doi:10.11959/j.issn.1000-436x.2016123

基于图分割的流应用多处理器映射算法

唐麒，吴尚峰，施峻武，魏急波
（国防科技大学电子科学与工程学院，湖南 长沙 410073）

摘 要：为了充分利用多处理器平台所提供的计算资源，需要将应用以适当的方式映射到不同处理器，从而最大程度地挖掘应用所提供的并发性以满足应用严格的实时性要求。提出了并发图来量化、建模应用任务间的并发性，提出了一种基于自同步调度的并发图构建算法，并将任务映射问题转换成图分割问题，然后将并发图分割问题建模为纯 0-1 整数线性规划模型并采用 ILP 求解器获得最优解。采用了大量随机生成的同步数据流图以及一组实际应用对所提方法进行性能评估，实验结果表明所提方法性能优于已有算法。
关键词：同步数据流图；映射；多处理器；图分割
中图分类号：TP399 **文献标识码**：A

Graph partition based mapping algorithm on multiprocessors for streaming applications

TANG Qi, WU Shang-feng, SHI Jun-wu, WEI Ji-bo
(College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China）

Abstract: To take advantage of multiprocessor platform, it is a necessity to map tasks of the application properly onto different processors to exploit the concurrency in the application and thus meet the stringent timing requirements. Parallelism graph was proposed to quantify and model the concurrency among tasks of the application. An algorithm was also proposed to construct the parallelism graph based on the self-timed schedule and transform the mapping problem to a graph partitioning problem. The graph partitioning problem as a pure 0-1 integer linear programming model was further formulated and the ILP solver to find the optimal result. A lot of randomly generated synchronous dataflow graphs and a set of practical applications were used to evaluate the performance of the proposed method. The experimental results demonstrate that the proposed method outperforms available algorithms.
Key words: synchronous dataflow graph, mapping, multiprocessor, graph partition

1 引言

同步数据流图（SDFG，synchronous dataflow graph）广泛用于建模现代流应用，包括视频、音频编解码、软件无线电等。为了满足消费者对应用的质量要求，这些应用的计算复杂度日益增加，给硬件设计带来了巨大挑战。许多应用有严格的实时性要求，例如，系统输入与输出间的延时或系统吞吐量要满足特定指标。为了满足这些要求，一种直接的方法是增加处理器时钟频率，然而，随着时钟频率的提升，系统能耗急剧增加，降低了这一方法的实用性。另外，时钟频率正逐渐接近物理极限，依靠这一方法增加计算能力难以为继。多处理器平台提供了另一种方案，可以在计算能力与功耗间达到更好的平衡，因此，在嵌入式系统设计中得到了大量应用。尽管多处理器可以提供很高的计算能力，这并不意味着部署在其上的应用可以完全挖掘这些处理能力。实际上，如果应用只提供了极少量的并发性，那么采用多处理器几乎不能提高系统性能。正如 Amdahl 定

收稿日期：2015-10-14；**修回日期**：2016-05-11
基金项目：国家自然科学基金资助项目（No.61471376）
Foundation Item: The National Natural Science Foundation of China (No.61471376)

理所表明的那样，使用多处理器带来的性能增益严重受限于应用并发性。尽管很多应用，如由 SDFG 建模的流应用，可以并行执行，如何在多处理器上利用应用的内在并发性仍然是个有待解决的问题，这就是所谓的并行调度问题。

应用调度包括任务映射、排序和定时[2]。任务映射是将任务分配到不同处理器上的过程。对于最优调度，无论最优性的评价标准是什么，总存在对应的最优映射，因此研究如何获得最优映射具有重要价值。对于映射问题，研究者提出了很多针对不同应用模型和平台的算法。其中，图分割法[3]和负载均衡法[4,5]得到了大量研究。然而，这 2 类算法要么在最小化处理器间通信的前提下最大化负载均衡水平，要么单纯地均衡不同处理器上的负载，而没有综合考虑应用的内在结构，因此降低了性能。另一类方法是链式调度算法[6,7]，这类算法将任务调度分成优先级分配和调度 2 个步骤，然而，这类算法主要用于优化系统延时，虽然可以通过修正应用于吞吐量优化，但性能并不是很好。

