

核心循环到粗粒度可重构体系结构的流水化映射

王大伟 窦 勇 李思昆

(国防科学技术大学计算机学院软件所 长沙 410073)

摘 要 粗粒度可重构体系结构为数据密集型应用提供了灵活性和高效的解决方法,而应用中的核心循环消耗了程序的大量执行时间,满足核心循环在 CGRAs 上实现的性能/开销的严格约束仍旧是个重大难题.针对已有工作在研究映射核心循环到 CGRAs 上的不足,文中提出一种新颖的核心循环自动流水映射到粗粒度可重构体系结构上的方法.文中形式化了核心循环到 CGRAs 的流水映射问题,阐述了 CGRAs 的资源共享和流水方法,定义了其循环自流水 CGRAs 体系结构模板,并给出核心循环流水映射方法.实验结果表明,与已有的先进的方法相比,文中方法的资源占用率降低 16.3%、吞吐量提高 169.1%.

关键词 可重构计算;粗粒度可重构体系结构;数据密集型应用;循环自流水

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2009.01089

Loop Kernel Pipelining Mapping onto Coarse-Grained Reconfigurable Architectures

WANG Da-Wei DOU Yong LI Si-Kun

(College of Computer Science, National University of Defence Technology, Changsha 410073)

Abstract Coarse-grained reconfigurable architectures provide flexible and efficient solution for data-intensive applications. Loop kernels of these applications always consume much execution time of the whole program. However, mapping loop kernels onto CGRAs is still hard for meeting performance/cost constraints. This paper proposes a novel approach for mapping loop kernels onto CGRAs with loop self-pipelining to solve the existing problems. The problem formulation is shown first. Then the resource sharing and pipelining of lspCGRAs are presented, together with its template standard. A field specific application driven mapping flow is described. Besides, a loop kernel pipelining mapping algorithm is proposed. The conclusions show that the proposed approach gains less resource utilization by 16.3% times and more throughputs by 169.1% times than previous advanced SPKM.

Keywords reconfigurable computing; CGRA; data-intensive application; loop self-pipelining

1 引 言

可重构计算技术同时具备定制计算的高效性和通用计算的灵活性,对于加速大多数应用发挥着重要作用^[1-2].相比细粒度可重构体系结构,粗粒度可重构体系结构(Coarse-Grained Reconfigurable

Architectures, CGRAs)可获得更多的加速和消耗更少的功耗,却具有较高的设计难度.尽管 CGRAs 可同时获得高性能和灵活性,但在开发特定领域的应用时,满足其在性能/开销等方面的严格约束仍旧是个重大难题.

CGRAs 提供了有效和方便的解决方法来加速数据密集型应用(Data Intensive Applications, DIA),

收稿日期:2008-10-28;最终修改稿收到日期:2008-12-25.本课题得到国家自然科学基金(90707003,90707001)资助.王大伟,男,1980年生,博士研究生,主要研究方向为 SoC 协同设计. E-mail: daweiwang@nudt.edu.cn. 窦 勇,男,1966年生,博士,教授,主要研究领域为计算机体系结构、可重构计算、编译优化技术等.李思昆,男,1941年生,教授,博士生导师,主要研究领域为 SoC 设计方法学、设计自动化等.

如图像压缩、模式识别和数字信号处理. 目前国际上已提出多种 CGRAs, 并开发各类应用到其体系结构上的映射. 有些研究使用专门的硬件加速器来执行应用的关键循环嵌套, 尽管该方法可通过开发硬件资源共享来设计较低开销的硬件, 与 CGRAs 相比, 却失去了较多灵活性^[3-4]. 大多数已有映射技术在高层编译上提出了复杂的流水化技术来支持循环加速^[5-6], 但获得的加速性能较低, 编译过程复杂且优化效果一般.

处理数据密集型应用通常要求较高的吞吐量和并行性, 而手工将应用映射到 CGRAs 费时且易出错; 同时, 应用中的核心循环消耗了大量系统资源, 有必要对这些核心循环进行专门的加速与优化. 针对已有工作在研究映射核心循环到 CGRAs 上的不足, 本文提出一种新颖的核心循环自动流水映射到粗粒度可重构体系结构上的方法 (Loop Kernel Pipelining Mapping, LKPM). 我们开发了循环自流水粗粒度可重构体系结构 (loop self-pipelining Coarse Grain Reconfigurable Architectures, lspCGRAs), 它支持循环自流水, 使用固定指令流多数据流执行模式, 可获得较高的计算吞吐量. 同时, 我们给出 lspCGRAs 体系结构模板的资源共享与流水特性, 并提出核心循环流水映射方法. 本文的主要贡献如下:

(1) 形式化了核心循环到粗粒度可重构体系结构的流水映射问题. 与已有的先进映射方法相比, 我们的方法支持循环流水化和计算/存储的同时映射, 通过循环分解流水、冲突消解和数据重用等策略, 消除了数据密集型应用的处理瓶颈, 可有效提高吞吐量.

(2) 本文的自动映射方法与手工映射优化的性能相差很小; 同时, 与已有的先进的 SPKM 方法相比, 我们的方法资源占用率降低 16.3%、吞吐量提高 169.1%.

本文第 2 节介绍相关研究工作; 第 3 节形式化定义核心循环到 lspCGRAs 上映射的问题形式化; 第 4 节描述 lspCGRAs 体系结构模板; 第 5 节给出领域应用驱动的映射流程和核心循环流水映射方法; 第 6 节给出实验结果以证明本文方法的有效性; 第 7 节给出本文的结论.

2 相关工作

为开发并行性, 已有的研究工作将很多优化技

术引入到体系结构设计中, 并通过开发异构专用多处理器 SoC, 来进行特定应用功能的加速^[7]. 在开发数据密集型应用的并行性时, CGRAs 提供了方便而有效的解决方法^[8-9]. 另一方面, 因为应用中的大多数循环消耗了程序的大量执行时间, 循环级并行成为程序并行中最为广泛使用的技术^[3,10].

CGRAs 的性能主要取决于其体系结构模板和数据密集型应用的映射策略. XPP 使用向量 C 编译器 XPP-VC, 支持自动循环展开, 使用并行数据流处理时间关键的内部循环, 同时并行处理外部循环^[8]. Mophosys 编译器能够分析 SA-C 程序、执行优化和映射图像应用^[9]. 但文献[8-9]仅能支持简单的循环. Lee^[11]提出一个新颖的将循环自动映射到动态可重构阵列 (Dynamic Reconfigurable ALU Array, DRAA) 的方法, 该阵列使用一个通用的体系结构模板以配置较为广泛的 CGRAs. 但由于其多种体系结构特征和它们之间的相互依赖性, 该 CGRAs 的编译比传统的微处理器编译更加复杂. Rong^[12]使用单维软流水处理多维循环, 可将 n 维软循环问题减少到 1 维软流水, 并能较好地支持循环选择. 但该方法仅将循环映射到 IA-64 体系结构. 此外, 文献[11-12]都没考虑数据密集型领域应用在其体系结构上的优化技术.

