一种粗粒度可重构体系结构多目标优化映射算法

陈乃金[1,3],江建慧[2]
(1. 安徽工程大学计算机与信息学院,安徽芜湖 241000;2. 同济大学软件学院,上海 201804;
3. 天津大学计算机科学与技术学院 天津 300072)

摘 要: 针对多约束下的行流水粗粒度可重构体系结构的硬件任务划分映射问题,提出了一种多目标优化映射算法.该算法根据运算节点执行时延、依赖度等因素构造了累加概率权值函数,在满足可重构单元面积和互连等约束下,通过该函数值动态调整就绪节点的映射调度次序,当一块可重构单元阵列当前行映射完毕后,就自动换行,当一块阵列被填满,就切换到下一块,当一个数据流图映射完毕后,就自动计算划分块数等参数.实验结果表明,与层贪婪映射算法相比,文中算法平均执行总周期降低了 8.4%($RCA_{4\times4}$)和 5.3%($RCA_{6\times6}$),与分裂压缩内核映射算法相比,文中算法平均执行总周期降低了 20.6%($RCA_{4\times4}$)和 21.0%($RCA_{6\times6}$),从而验证了文中提出算法的有效性.

关键词: 可重构单元阵列;时域映射;累加概率权值;异步计算时延;资源约束
中图分类号: TP302 **文献标识码:** A **文章编号:** 0372-2112 (2015)11-2151-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.11.003

A Multi-Objective Optimization Mapping Algorithm for Coarse Grained Reconfigurable Architectures

CHEN Nai-jin[1,3], JIANG Jian-hui[2]
(1. *College of Computer and Information Science, Anhui Polytechnic University, Wuhu, Anhui* 241000, *China*;
2. *School of Software Engineering, Tongji University, Shanghai* 201804, *China*;
3. *School of Computer Science and Technology, Tianjin University, Tianjin* 300072, *China*)

Abstract: Based on row pipelining coarse grained reconfigurable architecture (CGRA), we presented MOM (multi-objective optimization mapping) algorithm to solve the multi-constraints hardware task partitioning-mapping problem. The cumulative probability weight function was constructed by the execution delay of computing nodes and the dependence between two nodes, etc. With the constraints of reconfigurable cell area and interconnection, the proposed algorithm could adjust dynamically the scheduling order of the ready nodes by thefunction values. When a row of the RCA was mapped completely, MOM began on a new row. When the RCA was filled, MOM switched to the next one. When a DFG (data flow graph) was mapped completely, the number of modules and etc were calculated automatically in MOM. Experiment results show that the average execution total cycles of MOM decrease by 8.4%($RCA_{4\times4}$) and 5.3%($RCA_{6\times6}$) comparing with LBGM (level based greedy mapping) algorithm. Comparing with SPKM (split-push kernel mapping) algorithm, the average execution total cycles of MOM decrease by 20.6%($RCA_{4\times4}$) and 21%($RCA_{6\times6}$). Experimental evaluation confirms the efficiency of our approach.

Key words: reconfigurable cell array; temporal mapping; accumulation probability weight; asynchronous computation delay; resource constraint

1 引言

随着高性能计算的迅速发展,可重构计算越来越受到人们的关注,粗粒度可重构体系结构 CGRA (coarse grained reconfigurable architectures)具有高的加速比、灵活的可编程性等特点,已经在很多领域得以广泛的应用[1~3].CGRA 一直是国内外学术界的研究热点,相关的研究成果较多,如文献[4~16]等.

收稿日期:2015-02-28;修回日期:2015-06-05;责任编辑:马兰英
基金项目:国家 863 高技术研究发展计划(No.2009AA011705,No.2013AA013204);国家自然科学基金重点项目(No.61432017);安徽省自然科学基金(No.1408085MF124);芜湖市科技计划自然科学基金(No.芜科计字[2012]94 号);安徽工程大学国家自然科学预研基金;安徽省高校省级自然科学基金重点项目(No.Kj2015A003)

一般而言,CGRA 是由精简指令集主处理器、主存储器、DMA 控制器、AHB、RPU(Reconfigurable Processing Unit)等构成,RPU 是 CGRA 核心组件之一,RPU 是由若干个寄存器(即控制、数据、可配置寄存器等)、局部存储器、一块或多块可重构单元阵列 RCA(Reconfigurable Cell Array)等构成,其中,RCA 是由一组同或异构可重构单元 RC(Reconfigurable Cell)或称为 PE (Processing Element)通过不同的互连方式构成的 RCA(或称为 PEA (Processing Element Array)),RCA 可以完成多种简单或复杂的算术逻辑计算及数据路由传输等功能.音视频编解码等计算密集型任务或其关键循环占用了大量的运行时间,若将计算密集型任务或循环转换为中间表示如数据流图 DFG (Data Flow Graph)等,通过某种划分映射算法将其分割并映射到一块或多块 RCA 上去分时复用执行,这将大大提高计算密集型任务的计算性能,由于 RCA 具有通信机制、互连方式、硬件面积等多个约束,这使得 DFG 的运算节点很难被划分和调度映射到 RCA 去,但是只有计算密集型任务或其关键循环 DFG 被划分并映射到 RCA 上,并通过正确的配置,才能实现 CGRA 运算的高性能,故 DFG 的划分映射问题是 CGRA 编译工具要解决的瓶颈问题,其也是一个 NP 完全问题.

针对不同互连方式的 RCA,相应的划分映射方法被相继提出[7~11],但是当前已有研究存在以下方面不足:(1)RCA 运算性能的优化问题考虑不足,如计算和互连延迟等;(2)针对 RCA 的划分映射没有进行多个目标优化或考虑.针对已有工作的不足,本文对此进行了深入研究,本文的主要贡献列举如下:

(1)考虑映射成功 RCA 块内节点的异步计算延迟和互连延迟,对 CGRA 流水化映射问题进行了形式化定义.

(2)提出了一种面向行流水 CGRA 的多目标优化映射算法,该算法综合考虑了 RCA 并行度、重复使用次数、配置成本等多个指标.

2 相关工作及与本文工作的区别与联系

2.1 相关工作

目前,国内外科研院所关于 CGRA 的研究较为活跃,主要表现为两个方面:(1)CGRA 存储和容错等问题研究;(2)不同架构 CGRA 的划分映射问题研究.

关于 CGRA 存储和容错等问题,近年来典型研究列举如下:

文献[4]研究了 RCA 之间的原始或临时数据映射问题,考虑了局部存储器的大小、局部存储器和主存储器通信带宽等因素,提出了一种双缓冲局部存储机制,相比单缓冲局部存储机制而言,能耗消费减少了 47%,运行时间减少了 22%.文献[5]定义了 TMGC^2(Transformable Macro Granularity CGRA Computing)模型,提出了一种发掘循环迭代数据重用的存储体冲突消除算法,经过优化后的 CGRA 应用映射的资源利用率和数据吞吐量均得到了改进,提高的百分比分别为 41.3% 和 26.7%.文献[6]从软件层面对二维 mesh 架构的 CGRA 瞬时故障的容错机制进行了研究,并提出了一种基于数学分析的可靠性模型,通过实验验证该模型的可行性.

