第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

·开发研究与设计技术· 文章编号：1000—3428(2012)03—0276—04 文献标识码：A 中图分类号：TP301.6

面向可重构计算系统的模块映射算法

刘 杰，吴 强，赵全伟

(湖南大学信息科学与工程学院，长沙 410082)

摘 要：为消除重构时间对可重构计算系统性能的影响，针对多重构模块，提出一种基于动态部分可重构技术的顺序型应用程序模块映射算法。利用动态可重构技术的高效性和灵活性，通过隐藏重构时间，达到减少程序执行时间和提高系统性能的目的。基于 JPEG 编码测试实例的实验结果表明，运用该算法实现的模块映射方案其程序执行速度是软件实现方式的 3.31 倍，是硬件方式的 2.59 倍。

关键词：可重构计算；模块映射算法；动态部分可重构；重构时间；现场可编程门阵列

Module Mapping Algorithm for Reconfigurable Computing System

LIU Jie, WU Qiang, ZHAO Quan-wei

(College of Information Science and Engineering, Hunan University, Changsha 410082, China)

【Abstract】 In order to reduce the impact of the configuration time to the performance of reconfigurable computing, this paper proposes a module mapping algorithm based on dynamic partial reconfigurable technology for sequential applications. It deals with the mapping of multi-modules. It utilizes the high effectiveness and flexibility of dynamic reconfiguration to hide the configuration time, so that the program execution time can be reduced and the system performance is improved. Experiment based on JPEG encoding example shows that the algorithm achieves 3.31 and 2.59 speedups compared to implementation methods of pure software and pure hardware respectively.

【Key words】 reconfigurable computing; module mapping algorithm; dynamic partial reconfigurable; configuration time; Field Programmable Gate Array(FPGA)

DOI: 10.3969/j.issn.1000-3428.2012.03.091

1 概述

可重构计算的思想最早是由加利福尼亚大学的 Geraid Estrin 教授提出，并研制了原型系统，这为可重构计算系统的研究奠定了基础[1]。可重构计算是一种性能介于微处理器与专用集成电路之间的新型时空域计算模式，既保留了硬件的高性能，又兼具了软件的灵活性，在高性能计算领域具有广阔的应用前景。基于 SRAM 的现场可编程门阵列(Field Programmable Gate Array, FPGA)具有快速重新配置资源的特性，这一特性刚好满足可重构计算结构的要求，从而为可重构计算提供了一个可实现的硬件平台。

对可重构计算概念的理解因人而异，本文将可重构计算理解为可重构器件 FPGA 配置为执行某种专门的任务，随后可以按照系统的要求重新配置以执行其他的任务。目前，可重构计算作为一种全新的计算模式虽然有了较大的发展，但仍有不少问题有待研究。动态可重构计算系统存在一个缺点，即重构时间较长。在重新配置任务时，FPGA 端口对外呈高阻状态，配置完成后，才能实现新的逻辑功能，这个由配置所引起的延时就是重构时间。重构时间会影响系统功能的连续性，是限制可重构计算系统性能的一个重要因素。因此，如何减少重构时间，提高动态可重构计算系统的性能，成为一个新的研究热点。

可重构计算的出现使传统意义上的硬件和软件的界限变得模糊，其本质是利用可重构器件的可编程特性，在运行时根据需要动态改变可重构器件的逻辑功能，从而使系统兼具灵活、高效、硬件资源可复用等特点[2]。自 20 世纪 90 年代开始，可重构计算技术的发展比较迅速，国内外也掀起了对其的研究热潮并取得了不少成果。美国加州大学提出的 Grap 方案[3]和麻省理工学院提出的 MATRIX 方案[4]在可重构计算的体系结构方面取得重要的成果。在减少配置延时方面，文献[5]通过改进硬件平台来解决重构时间长的问题；文献[6]通过压缩算法减小配置文件的大小，从而达到减少配置延时的目的；文献[7]通过带重定位和碎片收集的配置预取方法来减小配置延时对系统性能的影响；文献[8-9]则是通过预配置来消除重构时间对系统性能的影响，从而加速硬件任务运算。本文从透明编程模型出发，对基于动态部分可重构的计算系统进行研究，提出针对多重构模块的模块映射算法，以提高程序的执行效率和系统的性能。