本文研究 SDFG 在多处理器上的映射问题，以获得可以最大化应用并发性的最优任务映射，从而使应用长期吞吐量最大化。本文提出了并发图（PG，parallelism graph）来量化和建模 SDFG 中不同任务间的并发性，提出了一种基于自同步调度的并发图构建算法，并将映射问题转换化成图分割问题，然后将图分割问题建模为纯 0-1 整数线性规划模型（ILP，integer linear programming），并使用 ILP 求解器获取最优解。

本文所提算法在 SDF3[8]中实现，并采用随机生成的 SDFG 和一组实际应用进行性能评估。算法的有效性通过与已有负载均衡算法[4,5]和 HEFT 算法[6]比较得以验证。

在余下部分使用下述记号。$\mathbf{Z}$、$\mathbf{Z}^+$ 和 $\mathbf{Z}_0^+$ 分别表示整数、正整数和非负整数集。使用黑体大写字母表示向量，并使用相应的斜体小写字母表示其中元素。对一个向量或集合，使用$|\cdot|$表示其中元素数目。

2 相关工作

图分割[3]被证明可以用于并行计算。图分割的主要目标是将计算均匀地分配到多个处理器，同时最小化处理器间通信[9]。然而，算法中没有考虑任务间的依赖关系。与此不同，本文提出并发图的概念，利用任务间的依赖关系，建模任务间并行执行关系，从而将问题转换成图分割问题。文献[4,5]将负载均衡思想应用于 SDFG 映射问题，所提算法考虑了计算、通信带宽及内存消耗的均衡，任务绑定按照优先级非递增的顺序进行，任务优先级采用估计最大均值（MCM，maximum cycle mean）[4]来计算。然而，算法没有充分利用任务间的依赖关系，不能最大化应用并发性，降低了算法性能。

文献[6,7]研究了如何利用应用结构特征，包括任务间依赖关系、任务执行时间及数据传输需求来提高调度性能。然而，这些方法针对由有向无环图（DAG，directed acyclic graph）建模的应用，不能直接应用于 SDFG。与 DAG 不同，SDFG 可以支持任务间环状依赖关系和多速率依赖，因此，任务间关系更为复杂。这种关系不能简单地由模型中的路径特征，比如路径长度来描述。一种可行的方法是将 SDFG 转换成同构 SDFG，并进一步将同构 SDFG 转换为无环依赖图[2]。由于无环依赖图属于 DAG，因此上面提到的调度算法可以直接使用。然而，这种方法主要用于允许任务复制的系统，为了适应无任务复制的系统，可以在调度时强制将同一任务的实例分配到同一处理器。

3 应用模型

本文采用 SDFG 来建模流应用，如软件无线电、通信协议、多媒体应用等。本文所采用 SDFG 模型的定义如下。

定义 1 （SDFG）同步数据流图是一种有向图，由二元组 $G=(V,E)$ 表示，其中，V 为节点或顶点的有限集合，每个节点表示一个应用任务或角色；E 表示有向边的有限集合，每条边表示被连接任务间的依赖关系或通信需求。每个节点 $v\in V$ 分配了一个属性 $c(v)$，是节点的代价函数，表示执行一次任务所需要的时钟周期。每条边 $e\in E$ 用一个五元组 $(src,p,dst,q,iniTok)$ 表示，其中，src 表示边的源任务，p 表示数据产出速率，dst 表示边的宿任务，q 表示数据消费速率，$iniTok$ 为边上的初始数据数目（数据单位为符号）。对于边 e，采用记号 $src(e)$、$p(e)$ 等表示边上的相应元素。当源任务 $src(e)$ 结束执行时，向边输出 $p(e)$ 个符号的数据，当宿任务 $dst(e)$ 开始执行时，从边上消费 $q(e)$ 个符号的数据。本文将边 e 称为任务 $src(e)$ 的输出边及任务 $dst(e)$ 的输入边。

