\cdots,t_{n-1}\}$,即每个顶点可以用 t_i 表示,那么,有向边可以用 (t_i,t_j) $(0<i,j<n)$ 表示,其中 t_i 被称作有向边 (t_i,t_j) 的头顶点(head),t_j 被称作有向边 (t_i,t_j) 的尾顶点(tail)。另外,有向边 (t_i,t_j) 象征了顶点 t_j 所代表的子任务只有在顶点 t_i 所代表的子任务完成后才能开始执行。也就是说,t_i 是 t_j 的前驱顶点(predecessor),记作 $t_i\in pred(t_j)$,而 t_j 是 t_i 的后继顶点(successor),记作 $t_j\in succ(t_i)$。

如果有向无环图中的顶点没有前驱顶点,只有后继顶点,那么该顶点就称作入口顶点(entry)。而如果图中的顶点只有前驱顶点,没有后继顶点,那么该顶点就称作出口顶点(exit)。本文所研究的有向无环图只有一个入口顶点和一个出口顶点,这是因为,在一个应用程序开始执行时,如果存在几个子任务可以没有约束地同时执行。那么,总存在一个子任务负责将该应用程序引导激活,这个子任务只可能有一个,这样的子任务在应用程序对应的有向无环图中会被建模成入口顶点。同样,在应用程序中,总有一个子任务负责在其余的子任务都执行完毕后通知系统终结该应用程序,释放其所占用的资源,这样的子任务在应用程序对应的有向无环图中会被建模成出口顶点。除了入口顶点和出口顶点对应的应用程序中第一个和最后一个执行的子任务,其余的子任务必然会被其前驱任务激活执行,在执行完毕后再激活其后继任务执行,否则该子任务在应用程序中是无意义的。也就是说,在应用程序对应的有向无环图中,没有任何顶点是孤立的,而且除了入口顶点和出口顶点,其余顶点都至少有一个前驱顶点和一个后继顶点。

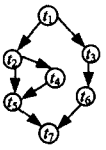
如果一个有向无环图是可归约的(见图 1(b),(c)),那么将其中的一个可归约子图归约成一个顶点时,任何两个顶点间的依赖关系依然保持不变。因此,一个可归约的有向无环图可以被一个更小规模的有向无环图表示而不破坏原图中各个顶点间的依赖关系,这就可以在将应用程序映射到片上网络时,在不降低映射方案解质量的同时,减少获取最优映射方案的时间。



(a)不可归约的有向无环图



(b)可归约的有向无环图



(c)嵌套图

图 1

定义 1(可归约性) 一个有向无环图 $G(V,E)$ 是可归约的当且仅当存在一个子图 $G'(V',E')(V'\subseteq V,E'\subseteq E)$ 只有一个入口顶点 v_i 和一个出口顶点 v_j，而且其有向边的集合 E' 包含了由其顶点的集合 V' 导出的所有有向边，除了以 v_i 为头顶点和以 v_j 为尾顶点的有向边。这样的一个子图 $G'(V',E')$ 就被称为是一个可归约子图。

换句话说，如果一个有向无环图包含了至少一个可归约子图，那么该有向无环图就是可归约的。由于可归约子图也可以由它的入口顶点 v_i 和出口顶点 v_j 唯一地确定，因此可归约子图也可以被表示成如$\langle v_i,v_j\rangle$的形式。

为了加速从有向无环图到片上网络的映射过程，本文将研究如下问题：给定一个只有一个入口顶点和一个出口顶点的有向无环图，找出其所包含的所有可归约子图。这些可归约子图在原有向无环图被归约时，都可能被替换成一个顶点，以达到归约图中顶点的目的。根据以 mesh 拓扑结构为基础的片上网络中节点的数目，归约应用程序所包含的原子任务组成有向无环图时，一些包含较少顶点数的可归约子图会首先被归约，如此循环执行直到有向无环图中顶点的数量不会远大于片上网络中节点的数量，以使给定的有向无环图可以在合理的时间内被映射到片上网络中。