2 模块映射

可重构计算系统多采用微处理器和可重构器件相结合的结构，其中，可重构器件主要用于执行计算密集型的硬件任务，而微处理器负责执行软件任务以及对硬件任务进行配置和控制。一方面，随着 FPGA 和集成技术的发展，已可以在单块 FPGA 芯片上实现一个完整的可重构计算系统；另一方面，基于 SARM 的 FPGA 支持动态部分可重构，因此，本文以内嵌了 PowerPC405 处理器的 Xilinx Virtex-II Pro 为实验开发平台。

动态部分可重构允许系统在运行过程中对 FPGA 中的部分逻辑进行重配置，而未经配置的部分可以继续正常运行。与全局可重构相比，部分可重构配置文件更小，并具有更高

基金项目：国家“863”计划基金资助项目(2007AA01Z104)；湖南省自然科学基金资助项目(07JJ6136)；中央高校基本科研业务费基金资助项目

作者简介：刘 杰(1985－)，男，硕士研究生，主研方向：可重构计算；吴 强，副教授、博士；赵全伟，硕士研究生

收稿日期：2011-07-30 **E-mail：**liujie0620@gmail.com

的灵活性；另外，动态部分可重构支持多重构模块，可以同时运行多个任务。因此，本文将对多重构模块的动态部分可重构系统进行研究。

可重构计算利用动态部分可重构的特性可以在更少更小的 FPGA 器件上实现大规模、复杂的计算系统。这是因为利用动态部分可重构技术，可以分时复用重构模块的逻辑资源，虽然单个重构模块在某时刻只能实现一个逻辑功能 fun_i $(i=0,1,\cdots,N)$，但是通过分时复用则可以在单个重构模块上顺序实现更多的功能，实现的功能为单个功能之并集，如式(1)所示：

$$fun = fun_0 \cup fun_1 \cup \cdots \cup fun_N \tag{1}$$

在动态可重构计算系统中，对应用程序中计算密集型的任务用软件执行，速度比较慢，会影响系统的性能和程序的执行速度，因此，本文利用硬件代替那些复杂的计算任务，这样就可以加速程序运行以提高系统的效率。但是，可重构系统中 FPGA 的逻辑资源有限，只能选择部分任务用硬件来实现；另外，硬件任务的重构时间也是制约程序加速的一个重要因素，所以，需要解决以下 3 个问题：(1)如何确定应用程序中的任务是划分为硬件还是软件；(2)如何确定硬件任务是映射到哪个重构模块中；(3)如何消除硬件任务的重构时间对程序加速的影响。

为了提高应用程序的执行速度，本文结合动态部分可重构的特性，提出基于多重构模块的模块映射算法，通过此算法来解决系统所面临的问题。与已有的研究成果不同，本文主要利用动态部分可重构的特性对程序进行动态软硬件划分，并将划分为硬件的任务映射到重构模块上，任务的重构时间会影响任务的划分结果；而文献[8-9]是对已经划分为软件或者硬件任务的调度问题进行研究。

3 动态可重构计算系统的编程模型

透明编程模型[10]是为了让设计人员使用任务函数库开发出高效率的软硬件混合系统而提出的。库中的每个任务(函数)具有软件和硬件 2 种实现方式，它们屏蔽了具体的软硬件实现细节，但具有相同函数封装形式，以便在运行时可动态链接到不同的实现上。本文动态可重构计算系统的编程模型如图 1 所示。运行在系统上的应用程序首先经过模块映射算法，将程序中的任务划分为软件或硬件，并指定硬件任务所映射的重构模块；然后利用控制器将硬件任务配置到 FPGA 中相应的重构模块中，而软件任务映射到 FPGA 的 CPU 中。

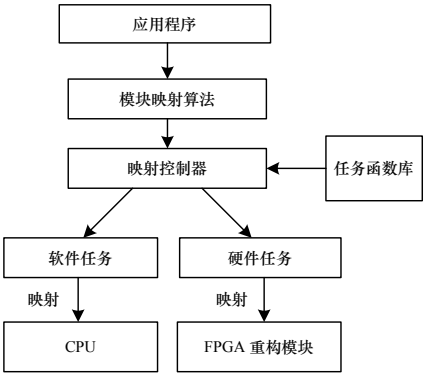

图 1 可重构计算系统的编程模型

4 模块映射策略

4.1 系统体系结构

可重构计算系统是由微处理器和可重构器件组成的混合系统，本文的计算系统采用动态部分可重构技术，并将整个系统集成在 FPGA 器件上，其体系结构如图 2 所示。整个系统将 FPGA 分成 2 个主要的区域：静态区域和动态区域(又称重构区域)。动态区域由多个重构模块组成，重构模块主要用于实现硬件任务的功能，其功能随任务的改变而变化，系统按需求动态地改变重构模块上的任务。重构模块是系统的核心部分，映射在其上的任务分时复用其资源。静态区域主要包含处理器、存储器及相关的外围器件等，静态区域的功能在系统运行的过程中保持不变，其中，处理器 PowerPC405 主要负责软件任务的执行和对重构模块上任务配置的控制；存储器(DDR 和 BRAM)为系统的运行提供内存空间；外围器件(RS232 串口、System_ACE 以及 ICAP)主要用于保存硬件任务的配置文件、提供配置信息写入重构模块的通道、程序调试和结果显示等。

