面向SoC任务分配的应用程序存储需求量分析方法

赵 鹏,王大伟,李思昆
(国防科技大学计算机学院,湖南长沙 410073)

摘 要: 通过分析应用程序的存储需求量,辅助片上系统(System-on-Chip,SoC)任务分配进行数据存储与传输优化,是改善SoC性能的途径之一.目前,存储需求量分析方法的分析粒度单一、速度缓慢,不利于进行多粒度任务分配空间的高效探索.本文面向SoC任务分配,提出一种多粒度、快速存储需求量分析方法.该方法可进行多粒度的存储需求量分析,并针对多媒体程序,引入数组数据域划分技巧,大幅度减少了存储需求量的分析时间.

关键词: 片上系统; 任务分配; 存储需求量分析; 多面体模型
中图分类号: TP391; TN4 **文献标识码:** A **文章编号:** 0372-2112 (2010) 03-0541-05

Research on Memory Size Estimation of Application Programs for System-on-Chip Task Allocation

ZHAO Peng[1], WANG Da-wei[1], LI Si-kun[1]
(1. *School of Computer, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: System-on-Chip (SoC) task allocation with memory size information can improve the data storage and transfer efficiency. Nowadays, the methods of memory size estimation often adopt the single granularity and run at slow speed. Therefore, the space exploration of multi-granularity task allocation cannot be supported well. For SoC task allocation, this paper proposes a fast, multi-granularity approach of memory size estimation. The approach estimates the memory size at three granularities. The technique of data domain partition is introduced into the time-consuming process of memory size estimation at loop granularity, which greatly accelerates the analysis processing.

Key words: System-on-Chip (SoC); task allocation; memory size estimation; polyhedron model

1 引言

多媒体处理程序往往具有大量的嵌套循环和频繁的数据存储与传输,对存储系统的容量和性能要求较高.而SoC的存储资源比较紧缺且成本昂贵.如何分析并利用多媒体处理程序的存储需求量信息,进行SoC系统的数据传输与存储性能优化,已经成为影响SoC系统性能和功耗的重要问题之一.

多媒体处理SoC普遍采用"主处理器+多个协处理器"的主流体系结构.该体系结构一般包含多个异构的处理器,对SoC任务分配方法提出了更高的要求.SoC任务分配是在满足系统约束的条件下,以各任务的特征信息(例如:计算时间、访存时间和存储需求量等)为分配依据,将应用中的各个任务分配到SoC的各个处理器上.在任务分配依据中,存储需求量信息与数据存储传输性能密切相关,对于提高任务分配质量与SoC性能有着重要作用.

本文针对多媒体处理程序,提出了一种面向SoC任务分配的多粒度、快速存储需求量分析方法.该方法在基本块、循环和函数粒度上,分别使用静态程序分析技术(包括:控制流分析、数据活跃性与相关性分析等)提取与存储需求量有关的信息(包括:控制流、数据活跃性与相关性等),并基于这些信息计算程序存储需求量.所得到的存储需求信息,作为分配依据输入任务分配算法,辅助任务分配寻找高质量分配方案,从而提高多媒体SoC性能.

2 相关工作

本文分析存储需求量的目标是:获取程序正常运行所需的最小和最大存储需求量.其中,最小存储需求量

收稿日期:2008-04-09;修回日期:2009-10-26
基金项目:国家自然科学基金(No.90207019);国家自然科学基金(No.90707003)

更为重要,且分析难度较大,是现有的研究重点[1~5].本文也重点分析最小存储需求量.

最小存储需求量分析方法可以分成两大类:动态分析方法和静态分析法.静态分析方法与程序的输入数据无关,分析结果较为准确.相对于动态方法而言,其分析速度快,能适应任务分配空间探索中的反复迭代过程.静态分析方法又可以分为2类:基于标量的方法和基于数组的方法.基于标量的方法[6],一般通过活跃性分析得到各个变量从生产到消费的时间,汇总所有变量的活跃性情况,计算最小存储需求量.对于数组需要转换成标量再处理.基于数组的方法[1~5],主要针对循环内数组进行分析.通过分析数组引用间的相关性,计算最小存储需求量.相对于基于标量的方法而言,基于数组的方法,速度较快,较好的改善了数据空间庞大的问题,是分析多媒体程序存储需求量的主流方法.