关于不同架构 CGRA 的划分映射问题,近年来典型研究列举如下:

文献[7]基于 RSPA(Resource Sharing and Pipelining Architecture)架构采用先列后行展开加旁节点 BN (Bypass Node)方式进行了节点映射,提出了 SPKM(Split-push Kernel Mapping)算法,获得了较好的映射效果,SPKM 算法的缺点一是加的 BN 较多;二是 RCA 块内跨层传输延迟较大.文献[8]针对一个 RC 的输入输出数过多影响了可重构处理器最大时钟频率的问题,设计了 2-D mesh CGRA + MIN(multistage interconnection networks)架构,RCA 块内一些数据传输通过块外一段或多段 Ω 互连网络来协助完成,并提出了一个放置和路由 PR (Placement and Routing)算法,在理想情况下,相比于简单 2-D mesh CGRA,该方法可获得了编译性能提升,但其硬件面积增大了很多.文献[9]初步给出了用一对 RC 单元共同参与浮点数尾数和阶码运算的映射方法.文献[10]提出了表达式粒度可重构阵列 EGRA(Expression-Grained Reconfigurable Array)架构,EGRA 由可重构算术逻辑单元簇 RAC(Reconfigurable ALU Cluster)、存储器 MEM(Memory)、乘法器 MULT(Multiplier)通过近邻和局部行列总线构成,一个 RAC 包含若干个 RC,其执行对象是在表达式粒度层面上的包含多个运算节点的簇,Ansaloni 等人基于 EGRA 给出了临时数据的寄存器存储和无寄存器存储数据相结合的 RAC 间通信方式,设计了计算模型松弛映射调度算法,该方法获得复杂计算内核运行时间的最大化,缺点是 RAC 的配置成本较大.文献[11]基于 2-D mesh CGRA + RF (Register File),RF 构成形式为 central、local、no shared,通过利用 RCA 中的 RF 存或取机制来进行中间存储 IS (Intermediate Storage),实现共享路由传输数据,达到改善节点调度优化的目的,并设计了一种单节点流水映射图较小化 GM(Graph Minor)映射算法,获得了较少的编译时间.但是上述工作均没有针对某一类架构来进行 RCA 划分映射问题研究,并且对 RCA 行运算节点并行执行最大化、RCA 块内异步计算延迟等问题考虑不足.

2.2 相关工作与本文工作的区别与联系

相关工作与本文工作的区别是:(1)文献[7]和文献

[8]等大多停留在 RCA 空域映射层面,而本文将其推广到时域映射层面,并且考虑了 RCA 块内并行度最大化、配置成本最小化等问题;(2)上述已有针对 CGRA 划分映射研究均基于某种特殊类型结构,并且大多停留在简单二维 mesh 极其类似架构,而本文研究对象为一类可行流水并行执行 CGRA 架构.

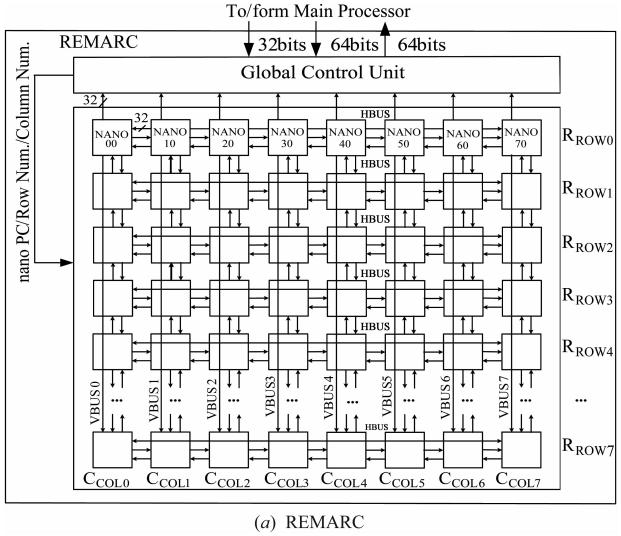
相关工作与本文工作的联系是:本文研究的对象是 CGRA 架构,并且关注的重点是多约束下的 RCA 的划分映射问题,这与上述相关工作研究基本一致.

本文研究基于如下 3 个前提条件:(1)本文研究映射目标架构统一为一类可满足行流水并行执行的 RCA 互连方式及类似结构(见 3.1 节),RCA 可按行动态配置.为了便于说明问题,只研究一块 RCA 的划分映射,没有考虑到点局部总线请求时延等,待映射的关键循环或任务已经转换为一个 DFG;(2)RCA 所涉及的中间运算数据或原始数据读写冲突现象已经消除,考虑了 RCA 运算节点异步计算时延,并且不允许 RCA 块内节点间的跨层数据传输;(3)由于增加过多 BN 映射会带来配置时间等的增加,故本文映射没有考虑.

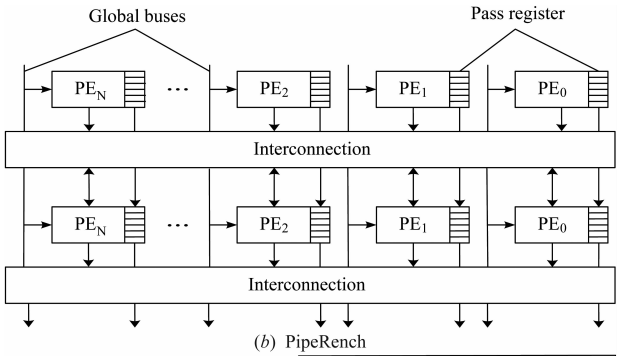
3　目标架构和问题定义

3.1　目标架构

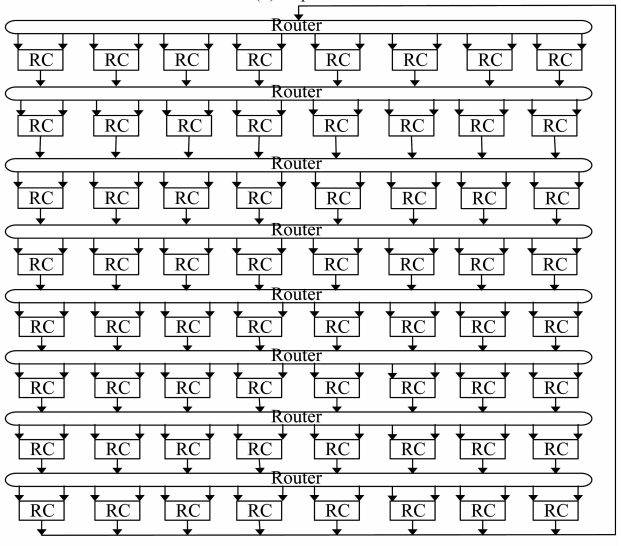
从计算密集型运算程序(如 C 程序等)提取出的关键循环任务 DFG,其大多具有多入度或多出度的特征,这样的 DFG 在 2-D mesh RCA 及类似架构上映射时,很难形成 RCA 行运算操作节点并行执行的最大化,所以本文研究的目标架构为一类可行并行执行的 CGRA 结构,其典型代表有 REMARC[12]、PipeRench[13]、REMUS[14]等(具体结构如图 1(*a*),(*b*),(*c*)所示),该类结构通过划分映射算法可以实现行操作节点并行执行,图 1(*d*)给出了 RC 或 PE 内部较为通用结构.