在体系结构模板上, 与我们的研究比较接近的 Kim 方法^[13], 其考虑了 CGRAs 的资源流水以处理某些专用领域的优化, 却没有进一步考虑循环流水和数据密集型领域.

在映射策略上, 比较先进的 SPKM 和 Spatial 映射方法^[5,14]可支持自动有效映射大多数应用到 CGRAs, 并取得比手工映射相当的结果, 获得较高的资源利用率; 但它们在数据密集型应用方面, 没有充分利用其数据重用性和消除存储瓶颈, 以提高数据处理的吞吐量.

同时, 以前的应用映射方法仅考虑简单的 CGRAs 模型, 没有考虑复杂的 CGRAs 模型 (如 lspCGRAs) 来加速数据密集型应用. 而我们的方法使用新颖的循环自流水粗粒度可重构体系结构, 通过循环自流水, 可直接将循环表达式语句映射到 PEs, 并可有效提高资源利用率和吞吐量.

3 核心循环到 lspCGRAs 映射的问题形式化

数据密集型应用通常具有较高的数据并行、规

处理单元. 处理单元(Processing Element, PE)是对数据进行处理的功能部件,为达到存储和计算的分离,PE被分成了两种:存储器处理单元 mPE 和计算处理单元 cPE, mPE 专门负责存储器访问, cPE 仅具有计算功能。

局部存储器. 局部存储器用于缓存数据. 为实现存储和计算的完全分离,局部存储器(Local Memory, LM)仅被相邻两个 mPE 共享. 处理单元 mPE 可直接从局部存储器中取得数据,并把结果存入局部存储器. 此外,由寄存器组成的智能缓冲区(SmartBuffer, SB),用来缓存重复使用的数据。

互连网络. mPE 直接与局部存储器相连,同时

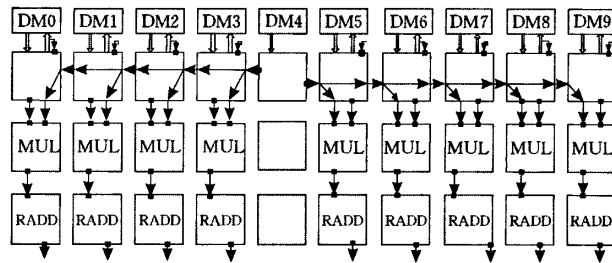


图 2 FDCT 到 3×10 LEAP 体系结构的映射示例

根据上述的定义,核心循环到 lspCGRAs 上映射问题的形式化如下。

定义 4 (M_LK-lspCGRAs 问题). 给定一个表示数据密集型应用的 PIA-CDTG 中的关键循环块 $V=(T, E)$ 和表示循环自流水粗粒度可重构体系结构的 RAG $R=(PE, C)$, 寻找一个映射绑定 $M: V \rightarrow R$, 在满足资源(如计算资源、存储资源、通信资源)约束的前提下,追求目标 $\min UPE$ 和 $\max THO$. 其中, UPE 为 PE 占用率, THO 为循环流水化的吞吐率,在资源充足的情况下,优先满足目标 $\max THO$, 在资源不足的情况下,优先满足目标 $\min UPE$.

该问题主要由目标函数及其约束这两个要素构成. 其目标函数定义为

$$Objective Function = \{UPE, THO\}.$$

对于 $m \times n$ 的 PE 阵列,其 cPE 有 $m \times (n-1)$ 个, mPE 有 n 个. 如果映射 M 占用了 s 个 cPE 和 t 个 mPE. 其中, $s \leq m$, $t \leq n$, 则

$$UPE = (s+t)/(m \times n).$$

在这里没有考虑 LM 和 SB 的大小,只考虑 PE 阵列的资源占用率. UPE 的值间接表示了映射 M 的面积开销,即 UPE 的值越小,面积开销越小,反之亦然。

通过 2D torus 网与 cPE 相连. 循环运行时 mPE 与 cPE 的连接关系是固定的,可通过互连网络的配置信息确定这种连接关系. 处理单元阵列(Process Element Array, PEA)同一行的 PE 挂在了同一个总线分支上,总线采用离散的结构有利于降低总线延迟. 配置总线用于传递配置信息和配置加载使能,启动总线用于控制阵列的运行。

为直观显示 lspCGRAs 的结构,图 2 给出了 FDCT 到 3×10 LEAP 体系结构的映射示例. 其中,第 1 行表示局部数据存储器,第 2 行是 mPE,第 3、4 行是 cPE。

$$THO = OP / \sigma,$$

其中, OP 表示执行循环的操作数量, σ 表示循环消耗的时钟周期数;吞吐量的单位用“每秒多少次操作”来表示. THO 的值间接表示了映射 M 处理操作的能力. 由于对循环程序使用了流水线并行,理想目标是每时钟节拍完成一个循环迭代,即每拍最大吞吐率 $\#OP$ 等于 $\#CPE$, 而实际吞吐率受多种因素影响,如流水线启动开销、存储访问延迟等。

影响这两个目标函数的 CGRAs 约束集合为

$$CGRAs-Constraint-Parameters =$$

$$\{mPE_c, cPE_c, LM_c, LBus_c\};$$

这些 CGRAs 约束的定义如下:

mPE 约束 (mPE_c). 对于 $m \times n$ 的 PE 阵列,所有 cPE 使用的 mPE 的个数不超过 mPE 的总量 n . 设 m_{ij} 为 PE 阵列中对 mPE_{ij} 进行使用配置,则 mPE_c :

$$\sum_j m_{ij} \leq \# \text{ of } mPE = n;$$

cPE 约束 (cPE_c). 对于 $m \times n$ 的 PE 阵列,使用 cPE 的总数不超过其总量 $m \times (n-1)$, 其中,每行使用的 cPE 的总数不超过 n . 设 c_{ij} 为 PE 阵列中对 cPE_{ij} 进行使用配置,则 cPE_c :

$$\sum_j c_{ij} \leq \# \text{ of } cPE \text{ in } i\text{th row} = n;$$

局部存储器约束 (LM_c)。局部存储器 (Local Memory, LM) 仅被相邻两个 mPE 共享, 设 m_{ij} 为 PE 阵列中对 mPE_{ij} 进行使用配置, l_{ij} 为 m_{ij} 中的原子局部存储器访问操作, 则 LM_c :

$$l_{ij} + l_{i(j+1)} \leq \# \text{ of LM resources in } i\text{th row, } j\text{th column;}$$

同行互连总线约束 ($LBus_c$)。处理单元阵列中同一行的 PE 挂在了同一个总线分支上, 设 c_{ij} 为 PE 阵列中对 cPE_{ij} 进行使用配置, b_{ij} 为 c_{ij} 中的原子总线访问操作, 则 $LBus_c$:

$$\sum_j b_{ij} \leq \# \text{ of memory bus resources in } i\text{th row.}$$

4 lspCGRAs 体系结构模板

4.1 资源共享与流水

在 lspCGRAs 中, 存在许多共享资源, 其中计算资源有 cPE, 存储资源有 mPE、LM 和数据存储器, 通信资源有互连网络, 同一行 PE 共享一个总线分支, 相邻两个 mPE 共享一个 LM, 如图 3 所示。