3 可归约子图的特性

本节将研究可归约子图的一些特性，利用这些特性，可以在一个有向无环图中高效地查找其所包含的可归约子图。假设有向无环图中的每个顶点都会产生一条消息(message)，而且所有顶点都会按照如下规则传递自身产生的或从其前驱顶点接收到的消息：一个顶点总是可以将自己产生的消息或从其前驱顶点接收到的消息发送给它的所有后继顶点，如果一个顶点有 k 个后继顶点，那么它会将要传输的消息平均分成 k 份，然后将每一份分割后的消息传递给一个后继顶点。如果有向无环图按照如上的规则产生或传递消息，那么就可以得到如下的一些结论。

引理 1 在有向无环图 $G(V,E)$ 中，如果顶点 $t_j(t_j\in V)$ 只有一个前驱顶点 $t_i(t_i\in V)$，而且顶点 t_i 也只有一个后继顶点 t_j，那么由顶点 t_i、t_j 与有向边(t_i,t_j)所组成的子图 G' 是有向无环图 G 的一个可归约子图。

证明：如图 1(b)所示，t_6 是 t_3 唯一的后继顶点，t_3 是 t_6 唯一的前驱顶点。在由顶点 t_3、t_6 和有向边(t_3,t_6)组成的子图中，t_3 是子图的入口顶点，t_6 是子图的出口顶点，子图中的入口顶点和出口顶点都只有一个，有向边(t_3,t_6)是顶点 t_3 和 t_6 能够导出的唯一一条有向边。根据定义 1，显然该子图是一个可归约子图。证毕。

由引理 1 可得，只要有向无环图中的所有顶点都按照之前给出的规则传递消息，那么很显然在一个可归约子图的入口顶点所产生的消息会被同一图中的出口顶点完整地收到，而不是只收到整个消息中的一部分。由此，可以得到如下的定理。

定理 1 有向无环图中的一个子图 G' 是可归约的当且仅当子图 G' 的出口顶点 t_i 可以接收到产生自子图 G' 的入口顶点 t_j 的完整消息，而且出口顶点 t_i 所能够接收到的产生自子图 G' 外部顶点的消息的量(完整的或部分的消息)，与入口顶点 t_j 所能接收到的消息的量相同。

证明：根据定义 1，在一个可归约子图中，只有出口顶点没有后继顶点，考虑到可归约子图中的内部顶点都有至少一个前驱顶点和后继顶点，因此，出口顶点可以接收到产生于可归约子图中所有其余顶点的完整消息。当然，出口顶点也能够接收到产生自入口顶点的完整消息。考虑到除了入口顶点，G' 中的顶点不可能是任何 G' 的外部顶点的后继顶点，所以 G' 的内部顶点接收到的产生于外部顶点的消息全都是源自于入口顶点。既然出口顶点能够接收到产生于可归约子图 G' 中所有其余顶点的完整消息，它也能接收到 G' 中所有其余顶点所接收到的所有消息，故出口顶点所能接收到的产生于 G' 以外顶点的消息的量(完整的或部分的消息)应该与入口顶点所能接收到的量完全一致。证毕。

定理 2 两个可归约子图 $G_i(V_i,E_i)$ 和 $G_j(V_j,E_j)$ $(|V_i|>|V_j|,|E_i|>|E_j|)$ 不可能有共同的入口顶点或出口顶点，除非 $G_i(V_i,E_i)$ 完全包含 $G_j(V_j,E_j)$，即 $V_j\subseteq V_i$ 且 $E_j\subseteq E_i$。另外，$G_i(V_i,E_i)$ 还要在将可归约子图 $G_j(V_j,E_j)$ 替换成一个顶点后仍然是一个可归约子图。

证明：假设一个有向无环图的子图 $G_i(V_i,E_i)$ 和 $G_j(V_j,E_j)(V_j\subseteq V_i,E_j\subseteq E_i)$ 是两个可归约子图，它们有共同的入口顶点 t_1，而它们的出口顶点分别是 t_x 和 t_y。根据定理 1，t_x 和 t_y 都能收到产生自 t_1 的完整消息。由于 $V_j\subseteq V_i$ 且 $t_x\in V_j$，t_y 所接收到的发自 t_1 的消息全部都是由 t_x 负责转发的，更进一步，所有发自顶点 t_x 和 t_1 间顶点的消息也都是经过 t_x 才发送给 t_y 的。显然，以 t_x 为入口顶点，t_y 为出口顶点的子图$\langle t_x,t_y\rangle$也是一个可归约子图。也就是说，可归约子图 $G_j(V_j,E_j)$ 的出口顶点和$\langle t_x,t_y\rangle$的入口顶点重合，其重合顶点为 t_x。将可归约子图 $G_j(V_j,E_j)$ 用顶点 t_x 代替，原可归约子图 $G_i(V_i,E_i)$ 就变成了子图$\langle t_x,t_y\rangle$，其仍然是一个可归约子图。