在基于数组的分析方法[1~5]中,比较典型的有 Balasa 等提出的 MSC 方法(Memory Size Computation)[2,5].该方法使用线性有界格和基本集进行数组建模与分析加速,使用代数方法进行数据流分析与最小存储需求量的计算,最终得到"精确"的最小存储需求量.该方法准确度较高.然而,其分析速度较慢,且分析仅仅针对循环进行.单一的分析粒度不能适应灵活的任务分配粒度,不利于多种任务粒度间的分配空间探索与寻优.

本文针对 MSC 方法的不足进行了改进,并提出一种面向 SoC 任务分配的多粒度、快速存储需求量分析方法.该方法在基本块、循环和函数粒度上,采用标量方法与数组方法结合的策略,分析存储需求量.针对耗时的循环内数组存储需求量分析,引入数组数据域划分技巧,从而在保持较好准确度同时,大幅提高分析速度.

3 存储需求量分析

3.1 基本块粒度的存储需求量分析

3.1.1 数据检测

将程序转换成 SUIF 中间表示[7].基于中间表示,通过数据检测得到各基本块内的数据信息,包括:数据名字和类型、各类型的数量.接着进行数据活跃性分析.

3.1.2 数据活跃性分析

在分析最小存储需求量时,需考虑存储的 in-place 优化问题,即活跃周期不交叉的多个数据,可以重用同一块存储空间.根据数据活跃性问题的定义[8],给出数据活跃性分析问题中的流值、合并操作和流函数,并且将活跃性分析过程映射到了一组流方程.之后,在 Sharlit 框架下求解活跃性问题[8].首先,通过控制流分析得到基本块结构,并对基本块结构进行转换和优化;然后,对基本块进行静态单一赋值处理,并生成稀疏数据流评估图;最后,基于评估图迭代求解流方程,得到基本块中数据的活跃性情况.

3.1.3 存储需求量计算

根据基本块内数据活跃性,计算基本块的存储需求量.计算方法如下:

基本块的最小和最大存储需求量分别为:$\mathrm{MinSize} = \mathrm{cs} + \mathrm{Max}\left(\sum_{i=1}^{n} \alpha_i \times \mathrm{num}(v_i)\right)$、$\mathrm{MaxSize} = \mathrm{cs} + \sum_{j=1}^{m}\left(\sum_{i=1}^{n}(\alpha_i \times \mathrm{num}(v_i))\right)$.其中,cs 表示程序目标代码的大小;$\alpha_i$ 表示第 i 类变量所需的存储空间;$\mathrm{num}(v_k)$ 表示该基本块内同时活跃变量中第 k 类变量的总数目.

3.2 循环粒度的存储需求量分析

3.2.1 数组检测与表示

本文以多面体模型和格[9]为工具进行分析.多面体模型和格可以准确描述数据、数据操作和数据间关系,适于对嵌套循环和多维数组进行建模与分析.

通过解析程序,得到循环内数组的有关信息,包括:数组迭代域、数组引用的数据域、"迭代域-数据域"的映射关系.

数组迭代域 $P_I = \{\boldsymbol{i} \mid \boldsymbol{Ai} \leqslant \boldsymbol{b}\}$.其中,$\boldsymbol{i} = (i_1, i_2, \cdots, i_n)^{\mathrm{T}} \in Z^n$ 表示数组所在循环的迭代子列向量.$(i_1, i_2, \cdots, i_n)$从左至右分别代表最外层循环迭代子直至最内层循环迭代子.$\boldsymbol{A}$ 表示 m 行 n 列的系数矩阵;变量 m 与多面体域的边界约束数量有关.$\boldsymbol{b} = (b_1, b_2, \cdots, b_n)^{\mathrm{T}} \in Z^n$ 为参数列向量.