为实现对资源的利用, 对 lspCGRAs 设计使用资源流水技术。循环自动流水化充分利用了数据驱

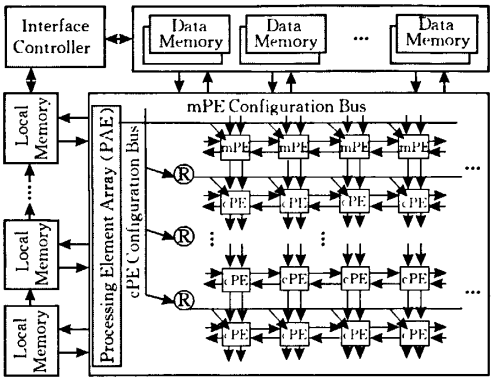


图 3 lspCGRAs 的单元阵列结构及其共享资源

动体系结构的优势, 把对循环的控制移至对存储体操作的控制上, 通过控制存储体操作来控制数据流的流动, 从而控制着整个计算的正确运行。mPE 成了数据流的阀门, 前两个 mPE 控制数据流的流入, 后一个 mPE 控制数据流的流出。当处理单元阵列都被数据流充满时, 吞吐率将达到每拍一个结果, 即一拍便执行完循环的一次迭代。该结构没有额外的同步机制控制数据流的流动, 却能使循环得到高效的执行, 并行度得到最大程度的开发, 如图 4 所示。

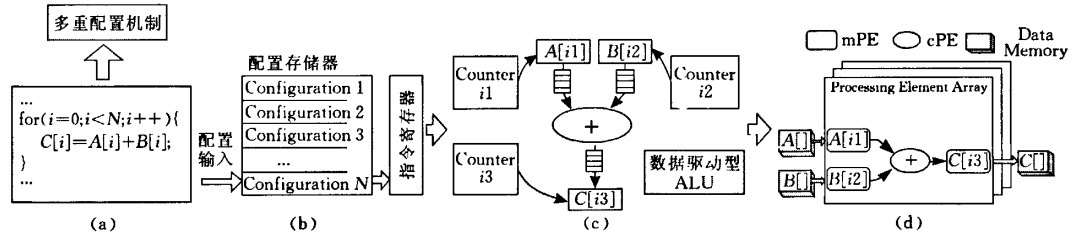


图 4 循环自动流水化

此外, 为支持数据密集型应用的循环流水, lspCGRAs 体系结构使用猜测执行来提高循环流水线适应条件判断的能力, 同时利用数据重用来提高对存储资源的利用。

4.2 lspCGRAs 模板规范

通常, CGRAs 具有不同的设计实现细节, 如阵列拓扑、通信协议与重构策略等, 以满足特定领域应用的不同需要。为更好地支持数据密集型应用到 lspCGRAs 的映射, 我们定制了 DIA-lspCGRAs 模板, 该模板具有参数化的特征, 通过不同的参数配置能够较好地表示 DIA 和 lspCGRAs。DIA-lspCGRAs 模板主要由一般描述 Header、领域应用描述 DOMAIN_DIA 和体系结构描述 ARCH_lspCGRAs 构成, 如图 5 所示。

```
class Template{
    //DIA-lspCGRAs template
    Header{
        //一般描述
        name DIA-lspCGRAs_template;
        description "xxx";
        path "X:/XX/XX...";
        author "xxx xxx";
        version "X.X.X";
        date "XXXX-XX-XX";
    }
    DOMAIN_DIA{
        //领域应用-数据密集型应用
        PIA-CDTG; //程序特征信息
        vi-DFG; //循环特征
        Data-Info; //循环内的数据信息
    }
    ARCH_lspCGRAs{
        //体系结构-lspCGRAs
        Loop_Pipeline; //循环流水
        Constraint_model; //约束模型
        Resource_model; //资源模型
        Performance_model; //性能模型
    }
}
```

图 5 DIA-lspCGRAs 模板的定义规范

其中, DOMAIN-DIA 描述数据密集型应用的程序特征信息、循环模型以及循环内的数据信息. Data-Info 提供了 vi-DFG 中所处理的数据的相关信息, 如数据重用性、滑动窗口特性等, 这些信息可以为映射提供有效的优化建议. 例如, 针对应用的特点进行存储调度使其能充分利用局部存储器的带宽并使其性能达到最优. 在进行 FFT 算法映射时, 使读写操作能在双存储体间快速切换; 在映射中值滤波和边缘检测算法时, 充分利用数据的相关性达到数据重用, 并通过计算和数据传递重叠执行隐藏数据传递时间等.

ARCH-lspCGRAs 描述了 lspCGAs 的循环流水配置、约束模型、资源模型和性能模型. 循环流水配置描述对循环的配置输入到配置存储器中, 该配置决定了对 PE 单元的功能和互连网络的数据交换方式. 约束模型包括了第 3 节中所描述的 mPE_c , cPE_c , LM_c 和 $LBus_c$. 资源模型描述了 lspCGRAs 中的资源配置情况, 如 PE 单元、LM、SB 和数据带宽等. 性能模型描述了 lspCGRAs 中的 mPE、cPE、路由单元和 $m \times n$ 阵列的 FPGA 综合结果, 例如一个 mPE 占用的逻辑单元、存储量和时钟频率等.

5 映射优化方法

5.1 领域应用驱动的映射流程

我们使用领域应用驱动的映射方法将 PIA-CDTG 映射到 lspCGRAs.

该方法使用应用特征分析技术, 尽早分析领域应用的多种算法的特征, 并提取出应用算法中的关键循环块, 将其循环流水化实现, 同时通过暂时划分和设计空间探索以进行应用到体系结构的映射优化, 如图 6 所示. 该流程中主要包括 5 个任务:

(1) 应用算法的特征分析与提取

使用应用程序预处理将领域应用算法转换成适合分析处理的中间格式, 应用特征分析过程对输入的应用算法进行特征分析, 得到其特征信息, 如控制流/数据流、计算/存储分布、关键代码块、依赖关系等特征, 并获取任务的统计特性, 自动构建应用任务模型 PIA-CDTG. 特定应用的特征分析结果和系统模型分别保存在应用特征信息库和系统模型库中, 以支持设计重用.