若有向无环图的子图 $G_i(V_i,E_i)$ 和 $G_j(V_j,E_j)(V_j\subseteq V_i,E_j\subseteq E_i)$ 是两个可归约子图，它们有共同的出口顶点 t_n，$G_i(V_i,E_i)$ 的出口顶点是 t_x，$G_j(V_j,E_j)$ 的出口顶点是 t_y，那么根据定理 1，t_n 可以接收到产生于 t_x 和 t_y 的完整消息。如果一些产生自 t_x 的消息没有通过 t_y 就到达了 t_n，那么子图 $G_j(V_j,E_j)$ 就不是一个可归约子图，因为除了子图的入口顶

点 t_y 和出口顶点 t_x，其他内部顶点导出的有向边或某些以出口顶点 t_x 为尾顶点的有向边没有被包括在有向边集合 V_j 内，使得产生自顶点 t_x 的消息可以不经过入口顶点 t_y 到达出口顶点 t_n。

另外，假设子图 $G_i(V_i,E_i)$不包含子图 $G_j(V_j,E_j)$。那么，若 t_1 是子图 $G_i(V_i,E_i)$和 $G_j(V_j,E_j)$的入口顶点，即 $t_1\in V_i\cap V_j$，那么两个子图的出口顶点均不可能收到产生自 t_1 的完整消息，也就说明了两个子图 $G_i(V_i,E_i)$和 $G_j(V_j,E_j)$都不是可归约子图。同样，若 t_n 是子图 $G_i(V_i,E_i)$和 $G_j(V_j,E_j)$的共同的出口顶点，即 $t_n\in V_i\cap V_j$。虽然 t_n 能够接收到产生自两个子图不同入口顶点的完整消息，但是任何一个子图都不包含其一个可归约子图按定义应该包含的所有有向边。所以这种条件下的两个子图 $G_i(V_i,E_i)$和 $G_j(V_j,E_j)$都不是可归约子图。

综上可知，定理成立。证毕。

根据本节中这些对可归约子图特性的了解，本文提出了在有向无环图中搜索可归约子图的通用算法。

4 搜索可归约子图的算法

令$\langle v_i,v_j\rangle$表示入口为 v_i、出口为 v_j 的可归约子图。根据上一节对可归约子图特性的分析，可以看出存在一种递归结构：如果两个可归约子图$\langle v_i,v_j\rangle$和$\langle v_j,v_k\rangle$的出口顶点和入口顶点重合，那么子图$\langle v_i,v_k\rangle$也是一个可归约子图。因此，以有向无环图中的每一个顶点作为一个可归约子图的入口顶点，找出以该顶点为入口顶点的最小可归约子图，所有这样找出的可归约子图均为该有向无环图中的不可进一步分割的原子级可归约子图，以这些可归约子图为基础，根据之前发现的可归约子图的递归结构，就可以组合出原有向无环图中的所有可归约子图。因此，本文提出的可归约子图搜索算法只要找出所有原子级的可归约子图，其他可归约子图就自然而然得到了。在将初始的有向无环图向片上网络映射时，可以根据需要将找出的所有可归约子图选择归约成单个顶点。

本文提出的算法分为串行算法和并行算法两种，均可用于在任意给定的应用程序对应有向无环图中搜索所有的可归约子图。文献[17]中提出算法只适用于嵌套图的可归约子图的搜索。与其不同，本文提出的算法对可归约子图的搜索扩展到了任意图上，并没有增加算法的计算复杂度，且提出的并行算法加速了可归约子图的搜索。