SDFG 在执行过程中，任务将以不同频率执行，

因此在一次调度中会出现不同次数。SDFG 迭代及重复向量可以用于来描述 SDFG 的这一特征。

定义 2　（SDFG 迭代）SDFG 迭代是执行每个任务最小正整数次数同时使各边符号数目恢复初始值的过程。

定义 3　（重复向量）假定 SDFG 有 n 个任务，各任务从 0 到 $n-1$ 编号，其重复向量 $\boldsymbol{R}$ 是一个长度为 n 的列向量，其中，第 k 个元素等于任务 k 在一次迭代中出现的次数。对任务 v，用 $r(v)$ 表示该任务在重复向量中的相应数值。

SDFG 的重复向量可以通过解平衡方程而得[1]。当平衡方程有非零解[1]时，重复向量 $\boldsymbol{R}$ 存在，且这样的 SDFG 总可以转换成等价的 HSDFG[2]。

本文只考虑强连通 SDFG，其中每个任务都与其他任意任务直接或间接相连。对实际应用，这是一个合理的假定，因为实际系统内存消耗是受限的，因此 SDFG 中任意边的最大符号数目是有限的。通过为 SDFG 中的每条边增加额外的最大符号数目约束边，可以将非强连通 SDFG 转换成强连通图。

图 1 所示为一个由 5 个任务组成的 SDFG 示例。图中各边末端数字为边的产出或消费速率，为简单起见，只有当速率大于 1 时才予以标注。边附近的文字为边的标签，标签后面括号中的数字表示边上的初始符号数，如果初始符号数为 0，则予以忽略。图中 SDFG 的重复向量为 $[2,2,3,1,1]^{\mathrm{T}}$，表示在一次应用迭代中，任务 a_0、a_1、a_2、a_3 和 a_4 分别需要执行 2、2、3、1 和 1 次。

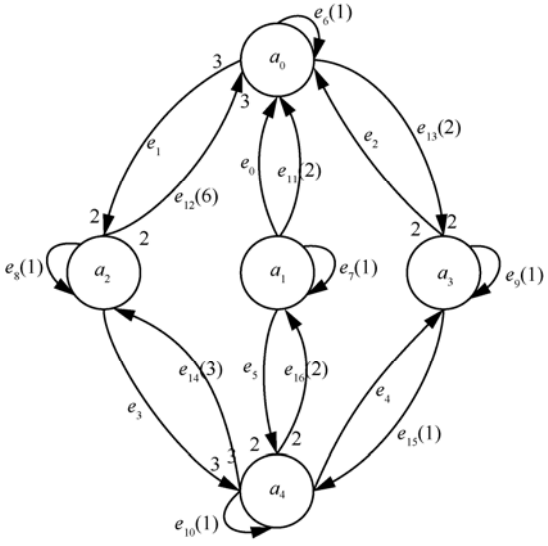


图 1　SDFG 结构示例

4　问题阐述

吞吐量是流应用最为重要的性能度量标准之一。本文研究如何将流应用映射到多处理器从而最大化应用的长期吞吐量。根据是否允许任务复制，映射问题可以分成基于复制的映射和无复制的映射。本文考虑不允许任务复制的应用映射问题，这意味着每个任务只能分配到一个处理器上执行。在本文所考虑的问题中，应用采用 SDFG 建模。在 SDFG 中，应用任务的并发执行能力并未明显建模。尽管应用的最大吞吐量可以通过自同步调度[2]获得，这一调度并未将资源约束，例如处理器数目纳入考虑之中。实际多处理器平台中处理器数目通常少于 SDFG 中最大可并发执行的任务数目，因此，如何在映射中尽可能地挖掘 SDFG 任务间并发性以提高系统吞吐量是一个极为重要的问题。

对于 SDFG，尽管可以从图模型结构推导阻止任务并发执行的优先约束关系，任务间并发性并未明显地建模出来，例如，2 个任务在多大程度上可以并行执行并未在应用图中得到描述。除此之外，也不能直接将任务间优先关系与任务间并行度关联起来，使在映射中挖掘任务并发性变得极为困难。尽管如此，如果可以量化任务间并发程度，则映射问题可以次优地转换成最大化任务间并发性的问题。基于以上观察，本文从提取、量化 SDFG 任务间并发性着手，并使用图分割法解决应用映射问题。