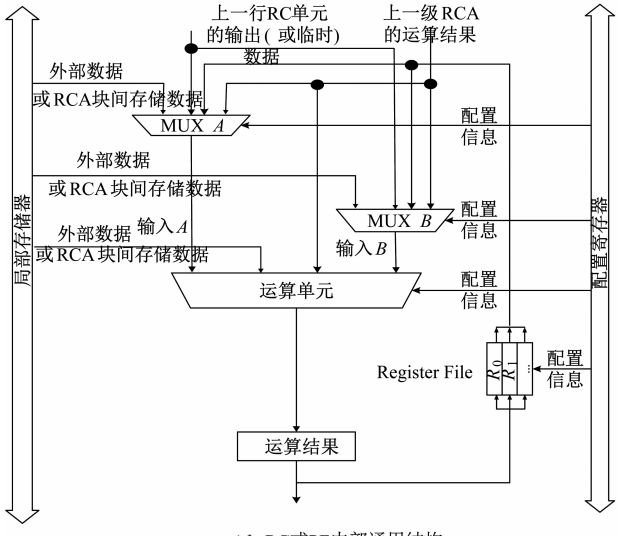
(*a*) REMARC



(*b*) PipeRench



(*c*) REMUS



(*d*) RC或PE内部通用结构

图1　RCA互连方式及RC内部结构说明

3.2　问题定义

为了便于说明问题,下面给出 CGRA 划分映射问题的相关定义:

定义 1　RC:RC 是构成 RCA 的核心,它具有如下功能:(1)通过配置具有完成简单或复杂算术逻辑运算功能;(2)具有数据路由或共享的功能;(3)在宏粒度层

面,若干个 RC 构成 RAC 模块,每个 RAC 可完成表达式等大粒度的计算;(4)一个或两个 RC 可完成定点或浮点数运算.

定义 2 单射:设 f 为图 G 到 RCA 的一个映射函数,$\forall v_1, v_2 \in G$,若 $v_1 \neq v_2$,则 $f(x_1) \neq f(x_2)$,称函数 f 是单射.一般而言,图 $G=(V,E,W,D)$经时域划分(定义见文[15])到 RCAM = (RC, I, O, E_L)(定义见文献[16])映射为单射.

定义 3 空域映射:图 G 的所有运算节点及连接边一次性全部被映射到 RCAM 的过程,可表示为 $G=(V,E,W,D) \xrightarrow{f}$ RCAM = (RC, I, O, E_L),则称该种映射方式为空域映射.空域映射只适合于任务 DFG 规模较小的情况,通常 DFG 规模均大于 RCA 阵列面积(可表示为 A_{RPU}).

定义 4 时域映射:若一块 RCA 可以重复使用,一个计算任务可以表示为图 $G(V,E,W,D)$,图 G 运算节点数大于 RCA 中的 RC 单元数,则图 G 要按多个约束规则被时域划分,然后依次被映射到 RCA 上,可表示为 $G=(V,E,W,D) \xrightarrow{f_i}$ RCAM = (RC, I, O, E_L),其中,f_i 表示第 i 次单射,则称该种映射方式为时域映射,本文重点研究时域映射.一般而言,一个 DFG 在 RCA 执行所需的总周期数 $T_{TOTAL}=T_{con}+T_{in}+T_{out}+T_{comp}+T_{link}=C_{CON}+\alpha[(N_1+N_{org1})+(N_2+N_{org2})]+S_{SD}+I_{ID}$,$\alpha$ 为修正系数,其大小反应了 RCA 块间通信成本,其由不同的 CGRA 架构及互连方式等所决定,REMUS 的 $\alpha=0.5$,C_{CON}、N_1、N_2、N_{org1}、N_{org2}、M、I_{ID}、S_{SD}等符号表示的含义与文献[16]一致,限于篇幅,不再累述,本文 S_{SD}按行统计运算节点的执行时延,并且包含了运算过程中产生的异步计算时延.本文的方法是 RCA 块点到点流水映射,I_{ID}近似为 0,故没有考虑.本文仅考虑定点运算,文中所用运算的时延为:加(减)法、乘法、条件运算所消耗的时延分别为 1cycle、2cycles、1cycle,占用资源量均为 1 个 RC.

定义 5 异步计算时延:设有一个二目运算节点 v_k,v_k所需的源操作数来自其前驱点 v_i,v_j的运算结果,delay(v_i),delay(v_j),delay(v_j)为点 v_i,v_j,$v_k \in V$ 的执行延迟,如 v_i,v_j被映射某个 RCA 的第 a 行,v_k被映射到同块的第 $a+1$ 行,按 RCA 行流水并行执行,若 delay(v_i) $\neq$ delay(v_j),v_k不能同步获得两个源操作数,则称 v_k产生了异步计算时延.

定义 6 RCA 行无数据依赖:设 $RCA_{n\times m}$,$G=(V,E,W,D)$,若存在一节点集 $v_1, v_2, \cdots, v_\delta \in V$,且被映射到 RCA 的第 k 行,$v_1, v_2, \cdots, v_\delta$ 与其已映射到当前块第 $k-1$ 行的前驱点集 prev(V) = $\{v_\alpha \cdots v_\tau\}$;或与其已映射到当前块第 $k+1$ 行的后继点集 succ(V) = $\{v_\beta \cdots v_\gamma\}$;或 RCA 块之间可以存在数据依赖或传输,但是 k 行上的 $v_1, v_2, \cdots, v_\delta$ 相互之间不存在数据依赖或传输,则称一块 RCA 存在行无数据依赖,否则称为行有数据依赖.

定义 7 RCA 行并行度:RCA 并行度可表示为一块 RCA 中存在行无数据依赖的节点数,设 RCA 的规模为 $n \times m$,则该 RCA 行并行度 $\leqslant n$,若 RCA 行并行度 $= n$,则称 RCA 并行度达到最大化.

定义 8 通信成本:RCA 数据通信成本大致包括 RCA 块内和块间数据传输两个部分,块内 RC 间的数据传输是通过数据流驱动的,一般通过 RC 的临时输出寄存器直接、寄存器文件存储转发或跨层等方式传输,其大小是由 I_{ID}的值来决定;RCA 块间数据通信大小主要是通过局部存储器、缓冲 cache、块外互连路由网络等方式传输,通过块间的存或取次数来体现,其大小是由 N_1 和 N_2的值来决定.

定义 9 配置时间:CGRA 的配置时间包括两个部分:一个是配置可重构硬件的功能和连接关系的控制字时间;一个是配置 RC 完成某个计算或路由功能的时间.