4.1 串行算法

·算法思想

在搜索可归约子图的串行算法中，使用信息传递矩阵 M 来记录每个顶点所产生的一份消息被哪些顶点接收到，以及所接收消息的完整程度。在矩阵 M 中，$M[t_x][t_y]$代表由顶点 t_y 产生的消息是否能被 t_x 接收到，如果能，其接收到的消息的完整程度如何。如果 $M[t_x][t_y]$的值为 0，则意味着 t_x 不能接受到 t_y 产生的消息，如果 $M[t_x][t_y]$的值为 1，那么 t_x 能够接受到产生自 t_y 的完整消息，意味着子图$\langle t_y,t_x\rangle$可能是一个可归约子图，另外需要说明，由于在串行算法中只查找原子级的、不可能进一步分割的可归约子图。因此当 $M[t_x][t_y]$的值为 1 时，t_x 不再继续将其所收到的产生于 t_y 的消息向 t_x 的后继顶点转发。这样，根据定理 1 和定理 2，$\langle t_y,t_x\rangle$如果不是一个可归约子图，那么就不存在以 t_y 为入口顶点的可归约子图。另外，若 $M[t_x][n+1]$的值不为 0，则意味着$\langle t_{M[x][n+1]},t_x\rangle$是一个可归约子图。

算法以有向无环图的一个拓扑排序(topology order)的顺序逐个遍历有向无环图中的所有顶点。对于每一个顶点 t_i，算法都先识别该顶点拥有多少个后继顶点。如果只有一个后继顶点，那么很可能 t_i 与其后继顶点可以构成一个引理 1 中所述的可归约子图。故算法检查该后继顶点是否只有一个前驱顶点，如果只有一个，可以判定这两个顶点能够构成一个可归约子图；如果不止一个，则肯定构不成一个可归约子图。接着，串行算法将 t_i 能接收到的来自其前驱顶点的消息向其后继顶点转发，即检查消息传递矩阵的 t_i 行的 t_1 列到 t_{i-1}列各项是否不为 0，如果表项 $M[t_i][t_x](1\leqslant i\leqslant x)$不为 0，且不为 1，就将该值平均分割，然后添加到顶点 t_i 的各个后继顶点行的 t_x 列的表项内。最后，顶点自己产生一份消息，其值为 1，也将 1 平均分割，然后添加到顶点 t_i 的各个后继顶点列的 t_i 行表项内。这样，串行算法在一个顶点 t_i 上的操作都执行完毕，继续对下一个顶点执行相同的操作。在串行算法遍历了所有的顶点后，将检查消息传递矩阵的每一行是否有值为 1 的表项，选取其中最靠近有向无环图入口顶点的顶点 t_y，通过比较两个顶点所能接收到的产生于子图$\langle t_y,t_i\rangle$以外的顶点的消息的量是否相同来判断子图$\langle t_y,t_i\rangle$是否是一个可归约子图。如果是，就将值 y 填充到 $M[t_i][n+1]$内。

·算法形式化描述

算法 1 串行算法

输入：有向无环图 G(V,E)

输出：n×(n+1)的消息传递矩阵 M

```
1. // 用每个顶点能接收到消息的数量填充消息传递矩阵 M
2. for each vertex t_i in G(V,E) by topology order do
3.   d=|succ(t_i)|;
4.   //如果 t_i 只有一个后继顶点，t_i 和其后继顶点 succ(t_i) 也许能构
     造一个可归约子图
5.   if d==1 then
6.     if |pred(succ(t_i))|==1 then M[succ(t_i)][n+1]=i;
7.     //<t_i,succ(t_i)>不是可归约子图，令 M[succ(t_i)][t_i]的值在矩
       阵填充完时不为 1
8.     else M[succ(t_i)][t_i]=-1;
9.   end if
10.  //将 t_i 接收到的消息传给它的后继顶点
11.  for each t_j where j<i do
12.    if M[t_i][t_j]<1 then
13.      for each t_k where t_k∈succ(t_i) do
14.        M[t_k][t_j]+=M[t_i][t_j]/d;
15.        //如果 t_k 能接收到产生自 t_j 的完整消息，那么<t_j,t_k>可
           能是可归约子图
           //如果 t_k 此时能收到其他顶点的完整消息，则距离 t_k
           最远的顶点才有可能与 t_k 构成可归约子图
16.        if M[t_k][t_j]==1 and (M[t_k][n+1]==0 or M[t_k]
           [n+1]>j) then M[t_k][n+1]=j;
```

```
17.        end for
18.      end if
19.    end for
20. // t_i 将自己产生的一份消息平均切割发送给其后继顶点 succ(t_i)
21.    for each vertex t_k where t_k∈succ(t_i) do
22.      M[t_k][t_i]+=1/d;
23.    end for
24. end for
25. //如果 M [t_i][n+1]的值不为 0,通过比较两个顶点接收到的外
    部消息的数量来检查是否存在可归约子图
26. for each vertex t_i in G(V,E) by topology order do
27.   if M [t_i][n+1]!=1 and M [t_i][n+1]!=0 then
28.     for each t_j in G(V,E) not in ⟨t_M[t_i][n+1], t_i⟩ do
29.       if M [M [t_i][n+1]][t_j]!=0 and M [t_i][t_j]!=M [M [t_i]
          [n+1]][t_j] then
30.         M[t_i][n+1]=0;break;
31.       end if
32.     end for
33.   end if
34. end for
```