5　周期静态调度与吞吐量分析

由于最终目标是获得吞吐量最优的映射，分析映射的吞吐量显得极为重要。本文假定系统根据静态周期调度循环执行，静态周期调度强制任务按照指定顺序依次执行。不同迭代可以在时间上重叠，采用流水线的方式执行。本文采用最早开始时间调度算法为每个处理器构建周期静态顺序调度。当同一时刻有多个任务可以同时执行时，任意一个任务被选择优先执行。为每个处理器构建好周期静态调度后，应用吞吐量可以通过已有方法来计算。

文献[12]提出了一种无资源约束下的基于状态空间搜索的 SDFG 吞吐量分析方法。这一方法采用 SDFG 边上符号分布及任务执行状态来表征应用状态。由于所考虑 SDFG 是强连通的，并且状态中各元素值的范围是有限的，因此，状态空间也是有限

的。经过有限次数的状态转移后，将重新遇到已经经历过的状态，从而形成状态环。获得状态空间中的状态环后，吞吐量可以直接计算出来。另一种计算吞吐量的方法是用 max-plus 代数[10]建模应用的自同步执行，max-plus 矩阵的特征值的倒数即等于应用吞吐量。上述吞吐量计算方法没有考虑映射和调度，只适应于无资源约束的场景。文献[11,13]提出了将周期静态顺序调度建模到 HSDFG/SDFG 中的方法，通过对 SDFG 进行调度扩展，可以用文献[10,12]中的吞吐量分析方法进行精确的吞吐量分析。本文使用文献[11,12]中的方法进行调度性能分析。

6 并发图及其构建

本节介绍并发图的概念及其构建方法。并发图用于量化和建模 SDFG 任务间的并发性，其定义如下。

定义 4 (并发图)对一个 SDFG $G=(V,E)$，相应的并发图是一个二元组 $PG=(V',E')$。并发图是无向加权图，每个顶点对应于 SDFG 中的一个任务，边权表示所连接任务间的并发程度，边权为正整数。

本文通过 SDFG 的 ASAP（as-soon-as-possible）调度（自同步调度[2]）的周期扩展来计算任务间并发度。在 ASAP 调度中，当任务所依赖数据就序时，任务即开始执行。ASAP 调度定义如下。

定义 5 (ASAP 调度)一个 SDFG $G=(V,E)$ 的 ASAP 调度是一个映射 $s:(v_i,k_i)\to\mathbf{Z}_0^+$，表示任务 v_i 的第 k_i 个实例的开始执行时间，其中 $v_i\in V$，$k_i\in\mathbf{Z}^+$。任务开始执行时间满足

$$s(v_i,k_i)=\begin{cases}0, & \neg\exists e_{j,i}\in E \text{ 或 } dep_{ij}\leqslant 0\\ \max\limits_{v_j\in V,e_{j,i}\in E}\{s(v_j,dep_{ij})+c(v_j)\}, & \text{其他}\end{cases}\tag{1}$$

其中，$e_{j,i}$ 表示从任务 v_j 到任务 v_i 的边，dep_{ij} 表示任务 v_i 的第 k_i 个实例依赖于任务 v_j 的第 dep_{ij} 个实例，其定义如下

$$dep_{ij}=\left\lceil\frac{k_i q(e_{j,i})-d(e_{j,i})}{p(e_{j,i})}\right\rceil$$

ASAP 调度由瞬态和紧随其后的周期态组成，其定义如下。

定义 6 (瞬态)ASAP 调度 $s:(v_i,k_i)\to\mathbf{Z}_0^+$ 的瞬态是 ASAP 调度进入周期状态前的一系列任务执行过程。

定义 7 (周期态)ASAP 调度 $s:(v_i,k_i)\to\mathbf{Z}_0^+$ 的周期态由三元组 (Q,I,T) 表示，其中，Q 有 $|V|$ 个元素，第 i 个元素 q_i 表示任务 v_i 在瞬态中执行的次数；I 表示周期态中应用的最小迭代次数；T 表示周期态的周期。ASAP 调度的周期态中，各任务的开始时间需要满足

$$s(v_i,k_i+Ir(v_i))=s(v_i,k_i)+T,\quad k_i\geqslant q_i$$

应用吞吐量由 ASAP 调度的周期态决定，因此使用它来计算任务间并发度。尽管周期态中一个周期可能包含多次应用迭代，如周期态定义中的 I 所指明的那样，无论最后计算出的任务间并发度是否除以这一数值，并不会影响最后结果。由于周期态具有周期特征，因此并不需要构建完整的 ASAP 调度，可以在获得一个完整的执行周期后停止执行。本文从周期态中提取一个执行周期，对其进行周期扩展，并根据这个周期扩展中不同任务的并行执行时间来量化任务间并发度。