映射关系 $L = \{\boldsymbol{x} = \boldsymbol{Ti} \mid \boldsymbol{i} \in Z^n\}$.其中,$\boldsymbol{T}$ 为迭代域到数据域的 m 行 n 列映射矩阵.

数组引用的数据域 $P_D = \mathrm{Polyhedron\_Image}(P_I, L)$.

为了表述方便,将数组 A 第 r 次引用的迭代域表示为 P_{IAr},数据域表示为 P_{DAr},映射关系表示为 P_{MAr}.

3.2.2 数据域划分

根据数组元素的数据重用性,将数组引用的数据域划分为多个解耦合多面体(decoupled Polyhedron,dP).dP 指两个不存在数据重用关系的多面体.

设数组 $\boldsymbol{A}$ 的所有引用对应的数据域集合为 P:$\{P_{DA1}, P_{DA2}, \cdots, P_{DAn}\}$.将数据域划分为 dP 即:将 P 转换为 P':$\{P'_{DA1}, P'_{DA2}, \cdots, P'_{DAm}\}$.其中,$m \geqslant n$,且 $\forall P'_{DAs}, P'_{DAt} \in P'$,满足 $P'_{DAs} \cap P'_{DAt} = \phi$.

数组数据域划分有4种方法:(1)当 $P_{DAr} \cap P_{DAr}' = \phi$ 时,$P' = P = \{P_{DAr}, P_{DAr}'\}$;(2)当 $P_{DAr} \cap P_{DAr}' \neq \phi$,且 $\exists P$,满足 $P \subset P_{DAr}$ 且 $P \subset P_{DAr}'$ 时,$P' = \{P_{DAr} - P, P_{DAr}' - P, P\}$;(3)当 $P_{DAr} \cap P_{DAr}' \neq \phi$,且 $P_{DAr} \subseteq P_{DAr}'$ 时,$P' = \{P_{DAr}, P_{DAr}' - P_{DAr}\}$;(4)当 $P_{DAr} \cap P_{DAr}' \neq \phi$,且 $P_{DAr} \supseteq P_{DAr}'$ 时,$P' = \{P_{DAr}', P_{DAr} - P_{DAr}'\}$.

3.2.3 基于相关性的存储需求量计算

完成数组数据域划分之后,根据 LooPo[10]分析得到

的各数组引用间相关性,推导出各个 dP 引用间的相关性,并构建 dP 粒度的数据相关图.

计算循环最小存储需求量时,需遍历 dP 粒度的数据相关图,并试图找到 dP 最优生产顺序,使存储需求量最小.当一个 dP 所相关的 dP 全部生产之后,该 dP 才可以被生产.在一个 dP 生产之后,所需存储空间会增大;在一个 dP 最后一次被消费之后,所需存储空间会减小.如果多个 dP 的生命周期不交叉,则这些 dP 可以共用同一块存储空间.根据数组的生产与消费周期,计算得到循环内数组的最小存储需求量 MinDataSize.

循环的最大存储需求量通过公式

$$\mathrm{MaxSize} = \mathrm{cs} + \sum_{j=1}^{m}\left(\sum_{i=1}^{n}(\alpha_i \times \mathrm{num}(v_i))\right)$$

得到.

3.3　函数粒度的存储需求量分析

函数的存储需求量计算方法如下:

设$[\mathrm{MinSize}_{l0}, \mathrm{MaxSize}_{l0}]$、$[\mathrm{MinSize}_{l1}, \mathrm{MaxSize}_{l1}]$、…、$[\mathrm{MinSize}_{lm}, \mathrm{MaxSize}_{lm}]$为该函数中各循环的存储需求量区间,$[\mathrm{MinSize}_{b0}, \mathrm{MaxSize}_{b0}]$、$[\mathrm{MinSize}_{b1}, \mathrm{MaxSize}_{b1}]$、…、$[\mathrm{MinSize}_{bn}, \mathrm{MaxSize}_{bn}]$为该函数中各基本块的存储需求量区间,则该函数的最小和最大存储需求量分别为:$\mathrm{MinSize}_f = \mathrm{Max}(\mathrm{MinSize}_{l0}, \mathrm{MinSize}_{l1}, \cdots, \mathrm{MinSize}_{lm}, \mathrm{MinSize}_{b0}, \mathrm{MinSize}_{b1}, \cdots, \mathrm{MinSize}_{bn})$、$\mathrm{MaxSize}_f = \mathrm{cs} + \sum_{j=1}^{m}\left(\sum_{i=1}^{n}(\alpha_i \times \mathrm{num}(v_i))\right)$.