·算法分析

根据引理 1,只有由两个顶点组成的可归约子图可以在填充消息传递矩阵的过程中就被识别出来。由于在填充消息传递矩阵时遍历顶点的顺序是有向无环图中顶点的一个拓扑排序,因此每个顶点在开始传递消息给其后继顶点时,已经接收完了该顶点能够接收到的所有消息。一个顶点也就能在向其后继顶点开始传递消息前判断接收到的产生于哪些顶点的消息是完整的,不需要向后继顶点继续传输产生自这些顶点的消息。如果一个顶点 t_x 收到了产生于多个顶点的消息,根据定理 2,两个不可再分的可归约子图的出口顶点不可能重合,所以在所有的子图中,只有距离 t_x 最远的顶点 t_y 才可能与 t_x 构成一个可归约子图。为了更进一步确认该子图$\langle t_y, t_x\rangle$是否是可归约子图,利用定理 1,对比入口顶点和出口顶点接收到的产生自子图外部顶点(即那些不能接收到入口顶点消息的顶点)的消息。如果入口顶点和出口顶点接收到的产生于子图外部的顶点的消息的量完全一致,那么就可以确定该子图是一个可归约子图。表 1 所列是串行算法在图 1(b)上执行后得到的消息传递矩阵,为方便查看,所有值为 0 的表项都未显示。

表 1　消息传递矩阵

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	入口
t_1								
t_2	1/3							
t_3	1/3							
t_4	1/3							
t_5	2/3							
t_6	1/3		1					3
t_7	1							1

可以看到,顶点 t_i 最多可以接收产生于 $i-1$ 个顶点的消息,最多有 $n-i$ 个后继顶点,因此,该串行算法的时间复杂度是 $O(n^3)$。

4.2　并行算法

·算法思想

实际上,有向无环图中下一个顶点不必在前一个顶点发送完全部产生于其它顶点和自身的消息后,才能开始运行,所有顶点都可以同时接收和发送消息而不干扰其他节点的操作。而且各个顶点在接收完所有能够接收到的消息后,可以各自独立判断自身是否是一个可归约子图的出口顶点,从而找出所有的原子级可归约子图。

在本文提出的并行算法中,每个顶点都是用一个消息记录表 *Tab* 记录自身都能接收到产生于哪些顶点的消息,以及其消息完整程度如何。在消息记录表中,表项 $Tab[t_i]$ 表示自身顶点接收到的产生于顶点 t_i 的消息的完整程度。图 1(c)中顶点 t_2 和 t_5 的消息记录表如表 2 所列。在顶点间传递消息时,实际上是传递了一个元组 (t_x, val),它表示所传递的消息产生于顶点 t_x,其消息的完整程度是 val。当顶点 t_i 从它的前驱顶点接收到一个元组(t_j, val),顶点 t_i 将更新自身的消息记录表,将表项 $Tab[t_j]$现有的值和 val 相加,再存回表项 $Tab[t_j]$,并在转发给 k 个后继顶点之前,将表项中的 val 更新为 val/k,再将元组传递给所有 k 个后继顶点。与串行算法相同,如果表项的值为 1,该元组不会被顶点 t_i 转发给自身的后继顶点,所以后继顶点的消息记录表中表项 $Tab[t_j]$的值永远不可能为 1。如果顶点 t_i 的前驱顶点集合为空,意味着 t_i 已经接收完了所有能够接收到的消息,顶点 t_i 会把自身从所有后继顶点中的前驱集合里删去,并开始检查自身的消息记录表中是否有表项的值为 1,如果有,就用与串行算法相同的逻辑判断 t_i 自身是否是一个原子级可归约子图的出口顶点,进而探测出有向无环图中所有的原子级可归约子图。除了有向无环图的入口顶点 t_1 因为没有前驱顶点不需要转发其他顶点的消息外,其余顶点既转发其他顶点消息,自身也要产生消息并发送。在表 2 中,t_5 能接收到产生于 t_2, t_4, t_7 的完整消息,而 t_2 是其中距离入口顶点的最近的顶点,且 t_2 与 t_5 能接收到的产生于子图$\langle t_2, t_5\rangle$以外的顶点 t_1 的消息的量相同,所以子图$\langle t_2, t_5\rangle$是一个可归约子图。