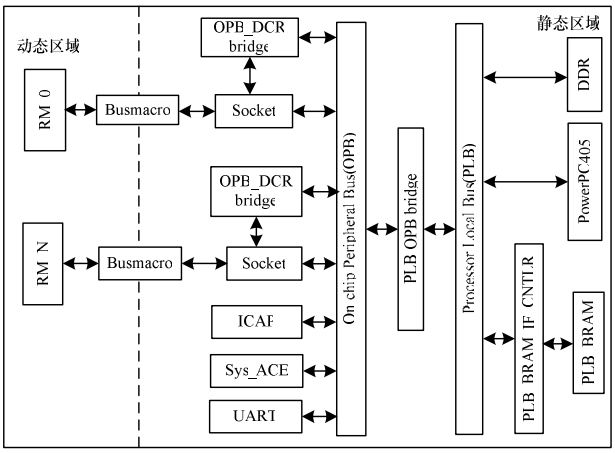

图 2 动态可重构计算系统的体系结构

4.2 模块映射算法模型

在可重构计算系统中，应用程序用软件去执行那些计算复杂的任务会比较慢，从而影响程序的执行效率。实例程序中的任务都为软件执行的情况如图 3 所示，其中，t 表示单位时间。为了加快程序的执行速度，用硬件来代替软件；同时，为了消除硬件的重构时间的影响，将硬件任务分成配置和计算 2 个部分。程序的硬件执行情况如图 4 所示，其中，s 为单位逻辑资源。可以看到，由于硬件重构时间的影响，程序没有起到很好的加速效果。

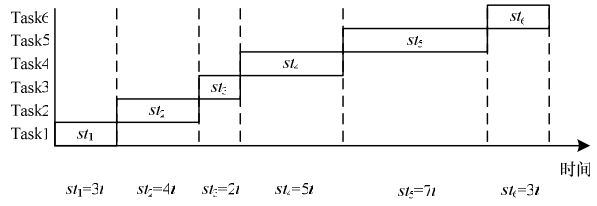

图 3 软件执行结果

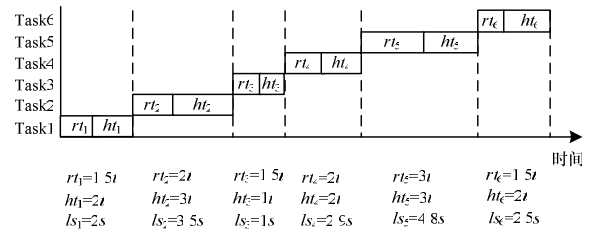

图 4 硬件执行结果

在模块映射算法中，利用动态部分可重构的特性，动态地改变重构模块上的硬件任务，从而使更多的软件任务能够被硬件所代替；对于硬件配置所引起的重构时间，可将其隐

藏在其他任务的计算过程中。能够隐藏重构时间的原因是配置 FPGA 某个重构模块时，其他的重构模块可以保持正常运行(计算)，通过使配置和计算在时间上重叠，在效果上达到隐藏重构时间的目的。

在本文的可重构计算系统中，应用程序是用 C 语言编写，程序中的任务是顺序执行且不存在嵌套的情况。假设某个应用程序有 n 个任务，动态区域中重构模块的数目为 m (即某时刻动态区域上最多有 m 个硬件任务)，重构模块所占用的逻辑资源数为 $l_j(j=1,2,\cdots,m)$。程序中的每个任务含有软件和硬件 2 种执行方式。每个任务包含的参数：软件的参数为软件运行时间 st_i；硬件的参数有硬件运行时间 ht_i、重构时间 rt_i 和任务所占的逻辑资源数 $ls_i(i=1,2,\cdots,n)$。本文以应用程序运行时间作为模块映射算法的评价准则。设程序运行时间为 T，任务 i 的运行时间为 $T_i(i=1,2,\cdots,n)$，第 i 个任务在系统中运行的时间为：