定义 8 (周期扩展) ASAP 调度的周期扩展是一个映射 $s_{\text{pe}}:(v_i,k_i)\to\mathbf{Z}_0^+$，表示任务 v_i 的第 k_i 个实例的开始执行时间，其中，$v_i\in V$，$k_i'\in\mathbf{Z}$。任务开始执行时间满足

$$s_{\text{pe}}(v_i,k_i'+iterIr(v_i))=s(v_i,k_i)+iterT\tag{2}$$

其中，$k_i'\in[q_i,q_i+Ir(v_i)]$，$iter\in\mathbf{Z}$。

图 1 所示 SDFG 的 ASAP 调度如图 2 所示，ASAP 调度由瞬态和周期态 $([2,1,1,0,0]^{\mathrm{T}},1,35)$ 组成。周期态从时刻 18 开始；在时刻 53，再次经历时刻 18 经历的状态。因此时刻 18 和 53 之间的执行形成一个周期，其中包含一次应用迭代。

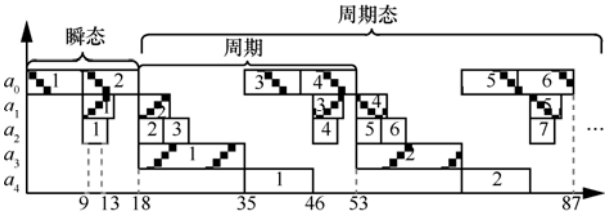


图 2 SDFG 的 ASAP 调度示例

并发图构建算法详细描述了构建并发图的步骤。对一个 SDFG，首先为其增加自边以约束任务自动并发，也即同一时间一个任务最多只能有一个实例处于运行状态。然后，初始化并发图，为其添

加节点。并发图中的每个节点对应 SDFG 中的一个任务。其次，反复调用方程(1)构建 SDFG 的 ASAP 调度。在此过程中，提取 ASAP 调度的一个执行周期，并对其进行周期扩展。根据得到的周期扩展，计算 PG 中任意 2 个节点间的边权，边权等于对应任务在周期扩展中的重叠时间。

并发图构建算法

输入　应用模型 $G=(V,E)$

输出　并发图 $PG=(V',E')$

1) 对 SDFG 中的每个任务 v，增加自边 $(v,1,v,1,1)$

2) 在集合 V' 中增加顶点 $v'_0,v'_1,\mathrm{L},v'_{|V|-1}$，顶点 v'_i 对应于 $v_i\in V$

3) 构建 ASAP 调度 $s:(v_i,k_i)\to\mathbf{Z}_0^+$

4) 获得周期状态 (Q,I,T)

5) 构建周期扩展 $s_{\mathrm{pe}}:(v_i,k_i)\to$ ¢ $_0^+$

6) for $i=0:|V|-2$ do

7)　　for　$j=i+1:|V|-1$　do

8)　　初始化 $w(i,j)=0$

9)　　for 周期扩展中任务 i 和 j 的实例 n 和 m do

10)　　将重叠时间加到 $w(i,j)$

11)　　end for

12)　　if $w(i,j)>0$ then

13)　　在 E' 中增加边 e_{ij}

14)　　设置边 e_{ij} 的边权为 $w(i,j)$

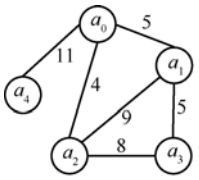
15)　　end if

16)　end for

17) end for

如算法中第 6)~17)行所示，并发图中 2 个节点间边的边权通过累加对应任务实例的并行执行时间而得。如果计算得到的边权非零，则在并发图中为相应节点增加一条边，边权设置为上面计算得到的值。