表 2　顶点 t_2 和 t_5 的消息记录表

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	入口
t_2	1/2							
t_5	1/2	1		1			1	2

·算法形式化描述

算法 2　并行算法

输入:有向无环图 G(V,E),每个顶点的消息记录表 Tab

输出:每个顶点的消息记录表 Tab

```
1. //每个顶点 t_i 并行地用能接收到消息的数量填充各自的消息记录
   表 Tab
2. for i=1 to n par-do
3. Begin:
4.   d=|succ(t_i)|;
5.   for each t_j where t_j∈succ(t_i) do tuple (t_i,1/d) to t_j;
6.   repeat
7.     // 尝试从前驱顶点接收元组
8.     if receives a tuple (t_k,val) from t_h then Tab[t_k] =Tab[t_k k]+
       val;
9.       if Tab[t_k]!=1 then
```

```
10.          for each t_j where t_j∈succ(t_i) do tuple (t_k,val/d) to t_j;
11.        end if
12.    end if
13.  until pred(t_i) is null
14.  for each t_j where t_j∈succ(t_i) do remove t_i from pred(t_j);
15.  if Tab[t_1] ==1 then Tab[n+1]=1;
16.  for each vertex t_x in G(V,E) by topology order except t_1 and t_i
     do
17.    if Tab[t_x] ==1 then
18.      for each vertex t_y not in <t_x,t_i> by topology order do
19.        if Tab[t_y]≠1 and Tab[t_y]≠0 and Tab[t_y] of t_x≠Tab
           [t_y] then
20.          Tab[t_x]=0;break;
21.        end if
22.      end for
23.      if Tab[t_x] ==1 then Tab[n+1]=x;break;
24.    end if
25.  end for
26. End
27. end for
```

·算法分析

并行算法与串行算法搜索可归约子图的策略都相同。但是，由于并行算法可以多个顶点同时执行，而不是逐个执行，整个算法的运行时间自然缩短。有向无环图中的顶点 t_i 会接收到 $n-1$ 个产生于顶点 t_j 的元组，因此，顶点 t_i 最多会接收到最多 $(n-1)^2$ 个元组。因此并行算法的运行时间可控制在 $O(n^2)$。

5 实验

本文算法均使用 C++语言实现，实验环境为 Intel Core i3 3.10GHz CPU、2.0GB 内存。为了客观公正地比较文献[17]中提出的算法(本文称为原始算法)与本文提出的串行和并行算法的性能，本文使用随机方法产生嵌套图作为算法的输入，从而保证了实验数据的随机性。在随机产生嵌套图时，本文首先生成了特定数量的可归约子图，包含 2—10 个顶点，再将这些可归约子图与一些其他顶点按照严格的层次关系连接起来，构成一个包含特定的总顶点数的嵌套图。在相同嵌套图上分别运行 3 种算法，探索随着嵌套图总顶点数目的增加和图中可归约子图数目的增加，算法运行时间的变化情况。将算法运行时间定义为在同种条件下 20 次实验运行时间的平均值。