(2) 获取关键循环块

在应用算法的特征分析与提取的基础上, 根据应用在计算时间和存储空间上的需求, 来识别和提

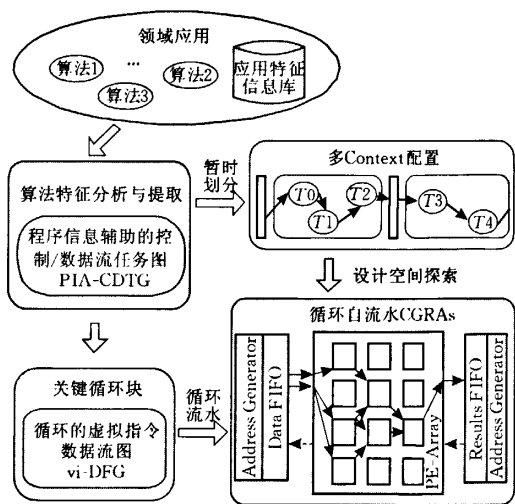


图 6 领域应用驱动的映射流程

取关键代码块。

首先, 我们将应用程序代码划分为基本块, 并构建以基本块为节点的任务流图. 通常任务流图有 3 种粒度: 指令级、基本块级和过程级. 我们采用基本块级, 其中, 基本块可以是普通语句块、循环或函数. 基本块可以是复合的, 包含子语句块、子循环或子函数. 而整个程序源代码由多个基本块构成层次式任务流图, 即复合基本块的内部也包含子任务流图.

其次, 根据应用算法的特征分析结果, 分析和统计每个基本块的特性, 如表 1 所示.

表 1 基本块的特征分析数据

	函数	循环	语句块
名字	✓	✓	✓
迭代次数	×	✓	×
调用次数	✓	✓	✓
指令数量	✓	✓	✓
每种指令的数量	✓	✓	✓
执行频率	✓	✓	✓
执行时间	✓	✓	✓
执行时间占总量的百分比	✓	✓	✓
存储需求	✓	✓	✓
存储需求占总量的百分比	✓	✓	✓

然后, 根据基本块的特征分析统计结果, 将执行时间百分比和存储需求百分比大于某个阈值的基本块作为关键代码块.

对于得到的关键循环块, 使用 if-conversion 技术, 将其控制流图转换为数据流图, 并将循环标准化:

1. 将循环初值、终值和步进值确定为常数;
2. 消除 break、continue 语句;

3. 消除间接取址操作;

4. 寄存器重命名,消除部分写后读相关。

然后根据 Kahn 进程定义的虚拟指令,构建该关键循环块的 vi-DFG,为循环流水化实现做准备。

(3) 暂时划分

通常, Spatial 映射受限于阵列的大小和拓扑结构。为了在有限资源的 lspCGRAs 上运行多个复杂的应用算法,我们采用暂时划分技术,将多个应用算法分解为子块,然后将每个子块分别配置到可重构阵列上调度执行,如图 7 所示。而暂时划分通常受配置存储器大小和 lspCGRAs 资源的限制。

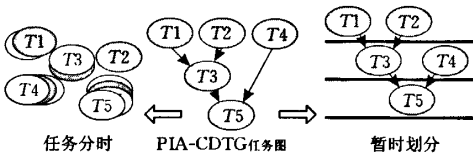


图 7 将 PIA-CDTG 暂时划分到多 Context 配置

根据应用特征分析提供的 PIA-CDTG 特征信息,如执行时间百分比、最大活跃变量的存储需求、目标代码的存储需求、节点间的数据通信量、节点的并行性以及节点的先后次序关系等,指导暂时划分,同时,使用应用相关性分析并将相关度较大的任务尽可能分簇到同一个子块内,以减少子块任务间的数据交换和通信开销。

此外,在进行暂时划分的时候, lspCGRAs 的资源情况影响着关键块的粒度选择:如果其资源充足,关键块可以较大一些;如果其资源有限,关键块需较小,对于关键块是多重循环的情况,需要对该多重循环进行分解,以转换为较小的循环块进行到 lspCGRAs 的映射。

(4) 循环流水

对于每个 vi-DFG,将其循环流水化实现在 lspCGRAs 上。LEAP 体系结构通过流水线的执行方式把循环映射在 PE 阵列上,循环程序自身被分解为多个部分由各个 PE 实现:

1. 循环状态机对应于 For 语句中的循环控制条件;
2. 运算阵列对应于 For 语句中循环体;
3. 循环状态机控制着从存储器中取数和存数;
4. 运算阵列以流水线的方式对存储器中取出的数据进行运算。

循环的这种流水线执行方式,由分布的 mPE 和 cPE 配合完成, mPE 提供灵活的存储器调度方式,而 cPE 提供强大的计算能力。 mPE 中循环控制变量 index value 顺序变化,根据循环下标不断地从 LM 中取得数据,并把数据传递给 cPE 阵列,

cPE 阵列按照固定的连接关系进行运算,运算的结果存储到 mPE 指定的地址。使用 lspCGRAs 体系结构处理循环,可以达到很高的吞吐率。图 8 给出对某一循环的十次迭代的流水线时空图,其中, PE 在执行循环时功能固定,形成静态的数据流图,流水线充满的状态同时也说明 PE 达到了很大的利用率。

	1	2	3	4	5	6	7	8	9	10
PE0	LD	LD	LD	LD	LD	LD	LD	LD		
PE1	LD	LD	LD	LD	LD	LD	LD	LD		
PE2		+	+	+	+	+	+	+	+	
PE3			ST	ST	ST	ST	ST	ST	ST	ST

图 8 循环 10 次迭代的流水线时空图

(5) 设计空间探索

在进行特定应用(以暂时划分后的子块为单位)的映射决策时,往往不存在唯一的映射,而是一个由多个设计因素折衷的设计空间,需要探索该空间的优化解。

对于给定的领域应用的多个算法,我们探索 lspCGRAs 体系结构模板的参数:

- ① mPE 和 cPE 资源数量;
- ② 局部存储器的大小与资源数量;
- ③ 输入数据带宽、输出数据带宽;
- ④ 智能缓冲区的大小;
- ⑤ 流水线的划分与栈数。

在进行特定应用的映射时,使用设计空间探索可以协调计算、存储、通信以及其它资源与应用需求之间的协调与折衷。同时,我们根据系统开销与应用性能的需求,提取出关键代码块,针对关键代码块进行设计优化,该策略可有效减少设计空间,提高设计效率与设计质量。

5.2 核心循环流水映射: Loop-Kernel Pipelining Mapping(LKPM)

对于核心循环块,我们使用 LKPM 方法将其映射到 lspCGRAs 上,如图 9 所示。

该方法主要有 3 个部分构成。

(1) 资源需求分析

分析 vi-DFG 中核心循环的运算,找到其中存储操作的节点 s 个(设运算操作的输入数据为 s_1 个,相关性分析后的最少输入数据为 s_2),其它的计算操作的节点 t 个。而 $m \times n$ 的 lspCGRAs 中 mPE 为 n 个, cPE 为 $(m-1) \times n$ 个。其中, $t < (m-1) \times n$ 。

设置目标函数如下:

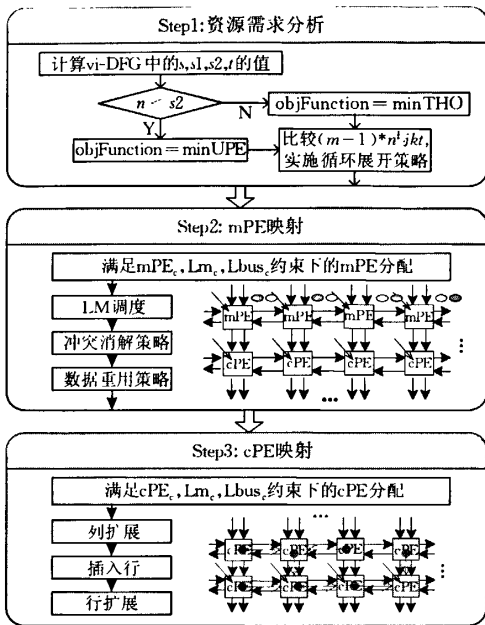


图 9 LKPM 方法: Loop-Kernel Pipelining Mapping

$$objFunction = \begin{cases} minUPE, & n < s_2 \\ maxTHO, (m-1) \times n, & n \geq s_2 \end{cases}$$

即如果 $n < s_1$, 设置目标函数为 $minUPE$; 如果 $n \geq s_1$, 设置目标函数为 $maxTHO$.

在 $maxTHO$ 目标下, 如果 $(m-1) \times n > k_t$, 可将多重循环 k 次展开, 如果 k 次展开后 mPE 和 cPE 映射失败, 则将多重循环 $k-1$ 次展开, 以此类推, 直到成功进行 mPE 和 cPE 映射为止.

循环分解流水

为了增强循环流水的吞吐量, 在资源充足的情况下, 可以将多重循环的外层循环分解, 以同时并行的处理多片内层循环, 如图 10 所示^[16]. 该 Fir 循环源代码的内层循环块由数组的一次乘加操作构成, 映射到 lspCGRAs 体系结构上循环流水化实现. 通过将外层循环 j 变量分解, 每次内层循环可同时执

行两次乘加操作. 因此, 在映射时, 根据 lspCGRAs 的资源适当地进行循环分解.

(2) mPE 映射

由于 lspCGRAs 体系结构中 cPE 和 mPE 实现了计算和存储的分离, 因此在进行核心循环映射时, 首先要将循环运算操作的输入数据逐一分配到 mPE 上, 并尽可能减少数据传送到 cPE 上的时间.

在该步骤中, 为了使计算和数据传递重叠执行, 隐藏数据传递时间, 在相邻 mPE 之间调度 LM 时, 保证其在流水线同一栈内的使用不冲突. 即当前栈中 cPE 使用的 LM, 其相邻的 LM 分配给 mPE 使用. 而在进行 LM 分配之前, 流水线第一栈输入数据所需的 LM 默认被 cPE 占用.

进行 mPE 映射时, 通常考虑两种策略: 冲突消解、数据重用.

冲突消解

在将核心循环块的数据映射到存储资源的时候, 尽可能避免数据访问的冲突. 通常存在两种数据访问冲突: 并行冲突和自冲突.

对于循环中的代码(如 Fir 循环):

$$tmp1 = S[i+j] \times C[i].$$

如果 S 和 C 的数据放在同一个存储资源上(如 cPE 的 FIFO 队列)而产生结构相关, 称为并行冲突.

对于循环中的代码(如卷积运算):

$$y = X[i] \times X[k-i].$$

如果 X 的数据放在同一个存储资源上而产生结构相关, 称为自冲突. 因此, 对于存在并行冲突和自冲突的数据, 在进行循环流水化映射时, 将其分配到不同的存储资源上, 使其可并行加载到 cPE 的 FIFO 不同队列上.

数据重用

绝大多数数据密集型应用, 如流处理、滑动窗口应用中, 存在着大量的数据重用. 对于这类应用的多重循环的映射, 我们将充分开发循环层内和循环层

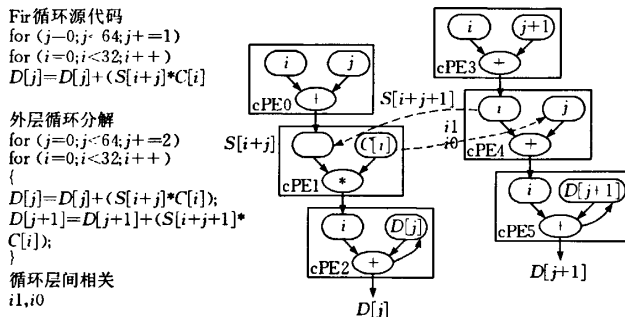


图 10 循环分解流水的数据流图表示

间的数据重用,以减少从外部存储器重新读入这些数据的时间。

对于循环层内数据重用,我们将这些数据保存到寄存器中。如

```
for (j=0; j<m; j=j+1)
    B[j]=A[j]+A[j+1]+C;
```

而对于循环层间的数据重用,我们将这些数据保存到LM中,并将当前处理的滑动窗口的数据保存到智能缓冲区内。如

```
for (i=0; i<n; i=i+1)
    for (j=0; j<m; j=j+1)
        B[i][j]=A[i][j]+A[i+1][j]+C;
```

(3) cPE映射

使用图论中 Split-Push 方法进行 cPE 映射,但有点与 SPKM 不同的设置之处^[5]。

cPEs 互连

在 SPKM 中,假设 PE 可以与其相距 2 个的水平与垂直的邻居 PE 直接互连,而我们的 LKPM 方法中,cPE 只与其相距 1 个的水平与垂直的邻居 cPE 直接互连。

mPE 调度

在 SPKM 中,假设同行最多有 2 个 LD 和 1 个

ST 操作,而我们的 LKPM 中,不存在该限制,只需满足 mPE_c 和 LM_c 约束即可。

ILP 匹配优化目标

在 SPKM 中,在进行列扩展时,将 ILP 匹配中的优化目标定义为 $|UR|_{\min}$,而我们的 LKPM 中,根据 *objFunction* 将优化目标定义为 $|UPE|_{\min}$ 或 $|THO|_{\max}$ 。

其它方面的设置基本上与 SPKM 相同。

6 实验及结果分析

我们在所开发的 lspCGRAs 体系结构——LEAP 上,设计实现了一些数据密集型算法^[14]。LEAP 体系结构采用循环自流水技术,将循环控制条件映射到 mPE,将循环体映射到 cPE 组成的运算阵列上执行;同时 LEAP 作为协处理器,通过接口控制器与微处理器主机互连通信^[17]。对于一些典型的数据密集型算法,如 FFT、Sobel Edge Detection、Median Filter、Matrix Multiply 等,WGM^[18]已进行了手工映射与优化。在该工作的基础上,我们使用本文方法进行了自动映射,其性能结果如表 2 所示。