4　应用实例与结果分析

4.1　应用实例

应用实例中的 SoC 平台,采用课题组自主研发并投片成功的嵌入式流媒体处理 SoC(Embedded Stream Processing Multi-Processor SoC,ESP-MPSoC)[11].ESP-MPSoC 作为任务分配的目标 SoC.在任务分配的过程中,利用存储需求量信息,寻找高质量的任务分配方案,以改善 SoC 的处理性能.图 1 描述了 ESP-MPSoC 的体系结构.

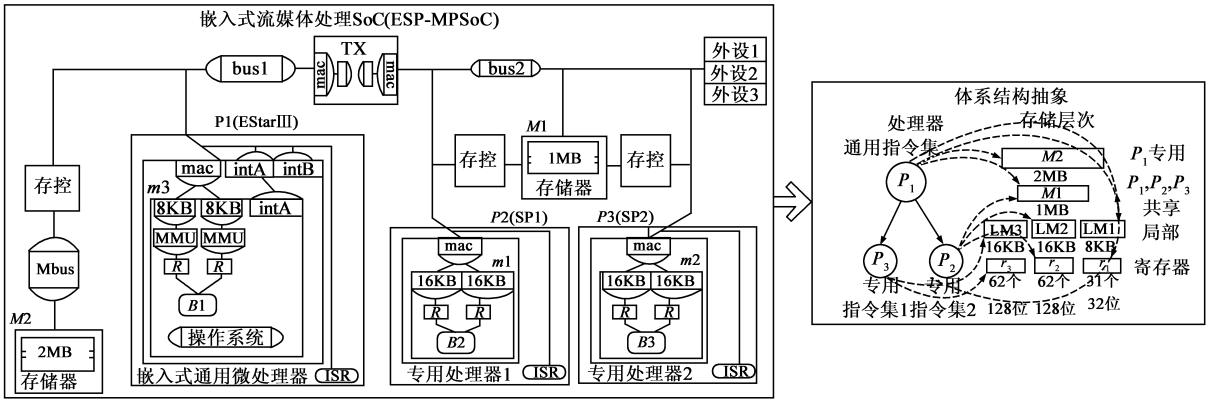


图1　ESP-MPSoC体系结构

4.2　存储需求量分析结果

基本块和循环的存储需求量是分析重点,这里仅给出基本块和循环粒度的分析结果.

4.2.1　基本块粒度的存储需求量

分析基本块的存储需求量时,以 JPEG[12]中的 Huffman 解码、量化和颜色空间转换程序为例.使用 3.1 节方法进行分析.图 2 以 Huffman 解码程序为例,给出分析结果.基本块的命名规则为:"所在函数名:基本块的序号".图 2 中横坐标为基本块名字的字典序升序序号.

SoC 各存储层次(如:寄存器、片上存储器等),能否满足基本块的最小存储需求量,决定了基本块能否正常运行及运行的性能.对于基本块粒度的任务分配而言,需要将基本块的存储需求量作为分配依据之一.

本节的实验结果说明:在基本块粒度的存储需求量分析中,本文方法能够得到程序中各个基本块的最小存储需求量,并找出最小存储需求量较大的基本块及其最小存储需求量,从而为任务分配提供分配依据.