4 实验目标和多目标优化映射算法设计

4.1 实验目标

本文实验目标包括以下两点:(1)基于一类可行并行的 CGRA 架构,设计实现一个同时考虑并行度等指标的多目标优化映射 MOM (Multi-Objective Optimization Mapping)算法,考虑了行并行执行计算时延的一致性;(2)考虑了 *RCA* 块内通信成本的优化和 *RC* 碎片的有效利用.

4.2 MOM 算法设计

层划分及其优化算法在细粒度可重构体系结构中获得了较大并行度[15],本文运用此方法按 RCA 行节点可并行执行和硬件面积等多个约束,设计实现了该映射算法,其名称为层贪婪映射 LBGM (level based greedy mapping)算法,研究发现 LBGM 算法可获得块内运算节点较好的并行度,但通过深入的分析,发现 LBGM 仍然按层选取节点映射造成 N_1和 N_2较大,并且存在计算延迟等缺陷.在此基础上,本文设计了 MOM 算法,该算法考虑 RCA 并行度和计算时延的进一步优化,同时兼顾了 N_1、N_2、M 等的优化,相关策略列举如下:

策略 1 考虑 M 最小化,进行可重构硬件碎片的有效利用.

MOM 在映射过程中遇到不满足约束的节点时,再次搜索就绪队列,尽可能每次找到满足硬件约束要求的节点以填满每一块 RCA,从而避免了空载 RC 硬件碎片的产生.由上可知,策略 1 侧重于优化 M、C_{CON}.

策略 2 考虑 RCA 并行度和计算时延的较小化.

MOM 首先从入度为 0 的就绪节点队列中选取满足要求的节点按行映射,在选取节点的过程中,进行了节点延迟的均衡化,即尽可能把执行延迟相等的节点按层均衡放置,在增大行并行度的同时,减少了异步计算时延发生的机会,同时为了实现行流水,不允许块内一行 RC 间发生有数据依赖映射.由上可知,策略 2 侧重于优化 S_{SD}.

策略 3　考虑 S_{SD}和 N_1、N_2之间的均衡化.

MOM 在映射过程中,在满足相关约束下,对当前点的后继进行了动态映射.具体方法列举如下:

(1)MOM 每成功映射一个点后,就把该点后继的入度减 1,如果该点后继入度为 0,并且没有跨层等现象发生就直接映射到当前块.

(2)MOM 在已经映射成功的一行节点中,如果该行中的节点有一个共同的后继且该后继入度为 0,就直接将该点映射到当前行的下一行.

(3)考虑异步计算时延,尽可能把执行延迟相同操作放置在 RCA 同一行.由上可知,策略 3 侧重于减少 RCA 块间的 N_1和 N_2,同时考虑 S_{SD}和 N_1、N_2之间的均衡.

策略 4　考虑 RC 单元间的点到点行流水映射.

RCA 块内重构单元间相邻两行间不允许跨层和隔行数据传输存在,目的是使 I_{ID}近似为 0,由上可知,策略 4 侧重于优化 I_{ID}.

根据上述策略设计实现了 MOM 算法,同时还为 MOM 的就绪映射列表中的任务构造了累加权值概率函数 $P_w(v_i)$函数,并约定函数值越大越优先,其函数形式为:

$$P_w(v_i) = \text{delay}(v_i) + p(v_i) + \alpha \cdot p(v_j \mid v_i) + \beta \cdot p(v_k \mid v_i v_j) + \gamma \cdot p(v_m \mid v_i v_j v_k) - l_{level}(v_i) \quad (1)$$

其中,delay(v_i)表示节点 v_i的执行时延,其作用是尽可能把延迟大节点放置在一行,把延迟小节点放置在一行,其目的是优化异步计算时延,从而减少 S_{SD}值;$p(v_i)$表示就绪节点 v_i的概率值,取值范围为[0,1];在单目运算或双目、三目运算其他操作数可从局部存储器或缓存可获得的情况下,$p(v_j \mid v_i)$表示就绪任一节点 v_i映射到当前块某一行后,其不跨层的直接后继可映射到当前下一行的概率;在双目运算的情况下,$p(v_k \mid v_j v_i)$表示就绪点 v_i、v_j映射到当前块某一行后,其不跨层的直接后继可映射到当前下一行的概率,在三目运算的情况下,$p(v_m \mid v_i v_j v_k)$表示就绪点 v_i、v_j、v_k映射到当前块某一行后,其不跨层的直接后继可映射到当前下一行的概率;$p(v_j \mid v_i)$、$p(v_k \mid v_i v_j)$、$p(v_m \mid v_i v_j v_k)$的作用把依赖度大的点在满足硬件要求的前提下,尽可能放置在同一块,本文算法概率的初始值均设为 1,α、β、γ 为调整系数,其取值范围为[1, C_{vi}], $C_{vi} = \max\{\text{delay}(v_i), i \in [1, n]\}$,本文算法设定 $\alpha = \beta = \gamma = 1$,其目的减少 N_1和 N_2,若当前节点的不跨层的直接后继没有就绪,则 $p(v_j \mid v_i) = p(v_k \mid v_i v_j) = p(v_m \mid v_i v_j v_k) = 0$; $l_{level}(v_i)$表示节点 v_i的层次数,使层次小的节点尽可能得以优先映射,其目的是增大 RCA 块内点的并行度,优化 S_{SD}.

MOM 算法流程说明如下:

Algorithm: MOM

Input: DFG representing the loop kernel

Output: configuration information, M, N_1, N_2, S_{SD}, C_{CON}, T_{TOTAL}

Constraint_condition: $A_{RPU} = row \times col$, illegal dependencies are not happened, each row nodes can execute in parallel after each RCA is mapped successfully, data cannot be transmitted by crossing-level and dislocation in RCA, $I_{ID} \approx 0$

1　Begin

2　　*Reading_dfg_table*();行变量 $rv = 0$;块数 $M = 0$;节点数 $n = 0$;起始行放满标志 *flag*1 = 0;其他变量初始化;

3　for $i = 1$ to *nodenumber* do

4　　for $v_i = 1$ to *nodenumber* do

5　　end for

6　　while| *dfg-level* | *increasing* do

找到当前优先高的就绪点;过滤产生块内 I_{ID}的跨层传输数据等节点;

7　　if(node(v_i).flag = 0 && 行列满足要求 && 行放满或行没有放满不能再映射 && 无跨层和隔行)

放起始行 start_row[rv] = node[v_i].*id*; rv ++; n ++; node[v_i].flag = 1; node[v_i].*part* = M + 1; indegree(succ(v_i)) − 1;

8　　if(num[*row*1] = *col* ‖ not meet Constraint_condition) {*flag*1 = 1;}/* *row*1 为动态映射层变量 */