图 3 为示例 SDFG 的并发图及其邻接矩阵。如图 2 所示，不同执行周期在时间上不重叠，例如，第 1 个周期开始于时刻 18 而结束于时刻 53，第 2 个周期开始于时刻 53 而结束于时刻 87。因此，构建并发图时，并不需要进行周期扩展。在图 2 所示 ASAP 调度的第一个周期中，任务实例 a_0^4（上标表示实例索引）与 a_1^3 重叠，重叠时间为 5，因此，在并发图中，任务 a_0 和 a_1 间有一条边，边权为 5。类似地，可以为并发图添加其他边。



(a) 并发图

0	5	4	0	11
5	0	9	5	0
4	9	0	8	0
0	5	8	0	0
11	0	0	0	0

(b) 邻接矩阵

图 3　示例 SDFG 并发图及其邻接矩阵

在并发图中，由于边权均为正整数，因此每一条边均表示该边所连接任务在一定程度上可以并发执行，边权记录了对应任务可并发执行的程度。如果在调度中有尽可能多的任务可以并发执行，则应用的一次迭代可以在更少的时间内完成，因而提高应用吞吐量。因此，以最大化吞吐量为目标的映射问题，在一定程度上可以等价于最大化映射的并发度。通过使用并发图，任务间并发性得以量化，映射问题可以等价于分割并发图并使割最大化。割定义为所连接任务被分配到不同处理器的边的边权之和。采用上述方法，可以将映射问题转化成图分割问题。

7　纯 0-1 整数线性规划模型

本节将图分割问题建模为纯 0-1 ILP 模型。使用 N、M、***adj*** 分别表示任务数目、处理器数目和并发图的邻接矩阵。使用二进制映射变量 $x_{i,k}$ 表示任务 i 是否映射到处理器 k。同时使用了二进制辅助变量 $y_{i,j},z_{i,j,k}$。如果任务 i 和 j 分配到不同处理器，则 $y_{i,j}$ 等于 0，否则为 1。当 $x_{i,k}+x_{j,k}>0$ 时 $z_{i,j,k}$ 等于 0，否则等于 1。具体模型如式(3)~式(8)所示。

$$\max\ cut=\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}adj_{i,j}y_{i,j} \tag{3}$$

$$\text{s.t.}\sum_{k=0}^{M-1}x_{i,k}=1 \tag{4}$$

$$x_{i,k}+x_{j,k}+2z_{i,j,k}\leqslant 2 \tag{5}$$

$$x_{i,k}+x_{j,k}+y_{i,j}+2z_{i,j,k}\geqslant 2 \tag{6}$$

$$x_{i,k}+x_{j,k}+y_{i,j}-2z_{i,j,k}\leqslant 2 \tag{7}$$

$$x_{i,k},x_{j,k},y_{i,j},z_{i,j,k}=0\text{或}1 \tag{8}$$

其中，$i,j,k\in\mathbf{Z}_0^+,i,j\in[0,N-1],k\in[0,M-1]$，式(3)~式(8)对任意 $i\neq j,k$ 均成立。

由于每个任务只能映射到一个处理器，因此必

须满足如式(4)所示的任务映射约束。由于引入了辅助变量 $y_{i,j}$ 使如式(3)所示目标函数线性化，需要引入约束式(4)~式(8)。这些约束方程的推导如下。

当任务 i 和 j 分配到同一个处理器时辅助变量 $y_{i,j}$ 等于1，否则等于 0。这一约束等价于如下约束：对任意 k，如果 $x_{i,k}+x_{j,k}=1$，则 $y_{i,j}=1$；如果 $x_{i,k}+x_{j,k}=2$，则 $y_{i,j}=0$。因为 $x_{i,k}$ 和 $x_{j,k}$ 均为二进制变量，可以用如下约束替代上一约束：对任意 k，如果 $x_{i,k}+x_{j,k}>0$，则 $x_{i,k}+x_{j,k}+y_{i,j}-2=0$。上面的约束为非线性 If-Then 约束，引用辅助变量 $z_{i,j,k}$，可以用如下线性约束替代[14]。