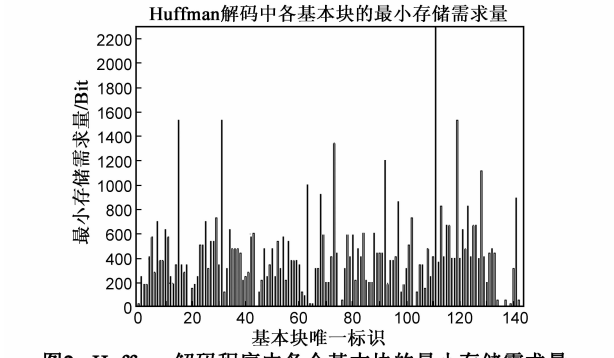


图2　Huffman解码程序中各个基本块的最小存储需求量

4.2.2　循环粒度的存储需求量

在循环粒度的存储需求量分析中,以典型的多媒体处理程序为分析实例,包括:运动检测、奇异值分解更新(Sigular Value Decomposition updating,SVD updating)和 2 维高斯模糊程序.

这些算法的核心均为嵌套循环与数组操作.因此采用 3.2 节方法进行分析.表 1 给出 3 种方法(标量方

法、MSC 方法与本文方法)的分析结果.

表 1 循环的最小存储需求量及分析时间(存储需求量单位:位;时间单位:秒)

算法名字及参数设置		标量方法		MSC 方法		本文方法	
		最小存储需求量	分析时间	最小存储需求量	分析时间	最小存储需求量	分析时间
SVD 更新	$N=25$	34,762	9	75,168	3	67,808	2.0
2 维高斯模糊	$M=50, N=100$	78,848	15	165,728	3	160,864	1.8
运动检测	$m=n=4, M=N=32$	49,440	3	93,248	2	78,336	1.1
	$m=n=4, M=N=64$	157,984	26	310,336	11	279,040	3.2
	$m=n=8, M=N=120$	538,048	1,903	1,070,656	477	954,880	81

下面对表 1 中 3 种分析结果进行比较与分析:(1)标量方法.该方法需要将数组转换成标量.其分析单位是单个数组元素,分析结果可以准确到单个数组元素,准确度最高.但是,该方法会导致所分析的数据空间庞大,分析速度缓慢.(2)MSC 方法.该方法采用基于基本集的分析方法.将所有的数组元素划分成多个基本集.在相关性分析与最小存储需求量计算的过程中,将处理单位由单个数组元素提高为基本集.在损失较小准确度的情况下,较大的提高了分析的速度.(3)本文方法.本文方法采用基于多面体的分析方法.将数组数据域划分为多个 dP,并以 dP 为单位,进行相关性与存储需求量分析.相对于标量方法,分析单位由单个数组元素提高为多面体,用较小的分析准确度损失换取了大幅度的速度提高.相对于 MSC 方法,使用了基于多面体的相关性分析与最小存储量计算.多面体模型理论与实现特点,决定了基于多面体分析的高效性.

本节实验结果说明:分析循环存储需求量时,本文方法相对于标量方法,能够以较小的准确度损失,换取较大的速度提高;相对于 MSC 方法,能够在保持较高准确度的同时,获取较大速度提高.

4.3 任务分配

为了验证存储需求量对于任务分配的有效性,分别在考虑与不考虑存储需求量的情况下,使用进化蚁群优化算法[13]进行任务分配.通过比较分配结果,说明存储需求量对于任务分配的重要性.

4.3.1 任务分配对象及属性

任务分配对象采用:Huffman 解码子程序、量化子程序和运动检测程序.为了更好的进行任务分配,使用 PluTo[14]将运动检测程序的核心循环转换为 16 个可并行的循环子块.将运动检测程序的 16 个循环子块、Huffman 解码子程序的 10 个基本块和量化子程序的 10 个基本块,共 36 个任务作为任务分配对象.这些分配对象的存储需求量,分别使用 3.1 节和 3.2 节方法得到.这些分配对象在 EStarIII 和 SP 上的计算时间,通过 ModelSim 上的寄存器传输级模拟得到,见表 2.