{for $k = 1$ to rv do

{*Mapping_succ*(start_row[k], *row*1, *flag*1);/* 后继点就绪,计算概率排序迭代映射,否则返回 */

$rv = 0$;若 num[*row*1]等于 *col*, *row*1 ++;}}

9　　if(行放满 ‖ 行没有满不能放了)/* 找下一个子图的起始行 */

{while(*row*1 < *row*){{if(列满) *row*1 ++; else break;}从 dfg 第 0 层开始下一轮搜索映射,break;}

10　　if(一块 RCA 放满了 ‖ 一块 RCA 没有放满,按约束不能放了)

{RCA 换块;RCA 块清零,变量初始化,更新就绪队列 by *priority_assigned*($P_w(v_i)$),转第 3 步}

```
11      if( n = nodenumber ) break;
12        End for
13      End for
14    End for
15    N1, N2 get by DFG_in_edges() and DFG_out_edges();
      M, SSD get by DFG_delays(); CCON get by DFG_Con();
      TTOTAL get by DFG_TOTAL_delays();
16    End MOM
```

5 实验及分析

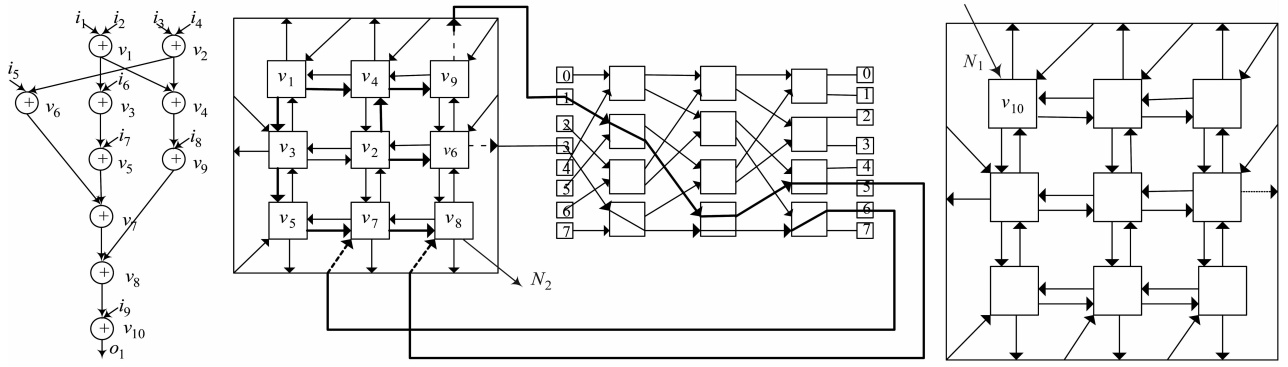
5.1 已有映射算法定性说明

目前,基于不同 CGRA 架构产生了较多国际先进映射算法,典型的有 PR[8]、GM[11]等算法,由于实验限制,现仅对这两种算法进行定性说明,两种算法的约束与 MOM 主要有以下不同:一是目标架构:PR 面向 2-D mesh CGRA + MIN,GM 面向 2-D mesh CGRA + RF,MOM 面向统一具有行流水特征的 CGRA;二是数据流水方式:GM、PR 为单节点流水,MOM 为行节点并行流水;三是评估目标:PR 目标是优化编译时间,GM 目标是优化资源利用率和编译时间,MOM 统一了量化评估指标体系,即 M、C_{CON}、N_1、N_2、S_{SD}、I_{ID}(为了便于统计,本文实验 S_{SD} 包含 I_{ID})、T_{TOTAL}.已有映射算法定性说明如下:

(1)PR 算法定性说明:PR 首先通过深度优先搜索获得关键路径上的节点放置,第二步通过 RCA 块外的 Ω 网络获得 RCA 块内所需数据传输,缺点是 I_{ID}增大了过多,虽然 N_1、N_2有所减少,但不足以平衡 I_{ID},举例:

例 1 现有一个循环子图 DFG(见图 2(a)),该图原始输入(或出)为 9(或 1),有加法运算 10 个,按 PR 算法获得的映射结果如图 2(b),(c)所示.由映射结果可知 S_{SD} = 5 + 1 = 6cycles,二目运算 v_7把来自 v_5的数据先存在 v_7所在的 RC 单元中的 RF,等 v_9传来数后,再取出来运算,时延为 2 $cycles$,v_9通过 Ω 网络传数到 v_7,每个路由的建立和传输为 1 拍,时延为 3cycles,故 I_{ID}合计为 5cycles;二目运算 v_8与其类似,综上,I_{ID} = 10cycles.小结,PR 算法的 I_{ID}较大,由文献[8]可知,PR 没有考虑异步计算时延、多循环子图行并行执行、RCA 复用次数 M 值的大小,由 4.2 节所述的策略 1-策略 4 可知,这些不足 MOM 均给予了优化或消除.

(2)GM 算法定性说明:GM 分时复用同一个 RC 单元,并通过 RF 实现数据共享传输,便于实现单节点流水,但是一个 RC 在不同节拍被分时复用,RC 的执行变为串行,异步计算时延较大.举例:



(a) 一个循环子图DFG (b) PR映射结果(M=1) (c) PR映射结果(M=2)

图2 定性说明PR映射的例子

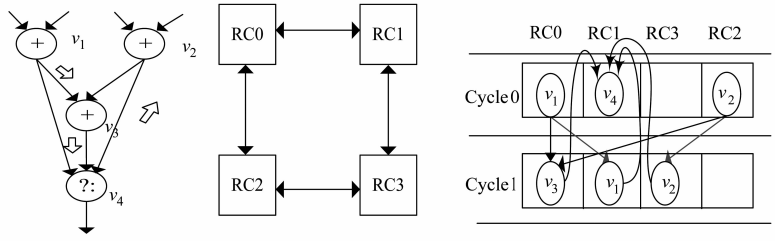
例 2 一个 DFG 循环子图和架构见图 3(a),(b),该图原始输入(或出)为 4(或 1),有加法运算 3 个,条件运算 1 个(注:本例引自文献[11],并给每个节点添加了运算类型.),按 GM 获得的映射结果如图 3(c)所示,由映射结果可知,RC0 被分时复用,在理想情况下,假设 v_1和 v_2可以并发执行,S_{SD} = 实际计算时间(即 T_{v1} + T_{v3} + T_{v4}) + 异步计算时延(即 $T_{v1存取}$ + 2 × $T_{v2存取}$) = 3 + (2 + 4) = 9cycles,S_{SD}值增大了;增加的配置时间消耗 C_{CON} = Cycle1 时 RC1 的路由配置 + Cycle1 时 RC3 的路由配置 = 1 + 1 = 2cycles;RCA 块间的通信成本也没有考虑.

小结,GM 对 S_{SD}、C_{CON}、N_1、N_2等指标考虑不足,由文[11]可知,GM 没有考虑 RCA 分时复用的 M 值的大小,由 4.2 节所述的策略 1-策略 4 可知,MOM 考虑了或尽可能避免了这些不足.