$$x_{i,k}+x_{j,k}-1\leqslant L(1-z_{i,j,k}) \tag{9}$$

$$x_{i,k}+x_{j,k}+y_{i,j}-2\geqslant -Lz_{i,j,k} \tag{10}$$

$$x_{i,k}+x_{j,k}+y_{i,j}-2\leqslant Lz_{i,j,k} \tag{11}$$

$$x_{i,k},x_{j,k},y_{i,j,k}=0\text{或}1 \tag{12}$$

因为 x 和 y 均是二进制变量，因此 $x_{i,k}+x_{j,k}-1$ 的下界为–1，而 $x_{i,k}+x_{j,k}+y_{i,j}-2$ 的下界和上界分别为–2 和1。令 $L=2$，由式(9)~式(12)可得到式(4)~式(8)。

8　实验与结果

本节通过实验来评估所提算法的性能，并将本文所提算法与负载均衡算法[4]及 HEFT 算法[6]进行了比较。

8.1　度量参数

吞吐量是流应用系统的关键优化目标，如同文献[4,5]一样，本文采用吞吐量对算法性能进行评估。吞吐量采用第 5 节所介绍的方法计算。同时，在实验中分析了不同映射算法所获得映射方案的割值及算法时间开销。割值大小反映了映射所挖掘的并发度，定义为跨越不同子集的边的边权之和。

8.2　测试基准

本文使用一组随机生成的 SDFG 及一组实际应用进行性能测试，并使用开源工具 SDF3[8]来产生随机 SDFG，其中任务数为 5 到 15。对每个图尺寸，均生成个 100 个随机 SDFG。SDFG 参数包括任务执行时间和边上的符号大小，均随机生成。任务累积执行时间在 400 和 1 000 间均匀产生。

8.3　实验结果

表 1 所示为随机应用在具有不同处理器数目的多处理器上的测试结果。表中第 1 列和第 2 列分别表示随机 SDFG 的任务数目和平台处理器数目。“吞吐量”、“割值”、“时间复杂度”列中所有数值均进行了归一化。从表中可以看出，对不同多处理器和有不同任务数的 SDFG，本文所提算法从吞吐量的角度看要优于其他 2 种算法，吞吐量平均增加 17.40%和 18.23%。这一性能增益表明最大化并发度的思想有助于提升映射的吞吐量。由于采用并发图来表示应用并发性，并采用图分割方法最大化映射的割值，采用本文所提算法所获得的映射在并发度的数值上要大于负载均衡算法和 HEFT。如“割值”列中数据所示，在所有测试中，所提出的算法所生成的映射的割值比负载均衡算法大 10%，比 HEFT 高 21.95%。采用本文所提算法的时间复杂度高于其他 2 种算法。当任务数目较小的，时间复杂度是负载均衡算法的数倍；而当任务数较大时，复杂度急剧增加，这表明所提算法可较好地应用于小规模问题。

表 2 所示为实际应用在多处理器上的测试结果。表中数据表明本文所提算法对实际应用仍然有

表 1　　**随机应用性能对比**

任务数	处理器数	吞吐量			割值			时间复杂度		
		ILP	HEFT	LB	ILP	HEFT	LB	ILP	HEFT	LB
5	2	1.066	0.987	1	1.041	0.849	1	5.902	0.905	1
	4	1.198	0.719	1	1.024	0.641	1	4.057	0.876	1
10	2	1.120	1.077	1	1.143	1.016	1	9.765	1.109	1
	4	1.305	0.949	1	1.095	0.845	1	129.6	0.812	1
15	2	1.092	1.098	1	1.164	1.080	1	10.304	0.738	1
	4	1.263	1.128	1	1.132	0.978	1	3 006	0.604	1
平均值		1.174	0.993	1	1.100	0.902	1	527.6	0.841	1

表 2　　实际应用性能对比

应用	吞吐量			割值			时间复杂度		
	ILP	HEFT	LB	ILP	HEFT	LB	ILP	HEFT	LB
MP3 解码器 1	1.076	0.951	1	1.002	0.980	1	35.44	0.383	1
MP3 解码器 2	1.160	0.922	1	1.187	0.923	1	1.037	1.071	1
调制解调器	1.096	0.924	1	1.186	0.820	1	1.647	0.470	1
信道均衡器	1.026	0.908	1	1.668	0.648	1	0.571	0.043	1
卫星接收器	1.014	1.069	1	1.135	1.122	1	0.981	1.000	1