图 2 和图 3 展示了当嵌套图共有 1000 和 2000 个顶点时，对于不同的提前生成的可归约子图数目，文献[17]中的原始算法与串行算法、并行算法的运行时间的比较。随着嵌套图规模的增加，3 种算法的运行时间也在增加，原始算法的运行时间是串行算法运行时间的 3 倍左右，而本实验使用四核处理器，从图中可以看到并行算法的执行时间比串行算法的执行时间短，最好的情况下能达到串行算法运行时间的 1/3。另外，由于固定规模嵌套图中可归约子图数目增加，很多子图很早就被算法识别出来，因此每个顶点产生的消息的传播数量在逐渐减少，因此 3 种算法的运行时间也随之不断减少。

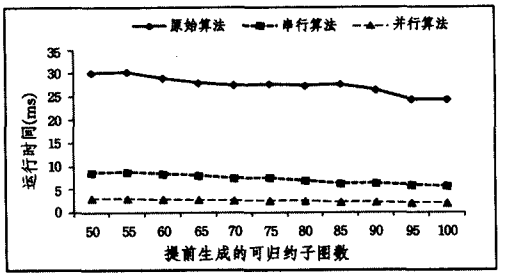


图 2 总顶点数为 1000 时，3 种算法的运行时间比较

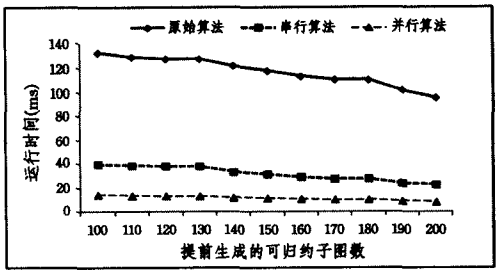


图 3 总顶点数为 2000 时，3 种算法的运行时间比较

结束语 随着能集成到单一芯片中的处理器数量快速增加，寻找一个从应用程序到片上网络的最优映射方案的计算复杂度也越来越大。为了缩短这一映射过程的计算时间同时不降低映射方案的质量，一个很重要的途径就是将应用程序对应的有向无环图的规模合理缩小。在文中，提出了在给定的任意有向无环图中搜索所有可归约子图的高效算法，具体可分为串行算法和并行算法两种。利用所找出的可归约子图，原有向无环图可以按照要求做任意粒度的收缩，而不破坏原图的依赖关系。算法的正确性通过对可归约子图特性的探索得到了证明。

参考文献

[1] Pasricha S, Dutt N. On-chip communication architectures: system on chip interconnect[M]. Morgan Kaufmann, 2010: 544

[2] Borkar S. Thousand core chips: a technology perspective[C]// Proceedings of the 44th annual Design Automation Conference. USA: ACM, 2007: 746-749

[3] Sylvester D, Keutzer K. A global wiring paradigm for deep submicron design[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2000, 19(2): 242-252

[4] Dally W J, Towles B. Route packets, not wires: On-chip interconnection networks[C]// Proceedings of 2001 Design Automation Conference. USA: IEEE, 2001: 684-689

[5] Flich J, Bertozzi D. Designing network on-chip architectures in the nanoscale era[M]. Chapman & Hall/CRC, 2010: 528

[6] Lei T, Kumar S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture[C]// Proceedings of Euromicro Symposium on Digital System Design (DSD 2003). USA: IEEE, 2003: 180-187

[7] Addo-Quaye C. Thermal-aware mapping and placement for 3-D NoC designs[C]// Proceedings of 2005 IEEE International SOC Conference. USA: IEEE, 2005: 25-28

[8] Hu J, Marculescu R. Energy-aware mapping for tile-based NoC

architectures under performance constraints[C]// Proceedings of the 2003 Asia and South Pacific Design Automation Conference. USA: ACM, 2003: 233-239

[9] Hu J, Marculescu R. Energy-and performance-aware mapping for regular NoC architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005, 24 (4): 551-562

[10] Hamedani P K, Hessabi S, Sarbazi-Azad H, et al. Exploration of temperature constraints for thermal aware mapping of 3D Networks on Chip[C]// Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). USA: IEEE, 2012: 499-506

[11] Murali S, De Micheli G. Bandwidth-constrained mapping of cores onto NoC architectures[C]// Proceedings of the conference on Design, automation and test in Europe-Volume 2. USA: IEEE, 2004: 20896