表 2 典型数据密集型应用在 lspCGRAs 上映射的性能

算法	UPE		THO (OP/cycle)	AP/%	ECycle/K	#LDelay
	mPE	cPE				
512 点 FFT	4	10(10)	3.9(4.0)	39.3(40.3)	26.2(25.6)	34(34)
Edge Detection (320×240)	7	16(16)	6.3(6.6)	39.6(41.5)	227(217)	54(54)
Median Filter (320×240)	7	30(30)	12.4(12.7)	41.3(42.3)	225(220)	54(54)
Matrix Multiply (64×64)	10	30(16)	12.5(6.7)	40.7(41.6)	42.4(79.1)	45(24)

我们测试了典型数据密集型应用算法在 lspCGRAs 上映射的性能。表 2 分别给出了每个算法所占用的资源 cPE、mPE、吞吐量 THO、计算活跃度 AP、执行节拍数 ECycle 以及流水线启动延迟 #LDelay。其中,括号内的数据表示手工映射优化所占用的 cPE 数量和吞吐量。可见,我们的自动映射方法与手工优化的结果相比,差别很小。

从表 2 中最后一行的 MM 算法映射可以看到,UPE 与 THO 这两个目标不能同时达到最优,占用的 cPE 数越多,理想情况下最高吞吐量等于 #cPE,从而 THO 也越高。在我们的方法中,考虑到这种多个应用算法同时映射时的多目标优化的特性,使用设计空间探索技术进行折衷。同时,表 2 中第 5 列活跃度 AP 表示 cPE 参与运算的时间占总运行时间的百分比,在实际映射时我们考虑了计算与存储时间的相重叠,但 AP 也能间接反映出存储访问的延迟是影响整体 THO 的重要因素。可见,存储访问的优化对于提高循环流水线的效率非常重要,而在我

们的方法中通过采用冲突消解和数据重用策略,尽可能在满足访存约束条件下达到最低的访存次数。

为阐述 LKPM 方法的有效性,我们选取国际上先进的 SPKM 方法进行比较实验,在 Benchmarks 上选择表 2 中的 4 个典型数据密集型应用算法,图 11 和图 12 分别给出了算法映射的资源利用率 UPE 和吞吐量 THO。由于 SPKM 只考虑计算 PE 的映射,没有考虑存储 PE 的映射,导致存储访问延迟很高,因此其吞吐量比 LKPM 低很多;同时,由于 LKPM 中假设每行 PE 只能使用 2 次 LD 和 1 次 ST,存储带宽成为瓶颈,导致映射需要的 PE 行数增加,因而其 UPE 也比 LKPM 低一些。此外,由于 lspCGRAs 的 LM 使用双端口存储器,映射 FFT 算法时读写操作能在双存储体间快速切换,节省了存储资源,从而提高了 UPE。ED 和 MF 存在着很多数据相关和数据重用,不仅减少了 PE 行数,还减少访存而提高了吞吐量,因此它们的 LKPM 映射效果比 SPKM 高很多。总体看来,我们的 LKPM 方法与

SPKM 相比,平均资源占用率 UPE 减少了 16.3%, 平均吞吐率 THO 提高了 169.1%。

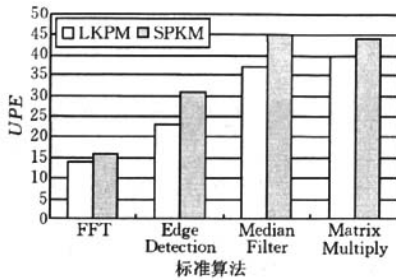


图 11 算法映射的资源占用率 UPE

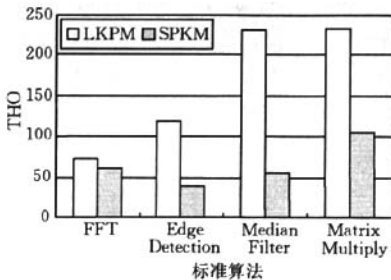


图 12 算法映射的吞吐量 THO

为了量化的评价 LKPM 相比于 SPKM 的映射质量效果,我们随机生成 40 个循环类的应用 RAG 图,以 RAG 图节点规模大小从 10~16 进行实验,在 40 个应用 RAG 图中,按每组 10 个分成四组不同类型的应用:循环层间数据重用型、循环层内数据重用型、数据访问冲突型和无数据相关型,如图 13 所示。通过测试,根据资源占用率 UPE 参数比较两种方法的映射质量。可见,从平均效果来看,在 40 个应用的映射中,LKPM 比 SPKM 效果更好的有 31 个,效果相当的有 6 个,即在 40 个应用的映射中 LKPM 高于 SPKM 的占 75.6%。

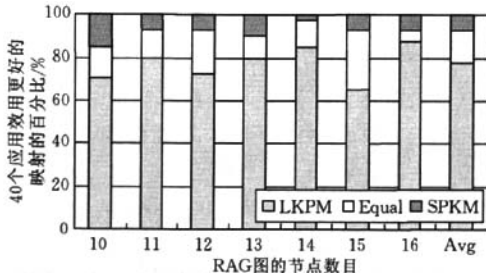


图 13 LKPM 与 SPKM 方法的映射质量百分比

7 结 论

CGRAs 可提供灵活而有效的平台来开发复杂

的应用,而研究数据密集型应用到 CGRAs 的映射仍旧是一个富有挑战性的艰巨任务。为了减少这种映射复杂性和充分开发数据密集型应用在 CGRAs 上的性能,本文提出了一种有效的核心循环到 $lspCGRAs$ 的流水映射方法,通过将核心循环块的代码实现其自流水化执行,获得了高计算吞吐量。

在定义了核心循环流水映射问题后,我们阐述了体系结构模板中的资源共享与流水特征以及 DIA- $lspCGRAs$ 模板规范;同时,给出了领域应用驱动的映射流程以及核心循环流水映射 LKPM 方法。典型应用算法映射的实验结果表明,本文的自动映射方法与手工映射优化的性能相差很小;同时,与已有的先进的 SPKM 方法相比,我们的方法的资源占用率降低 16.3%、吞吐量提高 169.1%。

下一步的工作我们将进行复杂的数据密集型应用系统的自动映射实验,在 $lspCGRAs$ 上进行更多的 Benchmarks 实验。

参 考 文 献

- [1] Katherine Compton, Scott Hauck. Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys*, 2002, 34(2): 171-210
- [2] Hartenstein R. A decade of reconfigurable computing: A visionary retrospective//*Proceedings of the DATE2001*. Munich, Germany, 2001; 642-649
- [3] Shrivastava Aviral, Earlie Eugene, Dutt Nikil, Nicolau Alex. PBEExplore: A framework for compiler-in-the-loop exploration of partial bypassing in embedded processors//*Proceedings of the DATE 2005*. Washington, 2005; 1264-1269
- [4] Fan Kevin, Kudlur Manjunath, Park Hyunchul, Mahlke Scott A. Increasing hardware efficiency with multifunction loop accelerators//*Proceedings of the CODES+ISSS*. Seoul, Korea, 2006; 276-281
- [5] Yoon Jonghee W, Shrivastava Aviral, Park Sanghyun, Ahn Minwook, Jeyapaul Reiley, Paek Yunheung. SPKM: A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures//*Proceedings of the ASP-DAC*. Seoul, Korea, 2008; 776-782
- [6] Lee J, Choi K et al. Mapping loops on coarse-grain reconfigurable architectures using memory operation sharing. *Center for Embedded Computer Systems, University of California, Irvine*; TR 02-34, 2002
- [7] Kejariwal Arun, Veidenbaum Alexander V, Nicolau Alexandru, Girkar Milind, Tian Xinmin, Saito Hideki. Challenges in exploitation of loop parallelism in embedded applications//*Proceedings of the CODES + ISSS*. Seoul, Korea, 2006; 173-180
- [8] Baumgarten Volker, Ehlers G, May F, Nuckel Armin, Vorbach Martin, Weinhardt Markus. PACT XPP—A self-reconfigurable data processing architecture. *The Journal of Supercomputing*, 2003, 26(2): 167-184