5.2 异步计算时延说明

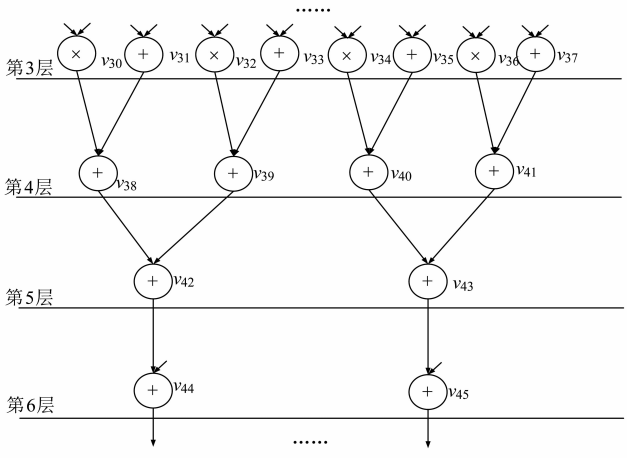
例 3 现有两个正在划分中的关键循环 DFG 子图 1(见图 4(a))和子图 2(见图 4(b)),两个子图均有原始输入(或出)为 18(或 2),有乘法运算 4 个,加法运算 12 个,但两子图具体算子的位置有差别,具体见图 4(a),(b)所示,图 4(a)按 LBGM 和 MOM 算法获得的映射结果如图 4(c),(d)所示,图 4(b)按 LBGM 和 MOM 算法获得的映射结果如图 4(e),(f)所示.针对图 4(a)的情况,

从图 4(*c*)LBGM 映射结果可知,v_{34}和 v_{35}、v_{36}和 v_{37}之间产生了异步计算时延,因为行并行执行,只统计一次,MOM 给予了消除,见图 4(*d*),但增大了 N_1、N_2,以 REMUS 为例,通过其编译器的测试,得出通信成本为 RCA 块间原始和非原始 I/O 次数总和的一半,由表 1(*a*)可知 MOM 得到了较好的均衡;针对见图 4(*b*)的情况,从图 4(*e*)LBGM 映射结果可知,v_{70}和 v_{71}之间产生了异步计算时延,MOM 进行了均衡,具体见表 1(*b*).

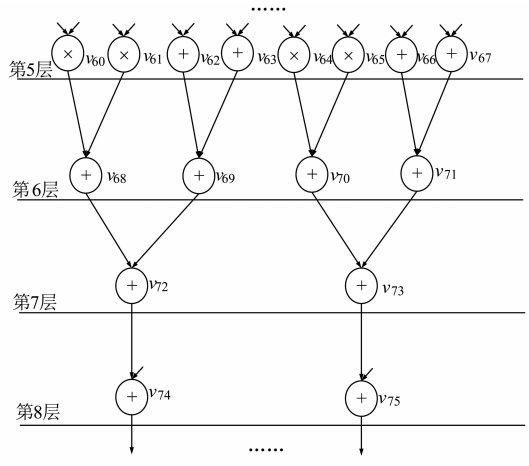


(*a*)一个DFG　(*b*) 2×2 CGRA　(*c*) GM映射结果

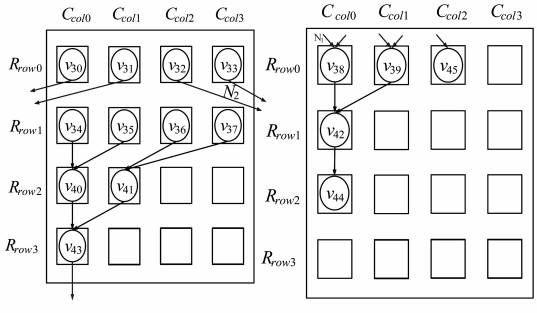
图3　定性说明GM映射的例子[11]



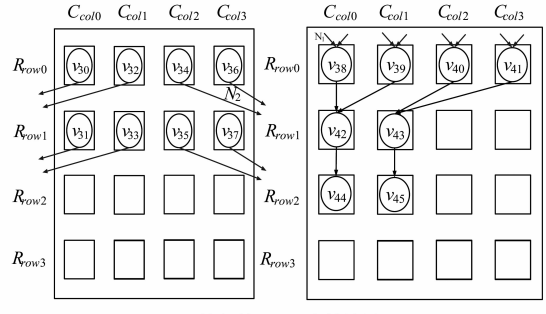
(*a*) 划分中的循环DFG子图1



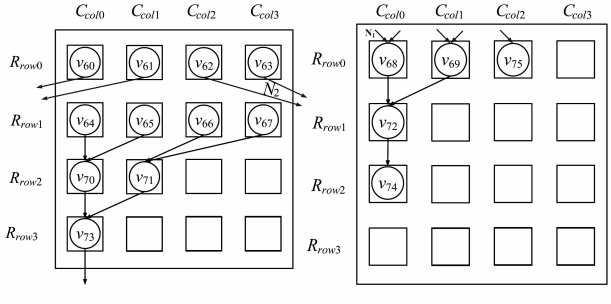
(*b*) 划分中的循环DFG子图2



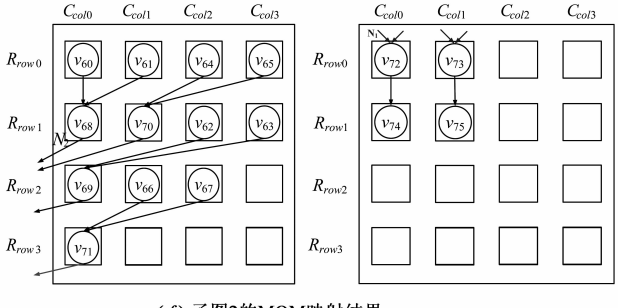
(*c*) 子图1的LBGM映射结果



(*d*) 子图1的MOM映射结果



(*e*) 子图2的LBGM映射结果



(*f*) 子图2的MOM映射结果

图4　说明两种LBGM和MOM映射的例子

表 1 两种情况映射结果比较

(*a*)子图 1LBGM 和 MOM 映射比较

参数指标 \ 类型	LBGM	MOM
M	2	2
N_1	5	8
N_2	5	8
S_{SD}	10	6
C_{CON}	50	50
T_{TOTAL}	75	74

(*b*)子图 2LBGM 和 MOM 映射比较

参数指标 \ 类型	LBGM	MOM
M	2	2
N_1	5	4
N_2	5	4
S_{SD}	10	7
C_{CON}	50	50
T_{TOTAL}	75	71

表 2 基准程序集

基准	总数	加法	减法	乘法
LOOP4	32	24	—	8
LOOP6	48	36	—	12
FFT8	36	12	12	12
EWF3	102	84	—	18
EWF6	204	168	—	36
FDCT3	126	39	39	48
FDCT6	252	78	78	96

5.3 实验基准

为了比较不同映射算法,设计了两部分基准程序,一是 FFT8、EWF3、EWF6、FDCT3、FDCT6 等五个基准;二是具有图 4(*a*)特征展开的 4 次循环内核 LOOP4 和具有图 4(*b*)特征展开的 6 次循环内核 LOOP6,表 2 列出了这些基准,各种运算的时延和占用资源量前面已经说明,本文 A_{RPU}随机选取 16RC(RCA_{4*4})、36 RC(RCA_{6*6})两个值,对 LBGM、SPKM、MOM 三种时域映射算法进行了比较.