[12] Zhao K, Bian J. Instruction-level hardware/software partition through DFG exploration[C]// Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD). USA: IEEE, 2011: 55-60

[13] Bouchhima A, Gerin P, Pétrot F. Automatic instrumentation of embedded software for high level hardware/software co-simulation[C]// Proceedings of the Design Automation Conference Asia and South Pacific (ASP-DAC). USA: IEEE, 2009: 546-551

[14] Wang D, Li S, Dou Y. Collaborative hardware/software partition of coarse-grained reconfigurable system using evolutionary ant colony optimization[C]// Proceedings of the Design Automation Conference Asia and South Pacific (ASP-DAC). USA: IEEE, 2008: 679-684

[15] Srinivasan V, Govindarajan S, Vemuri R. Fine-grained and coarse-grained behavioral partitioning with effective utilization of memory and design space exploration for multi-FPGA architectures[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2001, 9(1): 140-158

[16] Wolf W. Object-oriented cosynthesis of distributed embedded systems[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 1996, 1(3): 301-314

[17] Henkel J, Ernst R. An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2001, 9(2): 273-289

[18] Chai S, Li Y, Wang J, et al. A List Simulated Annealing Algorithm for Task Scheduling on Network-on-Chip[J]. Journal of Computers, 2014, 9(1): 176-182

[19] Liu W, Gu Z, Xu J, et al. Satisfiability modulo graph theory for task mapping and scheduling on multiprocessor systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(8): 1382-1389

[20] Tahaee S A, Jahangir A H, Habibi-Masouleh H. Improving the performance of heuristic searches with judicious initial point selection[C]// Proceedings of the Fifth IEEE International Symposium on Embedded Computing. USA: IEEE, 2008: 14-19

[21] Hu J, Marculescu R. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints[C]// Proceedings of the Design, Automation and Test in Europe Conference and Exhibition. USA: IEEE, 2004: 234-239

[22] Vahid F. Partitioning sequential programs for CAD using a three-step approach[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2002, 7(3): 413-429

[23] Wiangtong T, Cheung P Y K, Luk W. Comparing three heuristic search methods for functional partitioning in hardware-software codesign[J]. Design Automation for Embedded Systems, 2002, 6 (4): 425-449

(上接第 51 页)

[5] Fred A, Jain A K. Data clustering using evidence accumulation [C]// Proceedings of the 17th International Conference on Pattern Recognition. 2002: 276-280

[6] Zhou Z H, Tang W. Cluster ensemble [J]. Knowledge-Based Systems, 2006, 19(1): 77-83

[7] Lu X Y, Yang Y, Wang H J. Selective clustering ensemble based on covariance[C]// Proceedings of the 11th International Workshop on Multiple Classifier Systems, 2013. LNCS, 2013, 7872: 179-189

[8] 罗会兰,孔繁胜,李一啸. 聚类集成中的差异性度量研究[J]. 计算机学报, 2007, 30(8): 1315-1324
Luo Hui-lan, Kong Fan-sheng, Li Yi-xiao. An Analysis of Diversity Measures in Clustering Ensembles[J]. Chinese Journal of Computers, 2007, 30(8): 1315-1324

[9] Krogh A, Vedelsby J. Neural network ensembles, cross validation, and active learning[C]// Proceedings of NIPS94-Neural Information Processing Systems: Natural and Synthetic, 1994. Advances in Neural Information Processing Systems 7, MIT Press, 1995: 231-238

[10] Lodwich A, Rangoni Y, Breuel T. Evaluation of robustness and performance of early stopping rules with multilayer perceptrons [C]// Proceedings of the 2009 International Joint Conference on Neural Networks. 2009: 1877-1884

[11] 杨燕,靳蕃,Kamelk M. 聚类有效性评价综述[J]. 计算机应用研究, 2008, 25(6): 1632-1638
Yang Yan, Jin Fan, Kamel M. Survey of clustering evaluation [J]. Application Research of Computers, 2008, 25(6): 1632-1638

[12] Pakira M K, Bandyopadhyay S, Maulik U. Validity index for crisp and fuzzy clusters [J]. Pattern Recognition, 2004, 37: 487-501

[13] Brualdi. Introductory Combinatorics, Fifth Edition[M]. Beijing: Prentice Hall, 2012