- [9] Singh H, Lee M, Lu G, Kurdahi F J, Bagherzadeh N, Filho E, Maestre R. Morhposys: Case study of a reconfigurable computing system targeting multimedia applications//Proceedings of the DAC'00. Los Angeles, California, 2000; 573-578
- [10] Hu Qubo, Vandecappelle Arnout, Palkovic Martin et al. Hierarchical memory size estimation for loop fusion and loop shifting in data-dominated applications//Proceedings of the ASP-DAC 2006. Yokohama, Japan, 2006; 606-611
- [11] Lee Jong-eun, Choi Kiyong, Dutt Nikil D. An algorithm for mapping loops onto coarse-grained reconfigurable architectures//Proceedings of the LCTES 2003. San Diego, CA, 2003; 183-188
- [12] Rong Hongbo, Tang Zhizhong, Govindarajan Ramaswamy, Douillet Alban, Gao Guang R. Single-dimension software pipelining for multidimensional loops//TACO. 2007, 4(1); 1-4
- [13] Kim Yoonjin, Kiemb Mary, Park Chulsoo, Jung Jinyong, Choi Kiyong. Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization//Proceedings of the DATE 2005. Munich, Germany, 2005; 12-17
- [14] Ahn Minwook, Yoon Jonghee W, Paek Yunheung, Kim Yoonjin, Kiemb Mary, Choi Kiyong. A spatial mapping algorithm for heterogeneous coarse-grained reconfigurable architectures//Proceedings of the DATE 2006. Munich, Germany, 2006; 363-368
- [15] Wang Da-Wei, Li Si-Kun, Dou Yong. Collaborative hardware/software partition of coarse-grained reconfigurable system using evolutionary ant colony optimization//Proceedings of the ASP-DAC 2008. Seoul, Korea, 2008
- [16] Dou Yong, Lu Xicheng. LEAP: A data driven loop engine on array processor//Proceedings of the APPT 2003. Xiamen, China, 2003; 12-22
- [17] Dou Yong, Wu Gui-Ming, Xu Jinhui, Zhou Xingming. A coarse-grained reconfigurable computing architecture with loop self-pipelining. Science in China, 2008, 38(4); 579-591 (in Chinese)
(窦勇, 邬贵明, 徐进辉, 周兴铭. 支持循环自流水的粗粒可重构阵列体系结构, 中国科学, 2008, 38(4); 579-591)
- [18] Wu Guiming. Research on coarse-grain reconfigurable architecture base on loop pipelining[M. S. dissertation]. National University of Defense Technology, Changsha, 2006(in Chinese)
(邬贵明. 粗粒度可重构体系结构研究[硕士学位论文]. 国防科学技术大学, 长沙, 2006)



WANG Da-Wei, born in 1980, Ph. D. candidate. His research interests include SoC system level design, transaction level design and electric design automation.

DOU Yong, born in 1966, Ph. D., professor. His research interests include architecture design, high performance compiling.

LI Si-Kun, born in 1941, professor, Ph. D. supervisor. His research interests include SoC design methodology, electric design automation, computer-aided design.

Background




The problem this paper aims to solve belongs to high level design of reconfigurable SoC. Currently, reconfigurable computing has the efficiency of custom computation and flexibility of common computation, so it can accelerate most kinds of applications. CGRA can provide more speedup and less energy cost. However, although CGRA can provide high performance and flexibility, it still remains a hard problem to meeting the severe constraints on performance and cost while developing some field specific applications.

It needs high throughput and parallel to handle DIA, while it is inclined to errors and time-consuming to map DIA onto CGRA manually. Besides, there are many loop kernels which consume much system resources. It's necessary to speed up and optimize these critical loop kernels. Due to the insufficient work of mapping loop kernels onto CGRAs, this paper proposes a novel loop kernel pipelining mapping onto coarse-grained reconfigurable architectures. The authors developed a loop self-pipelining coarse-grained reconfigurable architecture. The lspCGRA uses fix instruction multiple dataflow, which can support loop self-pipelining and gain high

computation throughput. Besides, the authors show the details about the share resource and pipelining features of lspCGRA templates. A novel loop kernel pipelining mapping approach is presented together.

This paper is partly supported by the National Natural Science Foundation of China under project "Research on fixed instruction and multi-data stream computation model based coarse grained reconfigurable array architecture" with grant No. 90307001 and "Embedded Stream Media Process SOC Design Platform and Design Technology" with grant No. 90707003. With the support of above project, this research groups have developed 32bit RISC microprocessor and coarse grained reconfigurable architecture that accelerates applications through loop self-pipelining technique.

This paper contributes on the field of high level design to support the whole SoC design flow and optimization. The approach of loop kernel pipelining mapping is proposed with its automatic design flow. It shows less resource occupation by 16.3% and higher throughput by 169.1% than advanced SPKM.

作者: 王大伟, 窦勇, 李思昆, WANG Da-Wei, DOU Yong, LI Si-Kun
作者单位: 国防科学技术大学计算机学院软件所, 长沙, 410073
刊名: 计算机学报   
英文刊名: CHINESE JOURNAL OF COMPUTERS
年, 卷(期): 2009, 32(6)
被引用次数: 7次

参考文献(18条)