表 2 任务分配对象的计算时间与存储需求量(存储需求量单位:位;时间单位:纳秒)

任务分配对象	运动检测的单个循环子块	Huffman 解码的单个基本块	量化的单个基本块
EStarIII 计算时间	6,469	6,465	15,885
SP1 计算时间	1,420	560	2,960
SP2 计算时间	1,420	560	2,960
最小存储需求量	3,090	1,632	864
最大存储需求量	5,096	3,072	1,664

4.3.2 任务分配算法及配置

采用 2 种任务分配方案:(1)不考虑分配对象的存储需求量,仅考虑待计算时间;(2)同时考虑分配对象的计算时间和存储需求量.通过比较分配结果,说明存储需求量对于任务分配的作用.

进化蚁群算法的目标函数设置为:

$$OF=\max\{\sum_{i=1}^{k1}\text{time}(N_i^1),\ \sum_{i=1}^{k2}\text{time}(N_i^2),\ \sum_{i=1}^{k3}\text{time}(N_i^3)\}$$

其它参数设置参照文献[13].

不考虑存储需求量时,对于第 j 个处理器,$\text{time}(N_i^j)=C(N_i^j)$.

考虑存储需求量时:$\text{time}(N_i^j)=C(N_i^j)+D(N_i^j)$.其中,$C(N_i^j)$为计算时间,通过 ModelSim 模拟得到.$D(N_i^j)$为访存时间,根据存储需求量计算得到,见下面公式.$k_j$ 为已分配到处理器 j 上的任务数目,$1\leqslant j\leqslant 3$.MinSize 和 MaxSize 分别为程序最小和最大存储需求量.MemSize 为存储器大小.BandWidth 为访存带宽.T_m 为额外的访存开销.$\alpha\in(0,1]$为额外访存开销的调节系数.

$$D(N_i^j)=\begin{cases}\left(\dfrac{\text{MaxSize}}{\text{BandWidth}}\right)*\text{ClockCycle}, & \text{if MemSize}\geqslant\text{MaxSize}\\ \left(\dfrac{\text{MaxSize}}{\text{BandWidth}}\right)*\text{ClockCycle}+\alpha\times T_m, & \text{if MinSize}\leqslant\text{MemSize}<\text{MaxSize}\\ +\infty, & \text{if MemSize}<\text{MinSize}\end{cases}$$

4.3.3 任务分配结果与性能分析

表 3 中给出了 2 种任务分配结果及其对应的 SoC 性能.相对于不考虑存储需求信息的任务分配,考虑存储需求信息的任务分配获得了 12.97% 的性能提高.

通过向任务分配引入存储需求量,获得了性能提高.其原因为:多媒体程序的数据存储与传输效率极大影响着 SoC 性能.通过向任务分配引入存储需求信息,有助于寻找高效的任务分配方案,从而提高数据传输与存储效率.如果任务分配忽视了存储需求信息,则可能导致:任务分配结果虽然计算性能高,但数据存储与传输性能低,降低了 SoC 系统性能.尤其是对于计算简单,但数据存储与传输频繁的任务,不考虑存储需求信息的任务分配,其分配结果的质量会受到更大的影响.

表 3　任务分配结果及性能比较

基本情况	任务分配结果	时间
不考虑存储需求量	EStarIII:4 个运动检测循环子块.	52,843ns
	SP1:6 个运动检测循环子块; 5 个 Huffman 基本块;5 个量化基本块.	
	SP2:6 个运动检测循环子块; 5 个 Huffman 基本块;5 个量化基本块.	
考虑存储需求量	EStarIII:6 个 Huffman 基本块.	45,990 ns
	SP1:8 个运动检测循环子块; 2 个 Huffman 基本块;5 个量化基本块.	
	SP2:8 个运动检测循环子块; 2 个 Huffman 基本块;5 个量化基本块.	

5　结论

面向 SoC 任务分配,提出了一种针对多媒体程序的多粒度、快速存储需求量分析方法.该方法基于数据活跃性和多面体相关性,分别在基本块、循环和函数粒度上进行存储需求量分析.分析中将标量与数组区别对待,并将数组数据域划分为解耦合多面体.以多面体取代单个数组元素,作为相关性与存储需求量分析的单位,有效缓解了数据空间庞大的问题,大幅度减少了分析时间.所得到的存储需求信息,有助于提高任务分配质量并改善 SoC 系统性能.