5.4 LBGM 与 MOM 比较

以行可并行执行的 REMUS 为例,采用表 2 中的 7 个基准程序,采用 LBGM 和 MOM 算法进行了实验,表 3 给出了 $A_{RPU}=16$ 和 36 两种映射算法所得到的结果.对应于 M、N_1、N_2、S_{SD} C_{CON}、T_{TOTAL}等指标,每个映射算法和改进百分比($\Delta\%$)列出了四大列内容.对应于 M 列,以 EWF3 为例,当 $A_{RPU}=16$ 时,用 LBGM 算法所获得的 $M=8$,而采用 MOM 算法所获得的 $M=7$,因此,$\Delta\%=-12.5\%$,百分比为负数表示改进,为正数表示没有改进,后面与此相同.与 LBGM 相比,MOM 算法的 M、T_{TOTAL}、C_{CON}获得较好的改进,N_1、N_2、S_{SD}的均值也是最小的.

表 3(*a*) 5.4 节实验 $A_{RPU}=16,36$ 时 LBGM 与 MOM 映射比较

划分基准	M						N_1						N_2					
	LBGM		MOM		Δ%		LBGM		MOM		Δ%		LBGM		MOM		Δ%	
LOOP4	3	2	3	2	—	—	36	12	33	16	−8.3	+33.3	20	12	17	16	−15.0	+33.3
LOOP6	4	2	4	2	—	—	28	20	12	14	−57.1	−30.0	26	20	12	14	−53.8	−30.0
FFT8	4	4	4	4	—	—	18	17	18	17	—	—	18	17	18	17	—	—
EWF3	8	5	7	5	−12.5	—	79	51	64	47	−19.0	−7.8	64	45	54	41	−15.6	−8.9
EWF6	14	8	13	6	−7.1	−25.0	241	161	126	120	−47.7	−25.5	167	124	102	94	−38.9	−24.2
FDCT3	9	5	8	5	−11.1	—	102	60	55	44	−46.1	−26.7	102	60	59	51	−42.2	−15.0
FDCT6	16	8	16	8	—	—	204	204	108	101	−47.1	−50.5	204	204	102	101	−50.0	−50.5
平均 Δ%	—	—	—	—	−4.4	−3.6	—	—	—	—	−32.2	−15.3	—	—	—	—	−30.8	−13.6

表 3(*b*) 5.4 节实验 $A_{RPU}=16,36$ 时 LBGM 与 MOM 映射比较

划分基准	S_{SD}						C_{CON}						T_{TOTAL}					
	LBGM		MOM		Δ%		LBGM		MOM		Δ%		LBGM		MOM		Δ%	
LOOP4	14	15	13	9	−7.1	−40.0	83	66	83	66	—	—	145	113	141	111	−2.8	−1.8
LOOP6	21	14	18	14	−14.3	—	116	82	116	82	—	—	194	146	176	140	−9.3	−4.1
FFT8	15	11	13	11	−13.3	—	104	104	104	104	—	—	151	146	149	146	−1.3	—
EWF3	39	26	37	29	−5.1	+11.5	238	187	221	187	−7.1	—	386	299	355	298	−8.0	−0.3
EWF6	73	54	72	43	−1.4	−20.4	442	340	425	306	−3.8	−10.0	794	612	686	531	−13.6	−13.2
FDCT3	47	37	47	32	—	−13.5	279	211	262	211	−6.1	—	488	368	426	351	−12.7	−4.6
FDCT6	93	65	90	65	−3.2	—	524	388	524	388	—	—	941	777	839	674	−10.8	−13.3
平均 Δ%	—	—	—	—	−6.3	−8.9	—	—	—	—	−2.4	−1.4	—	—	—	—	−8.4	−5.3

5.5 SPKM 与 MOM 比较

国际上,SPKM 是一个先进的映射算法,表 4 给出了 $A_{RPU}=16$ 和 36 时,MOM 和 SPKM 两种算法的实验结果比较.由表 4 可知,基于 RSPA 硬件约束,采用添加 BN 的方法来解决 RCA 块内跨层或数据过渡,加少量的 BN 可以减少 N_1和 N_2,但是一个 BN 需要 1 个 cycle 的配置时间和传输过渡数据时延,SPKM 一遇到跨层点就加 BN,导致 C_{CON}等的增大,而且 SPKM 没

有考虑异步计算时延,对 RCA 并行度和计算时延等考虑不足,并且允许 RC 跨层,从而产生较多 I_{ID},导致增大 S_{SD}(本文 S_{SD}的值包括 I_{ID});另外,SPKM 只有遇到倾斜点时才处理 DFG 节点间的依赖关系,产生较多的不满足硬件约束的无效映射,SPKM 对 RC 硬件碎片也没有进行有效利用及贪婪映射,导致 M、C_{CON}增大较多.与 SPKM 相比,MOM 算法全面优化了 M、T_{TOTAL}、C_{CON}等指标,获得了较小的 S_{SD}均值,但是 N_1、N_2不及 SPKM 算法.

表 4(*a*)　5.5 节实验 $A_{RPU}=16,36$ 时 MOM 与 SPKM 的映射比较

划分基准	M						N_1						N_2					
	SPKM		MOM		Δ%		SPKM		MOM		Δ%		SPKM		MOM		Δ%	
LOOP4	4	4	3	2	−25.0	−50.0	12	16	33	16	+175	—	12	16	17	16	+41.7	—
LOOP6	7	4	4	2	−42.9	−50.0	16	8	12	14	−25.0	+75.0	20	18	12	14	−40.0	−22.2
FFT8	5	4	4	4	−20.0	—	22	23	18	17	−18.2	−26.1	21	21	18	17	−14.3	−19.0
EWF3	9	6	7	5	−22.2	−16.7	63	47	64	47	+1.6	—	51	31	54	41	+5.9	+32.3
EWF6	17	12	13	6	−23.5	−50.0	156	75	126	120	−19.2	+60.0	102	60	102	94	—	+56.7
FDCT3	15	9	8	5	−46.7	−44.4	60	64	55	44	−8.3	−31.3	60	60	59	51	−1.7	−15.0
FDCT6	31	20	16	8	−48.4	−60.0	170	174	108	101	−36.5	−42.0	121	170	102	101	−15.7	−40.6
平均 Δ%	—	—	—	—	−32.7	−38.7	—	—	—	—	+9.9	+5.1	—	—	—	—	−3.4	+1.1