1. KATHERINE COMPTON; SCOTT HAUCK [Reconfigurable Computing: A Survey of Systems and Software](#) [外文期刊] 2002(2)
2. Hartenstein R [A decade of reconfigurable computing: A visionary retrospective](#) 2001
3. Shrivastava Aviral; Earlie Eugene; Dutt Nikil; Nicolau Alex [PBExplore: A framework for compiler-in-the-loop exploration of partial bypassing in embedded processors](#) 2005
4. Fan Kevin; Kudlur ManAunath; Park Hyunchul; Mahlke Scott A [Increasing hardware efficiency with multifunction loop accelerators](#) 2006
5. Yoon Aonghee W; Shrivastava Aviral; Park Sanghyun; Ahn Minwook, Aeyapaul Reiley, Paek Yunheung [SPKM: A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures](#) 2008
6. Lee J; Choi K [Mapping loops on coarse-grain reconfigurable architectures using memory operation sharing](#). Center for Embedded Computer Systems 2002
7. KeAariwal Arun; Veidenbaum Alexander V; Nicolau Alexandru; Girkar Milind, Tian Xinmin, Saito Hideki [Challenges in exploitation of loop parallelism in embedded applications](#) 2006
8. Baumgarten Volker; Ehlers G; May F; Nuckel Armin Vorbach Martin Weinhardt Markus [PACT XPP-A selfreconfigurable data processing architecture](#) 2003(02)
9. Singh H; Lee M; Lu G; Kurdahi F A, Bagherzadeh N, Filho E, Maestre R [Morhposys: Case study of a reconfigurable computing system targeting multimedia applications](#) 2000
10. Hu Qubo; Vandecappelle Arnout; Palkovic Martin [Hierarchical memory size estimation for loop fusion and loop shifting in data-dominated applications](#) 2006
11. Lee Aong-eun; Choi Kiyong; Dutt Nikil D [An algorithm for mapping loops onto coarse-grained reconfigurable architectures](#) 2003
12. Rong Hongbo; Tang Zhizhong [GovindaraAan Ramaswamy, Douillet Alban, Gao Guang R. Single-dimension software pipelining for multidimensional loops//TACO](#) 2007(01)
13. Kim YoonAin; Kiemb Mary; Park Chulsoo; Aung Ainyong, Choi Kiyong [Resource sharing and pipelining in coarsegrained reconfigurable architecture for domain-specific optimization](#) 2005
14. Ahn Minwook; Yoon Aonghee W; Paek Yunheung; Kim YoonAin, Kiemb Mary, Choi Kiyong [A spatial mapping algorithm for heterogeneous coarse-grained reconfigurable architectures](#) 2006
15. Wang Da-Wei; Li Si-Kun; Dou Yong [Collaborative hardware/software partition of coarse-grained reconfigurable system using evolutionary ant colony optimization](#) 2008
16. Dou Yong; Lu Xicheng [LEAP: A data driven loop engine on array processor](#) 2003

17. [窦勇, 邬贵明, 徐进辉, 周兴铭](#) [支持循环自动流水线的粗粒度可重构阵列体系结构](#) [期刊论文] - [中国科学E辑](#) 2008 (4)
18. [邬贵明](#) [粗粒度可重构体系结构研究](#) 2006

本文读者也读过(10条)

1. [周学功, 梁樑, 黄勋章, 彭澄廉](#). [ZHOU Xue-Gong, LIANG Liang, HUANG Xun-Zhang, PENG Cheng-Lian](#) [可重构系统中的实时任务在线调度与放置算法](#) [期刊论文] - [计算机学报](#) 2007, 30 (11)
2. [徐进辉, 杨梦梦, 窦勇, 周兴铭](#). [XU Jin-Hui, YANG Meng-Meng, DOU Yong, ZHOU Xing-Ming](#) [粗粒度可重构平台中循环自流水硬件实现](#) [期刊论文] - [计算机学报](#) 2009, 32 (6)
3. [王新安, 叶兆华, 戴鹏, 周丹](#). [WANG Xin-an, YE Zhao-hua, DAI Peng, ZHOU Dan](#) [可重构阵列DSP结构ReMAP](#) [期刊论文] - [深圳大学学报 \(理工版\)](#) 2010, 27 (1)
4. [周学海, 罗赛, 王峰, 齐骥](#). [ZHOU Xue-hai, LUO Sai, WANG Feng, QI Ji](#) [一种数据驱动的可重构计算统一编程模型](#) [期刊论文] - [电子学报](#) 2007, 35 (11)
5. [王颖, 周学功, 游红俊, 彭澄廉](#). [Wang Ying, Zhou Xuegong, You Hongjun, Peng Chenglian](#) [支持动态可重构的软/硬件统一多线程编程模型](#) [期刊论文] - [计算机辅助设计与图形学学报](#) 2009, 21 (6)
6. [王颖, 陈伟男, 周学功, 彭澄廉](#). [WANG Ying, CHEN Wei-nan, ZHOU Xue-gong, PENG Cheng-lian](#) [可重构计算中的负载可分应用性能分析与预测](#) [期刊论文] - [小型微型计算机系统](#) 2010, 31 (8)
7. [王晟中, 陈伟男, 彭澄廉](#). [WANG Sheng-zhong, CHEN Wei-nan, PENG Cheng-lian](#) [可重构计算硬件平台的改进设计](#) [期刊论文] - [计算机工程](#) 2010, 36 (5)
8. [齐骥, 李曦, 于海晨, 胡楠, 龚育昌, 王立刚](#). [Qi Ji, Li Xi, Yu Haichen, Hu Nan, Gong Yuchang, Wang Ligang](#) [一种面向动态可重构计算的调度算法](#) [期刊论文] - [计算机研究与发展](#) 2007, 44 (8)
9. [孟涛, 戴紫彬](#). [Meng Tao, Dai Zi-bin](#) [分组密码处理器的可重构分簇式架构](#) [期刊论文] - [电子与信息学报](#) 2009, 31 (2)
10. [刘彦, 李仁发, 许新达, 徐成](#). [Liu Yan, Li Renfa, Xu Xinda, Xu Cheng](#) [一种异构可重构片上系统的实时任务调度算法](#) [期刊论文] - [计算机研究与发展](#) 2010, 47 (6)

引证文献(7条)

1. [杨子煜, 赵鹏, 王大伟, 李思昆](#) [关键循环到粗粒度可重构体系结构的存储感知映射](#) [期刊论文] - [国防科技大学学报](#) 2012 (06)
2. [陈乃金, 江建慧](#) [融合面积估算和多目标优化的硬件任务划分算法](#) [期刊论文] - [通信学报](#) 2013 (02)
3. [杨子煜, 严明, 王大伟, 李思昆](#) [面向CGRA循环流水映射的数据并行优化](#) [期刊论文] - [计算机学报](#) 2013 (06)
4. [杨子煜, 李思昆, 赵鹏](#) [虚拟环境下可重构流媒体处理的应用分析方法](#) [期刊论文] - [系统仿真学报](#) 2012 (09)
5. [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](#) [期刊论文] - [通信学报](#) 2015 (4)
6. [王海峰, 陈庆奎](#) [静态程序切片的GPU通用计算功耗预测模型](#) [期刊论文] - [软件学报](#) 2013 (08)
7. [王海峰, 陈庆奎](#) [静态程序切片的GPU通用计算功耗预测模型](#) [期刊论文] - [软件学报](#) 2013 (08)

引用本文格式: [王大伟, 窦勇, 李思昆](#). [WANG Da-Wei, DOU Yong, LI Si-Kun](#) [核心循环到粗粒度可重构体系结构的流水化映射](#) [期刊论文] - [计算机学报](#) 2009 (6)