参考文献:

[1] Zhu H. Computation of the Minimum Data Storage for Multi-dimensional Signal Processing Systems[D]. Chicago: University of Illinois, 2007.

[2] Balasa F. Background Memory Allocation for Multi-dimensional Signal Processing[D]. Leuven, Belgium: Katholieke Universiteit, 1995.

[3] IMEC Research Center. Introduction to the ATOMIUM Tool Suite for Memory Centric Code Optimization [EB/OL]. http://www.imec.be/design/atomium, 2000 - 01 - 01/2007 - 01 - 01.

[4] Hu Q. Hierarchical Memory Size Estimation for Loop Transformation and Data Memory Platform Optimization[D]. Trondheim, Norway: Norwegian University of Science and Technology, 2007.

[5] Balasa F, et al. Computation of storage requirements for multi-dimensional signal processing applications[J]. IEEE Trans on Very Large Scale Integration Systems, 2007, 15(4): 447 - 460.

[6] Gajski DD, et al. Specification and design of embedded hardware/software systems[J]. IEEE Design & Test of Computers, 1995, 12(1): 53 - 67.

[7] Stanford University. Homepage of Stanford University Intermediate Format Group[EB/OL]. http://suif.stanford.edu, 2005 - 09 - 01/2009 - 10 - 25.

[8] Tjiang S W. Automatic Generation of Data-flow Analyzers: a Tool for Building Optimizers[D]. Stanford: Stanford University, 1993.

[9] Wilde D K. A library for doing polyhedral operations[J]. International Journal of Parallel, Emergent and Distributed Systems, 2000, 15(3): 137 - 166.

[10] Lengauer C. Loop parallelization in the polytope model[A]. Proceedings of the 4th International Conference on Concurrency Theory[C]. Hildesheim, Germany: Springer-Verlag Press, 1993. 398 - 416.

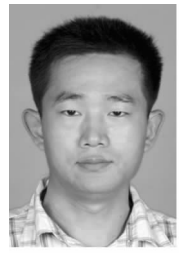
[11] Yan M, et al. A heterogeneous multicore SoC optimized for embedded visual media process[A]. Proceedings of WRI International Conference on Communications and Mobile Computing[C]. Kunming: IEEE Computer Society Press, 2009. 12 - 16.

[12] Lee C, et al. MediaBench: a tool for evaluating and synthesizing multimedia and communications systems[A]. Proceedings of the 30th Annual International Symposium on Microarchitecture[C]. Research Triangle Park, North Carolina: IEEE Computer Society Press, 1997. 330 - 335.

[13] Wang D, et al. Collaborative hardware/software partition of coarse-grained reconfigurable system using evolutionary ant colony optimization[A]. Proceedings of Asia and South Pacific Design Automation Conference[C]. Seoul: IEEE Computer Society Press, 2008. 679 - 684.

[14] Bondhugula UKR. Effective Automatic Parallelization and Locality Optimization Using the Polyhedral Model[D]. Columbus, Ohio: the Ohio State University, 2008

作者简介:



赵　鹏　男,1979 年 9 月出生于河南鹤壁.现为国防科技大学计算机学院博士研究生.主要研究方向为嵌入式系统与 SoC 设计方法,多媒体程序自动分析与性能优化,虚拟现实与计算机仿真.　E-mail: pengzhao@nudt.edu.cn

王大伟　男,1980 年 2 月出生于辽宁东港.现为国防科技大学计算机学院博士研究生,主要研究方向为电子设计自动化与 SoC 设计方法学.　E-mail: daweiwang@nudt.edu.cn.

李思昆　男,1941 年 4 月出生于山东青岛,国防科技大学计算机学院教授,博士生导师,主要研究方向为嵌入式系统与 SoC 设计方法,虚拟现实与可视化.　E-mail: lisikun@263.net.cn.