表 4(*b*)　5.5 节实验 $A_{RPU}=16,36$ 时 MOM 与 SPKM 的映射比较

划分基准	S_{SD}						C_{CON}						T_{TOTAL}					
	SPKM		MOM		Δ%		SPKM		MOM		Δ%		SPKM		MOM		Δ%	
LOOP4	11	9	13	9	+18.2	—	100	100	83	66	−17.0	−34.0	143	145	141	111	−1.4	−23.4
LOOP6	17	10	18	14	+5.9	+40.0	167	116	116	82	−30.5	−29.3	232	169	176	140	−24.1	−17.2
FFT8	19	22	13	11	−31.6	−50.0	126	109	104	104	−17.5	−4.6	222	167	149	146	−32.9	−12.6
EWF3	37	32	37	29	—	−9.4	257	240	221	187	−14.0	−22.1	384	349	355	298	−7.6	−14.6
EWF6	62	73	72	43	+16.1	−41.1	499	486	425	306	−14.8	−37.0	810	702	686	531	−15.3	−24.4
FDCT3	72	44	47	32	−34.7	−27.3	381	279	262	211	−31.2	−24.4	573	453	426	351	−25.7	−22.5
FDCT6	143	97	90	65	−37.1	−33.0	921	608	524	388	−43.1	−36.2	1330	997	839	674	−36.9	−32.4
平均 Δ%	—	—	—	—	−9.0	−17.3	—	—	—	—	−24.0	−26.8	—	—	—	—	−20.6	−21.0

5.6 总结

从定性的角度分析了国际先进 PR、GM 等算法的不足,MOM 在调度映射给予了考虑;从表 3 和表 4 的实验比较结果可以看出,MOM 算法相比于 LBGM 和 SPKM 算法,M、T_{TOTAL}、C_{CON}均获得了较好的改进;MOM 算法的 S_{SD}均优于 LBGM 和 SPKM,N_1、N_2均值优于 LBGM,但是 N_1、N_2均值不如 SPKM.MOM 算法适用场合:要求 RCA 互连满足行流水并行执行和配置,在此前提下,可获得块内两个相邻行运算节点间的点到点流水化映射,从而获得了 M、T_{TOTAL}、C_{CON}较小化和计算延迟的优化.

6 结语

本文提出了一种多目标优化的时域映射 MOM 算法,从定性的角度,MOM 能消除国际先进 PR、GM 等算法的不足;从定量的角度,采用了一组划分映射基准程序集进行了 MOM、LBGM、SPKM 等算法的实验比较,结果表明,MOM 算法在 M、T_{TOTAL}、C_{CON}等方面均较具优势,获得了 RCA 并行度最大化和计算时延的最小化,在减少 T_{TOTAL}方面,本文算法具有合理性和可行性.

参考文献

[1] Cardoso J M P, Diniz C D, Weinhardt M. Compiling for reconfigurable computing: a survey[J]. ACM Computing Surveys, 2010, 42(4): 1301 – 1365.

[2] Yoon J W, Lee J, Park S, et al. Architecture customization of on-chip reconfigurable accelerators[J]. ACM Transactions on

Design Automation of Electronic Systems,2013,18(4):52:1 – 52:22.

[3] Zhao X, Erdogan A T, Arslan T. High-efficiency customized coarse-grained dynamically reconfigurable architecture for JPEG2000[J]. IEEE Transactions on Very Large Scale Integration Systems,2013,21(12):2343 – 2348.

[4] Kim Y, Lee J, Shrivastava A, et al. High throughput data mapping for coarse-grained reconfigurable architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2011,30(11):1599 – 1609.

[5] 杨子煜,严明,王大伟,等.面向 CGRA 循环流水映射的数据并行优化[J].计算机学报,2013,36(6):1280 – 1289.
Yang Ziyu, Yan Ming, Wang Dawei, et al. Data parallelism optimization for the CGRA loop pipelining mapping[J]. Chinese Journal of Computers,2013,36(6):1280 – 1289.(in Chinese)

[6] Han K, Lee G, Choi K. Software-level approaches for tolerating transient faults in a coarse-grained reconfigurable architecture [J]. IEEE Transactions on Dependable and Secure Computing, 2014,11(4):392 – 398.

[7] Yoon J W, Shrivastava A, Park S, et al. A graph drawing based spatial mapping algorithm for coarse-grained reconfigurable architectures[J]. IEEE Transactions on Very Large Scale Integration Systems,2009,17(11):1565 – 1578.

[8] Ferreira R S, Cardoso J M P, Damiany A, et al. Fast placement and routing by extending coarse-grained reconfigurable arrays with Omega Networks[J]. Journal of Systems Architecture, 2011,57(8):761 – 777.

[9] Lee G, Choi K, Dutt N D. Mapping multi-domain applications onto coarse-grained reconfigurable architectures [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2011,30(5):637 – 650.

[10] Ansaloni G, Tanimura K, Pozzi L, et al. Integrated kernel partitioning and scheduling for coarse-grained reconfigurable arrays[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2012,31(12):1803 – 1816.

[11] Chen L, Mitra T. Graph minor approach for application mapping on CGRAs[J]. ACM Transactions on Reconfigurable Technology and Systems,2014,7(3):21:1 – 21:25.

[12] Miyamori T, Olukotun K, Budiu M, et al. REMARC: reconfigurable multimedia array coprocessor[J]. IEICE Transactions on Information and Systems,1999,E82 – D(2):389 – 397.

[13] Goldstein S C, Schmit H, Budiu M, et al. PipeRench: A reconfigurable architecture and compiler[J]. Computer, 2000, 33 (4):70 – 77.

[14] 魏少军,刘雷波,尹首一.可重构计算处理器技术[J].中国科学:信息科学,2012,42(12):1559 – 1576.
Wei Shaojun, Liu Leibo, Yin Shouyi. Key techniques of reconfigurable computing processor[J]. Science China: Information Sciences,2012,42(12):1559 – 1576.(in Chinese)

[15] 陈乃金,江建慧,陈昕,等.一种考虑执行延迟最小化和资源约束的改进层划分算法[J].电子学报,2012,40(5):1055 – 1066.
Chen Naijin, Jiang Jianhui, Chen Xin, et al. An improved level partitioning algorithm considering minimum execution delay and resource restraints[J]. Acta Electronica Sinica, 2012, 40 (5):1055 – 1066.(in Chinese)

[16] 陈乃金,冯志勇,江建慧.用于二维 RCA 跨层数据传输的旁节点无冗余添加算法[J].通信学报,2015,36(4):2015132:1 – 17.
Chen Naijin, Feng Zhiyong, Jiang Jianhui. Bypass node non-redundant adding algorithm for crossing-level data transmission in two-dimension reconfigurable cell array[J]. Journal on Communications,2015,36(4):2015132:1 – 17.(in Chinese)

作者简介

陈乃金 男,1972 年生于安徽合肥,天津大学博士后,安徽工程大学副教授,硕士生导师,主要研究方向为可重构计算、时域划分与映射、VLSI/SoC 测试与容错等.
E-mail:86naijinchen@tongji.edu.cn

江建慧 男,1964 年生于浙江淳安,博士,同济大学教授,博士生导师,主要研究方向为 VLSI/SoC 测试与容错、可信系统与网络、软件可靠性工程等.
E-mail:jhjiang@tongji.edu.cn