效，吞吐量要优于负载均衡算法和 HEFT。对调制解调器，使用所提算法与负载均衡算法和 HEFT 相比可以提高 9.6%和 18.6%的吞吐量。对其他应用，也可取得一定的性能增益。与随机应用一样，使用本文算法可以提高映射割值，再次验证了映射割值与吞吐量之间的正相关关系，表明采用割值来优化映射问题要优于其他方法。在时间复杂度方面，除 MP3 解码器 1 外，所提算法与负载均衡算法相当，略高于 HEFT，表明所提算法可较好地应用于实际应用。

9　结束语

本文研究了 SDFG 在多处理器上的映射问题，以最大化应用吞吐量为目标。提出了并发图来量化、建模应用并发性，提出了一种基于自同步调度的并发图构建算法，并将映射问题转换为图分割问题，最后将图分割问题建模为纯 0-1 整数线性规划问题。本文采用了一组实际应用和随机产生的 SDFG 进行了算法性能评估。实验结果表明本文所提算法的吞吐量性能要优于负载均衡算法和 HEFT 算法，证明采用并发图及图分割的方法求解映射问题优于已有方法。

参考文献：

[1] LEE E A, MESSERSCHMITT D G. Static scheduling of synchronous data flow programs for digital signal processing[J]. IEEE Transactions on Computers, 1987, 100(1): 24-35.

[2] SRIRAM S, BHATTACHARYYA S S. Embedded multiprocessors: scheduling and synchronization[M]. CRC Press, 2012.

[3] AUBANEL E. Resource-aware load balancing of parallel applications[M]// Handbook of research on grid technologies and utility computing: concepts for managing large-scale applications. 2009: 12-21.

[4] STUIJK S, BASTEN T, GEILEN M, et al. Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs[C]// The 44th Annual Design Automation Conference. c2007: 777-782.

[5] AMBROSE J A, NAWINNE I, PARAMESWARAN S. Latency-constrained binding of dataflow graphs to energy conscious GALS-based MPSoCs[C]//IEEE International Symposium on Circuits and System. c2013: 1212-1215.

[6] TOPCUOGLU H, HARIRI S, WU M. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13 (3): 260-274.

[7] SINNEN O. Task scheduling for parallel systems[M]. John Wiley & Sons, 2007.

[8] STUIJK S, GEILEN M, BASTEN T. SDF3: SDF for free[C]// Conference on Application of Concurrency to System Design. c2006: 276-278.

[9] HENDRICKSON B, KOLDA T G. Graph partitioning models for parallel computing[J]. Parallel Computing, 2000, 26 (12): 1519-1534.

[10] GEILEN M. Synchronous dataflow scenarios[J]. ACM Transactions on Embedded Computing Systems, 2010, 10 (2): 16.

[11] DAMAVANDPEYMA M, STUIJK S, BASTEN T, et al. Schedule-extended synchronous dataflow graphs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32 (10): 1495-1508.

[12] GHAMARIAN A H, GEILEN M, STUIJK S, et al. Throughput analysis of synchronous data flow graphs[C]//Sixth International Conference on Application of Concurrency to System Design. c2006: 25-36.

[13] BAMBHA N, KIANZAD V, KHANDELIA M, et al. Intermediate representations for design automation of multiprocessor DSP systems[J]. Design Automation for Embedded Systems, 2002, 7 (4): 307-323.

[14] WINSTON W L, GOLDBERG J B. Operations research: applications and algorithms[M]. Duxbury Press, 2004.

作者简介：

唐麒（1986-），男，湖南益阳人，国防科技大学博士生，主要研究方向为软件无线电技术和嵌入式并行计算。

吴尚峰（1986-），男，重庆人，国防科技大学博士生，主要研究方向为软件无线电技术。

施峻武（1977-），男，云南曲靖人，国防科技大学讲师，主要研究方向为软件无线电技术。

魏急波（1967-），男，湖南汉川人，国防科技大学教授、博士生导师，主要研究方向为通信信号处理与通信网络。