$$T_i = p_i \cdot st_i + (1-p_i)\cdot(rt_i + ht_i) \tag{2}$$

其中，参数 $p_i=\{0,1\}$，$p_i=0$ 表示任务划分为硬件执行；$p_i=1$ 表示任务划分为软件执行。模块映射算法欲将硬件任务的重构时间隐藏在其他任务(包括硬件和软件)的计算过程中，因此，引入参数 q_i 表示任务能否隐藏，式(2)变形为：

$$T_i = p_i \cdot st_i + (1-p_i)\cdot(q_i \cdot rt_i + ht_i) \tag{3}$$

其中，$q_i=\{0,1\}$，$q_i=0$ 表示任务可以隐藏，$q_i=1$ 表示任务不可以隐藏。所以，程序运行时间为：

$$T=\sum_{i=1}^{n}T_i=\sum_{i=1}^{n}\left[p_i \cdot st_i + (1-p_i)\cdot(q_i \cdot rt_i + ht_i)\right] \tag{4}$$

由式(4)可知，程序运行时间 T 与参数 p_i 和 q_i 相关。

影响任务映射的因素包括：(1)重构模块数目 m，其决定了某时刻系统中最大的硬件任务数；(2)每个重构模块所分配的逻辑资源数 $l_j(j=1,2,\cdots,m)$，其决定了哪些任务能够映射到该模块中；(3)配置硬件任务所引起的重构时间 rt_i。

下面根据影响任务映射的因素，分 3 步求得最小程序运行时间 T 和确定程序中任务的划分方式。

步骤 1 通过动态区域的重构模块数目 m 和每个重构模块的逻辑资源 $l_j(j=1,2,\cdots,m)$ 对应用程序中的任务进行分类。将映射到相同重构模块的任务归为一类，分类的原则是更有效地利用重构模块的资源，即将所占资源 ls_i 小于或约等于某个重构模块资源的硬件任务初步映射到此重构模块中。程序分类的结果如图 5 所示，其中假设有 2 个重构模块，且 l_1=3s，l_2=5s。

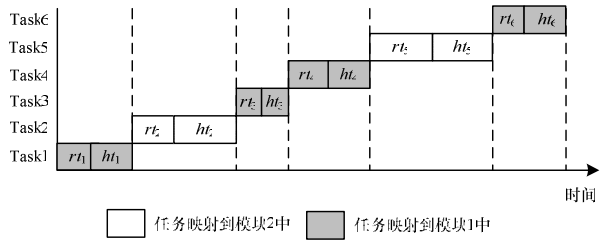

图 5 任务的初步映射

具体实现步骤如下：

(1)根据任务 i 需要的逻辑资源数 ls_i，找到满足此关系 $ls_i<l_j(j=1,2,\cdots,m)$ 的所有重构模块。

(2)找到资源数与 ls_i 最相近的重构模块。

(3)建立一个数组 $a[n]$ 用于标记程序中 n 个任务初步映射到的重构模块，以便分类。例如，任务 i 与重构模块 k 满足(1)和(2)的要求，则令 $a[i-1]=k$，表示任务 i 分配在重构模块 k 中，那么数组 $a[n]$ 中所有等于 k 的任务视为同一类。

(4)重复(1)~(3)，确定程序中所有任务初步映射到的重构模块。

步骤 2 根据任务的分类情况和重构时间的大小确定参数 q_i 的值。为了将硬件任务的重构时间隐藏在其他任务的计算过程中，应遵循以下原则：某个重构模块在执行任务时不能被配置，否则会中断计算，换而言之，相邻 2 个硬件任务不能映射在同一重构模块中；连续 2 个或者多个硬件任务要保证配置的顺利执行，需要满足后面任务的重构时间不大于前面任务的计算时间。具体实现步骤如下：

(1)程序中所有的任务初始化为硬件；查找 $a[n]$，找出相同类的所有任务，即初步映射在相同重构模块的任务。

(2)应用程序中，如果连续 2 个或者多个任务是同一类，依次设定任务的实现方式，换而言之，假设程序中 n 个连续任务是同一类，依次设定 $n!$ 种不同的实现方式。例如，如果程序中连续 2 个任务是同一类，则实现方式共有 2 种：前一个任务为硬件，后一个任务为软件；前一个任务为软件，后一个任务为硬件。

(3)按照程序中任务的执行顺序，判定为硬件实现、相邻且不是同类的任务是否满足约束 $ht_{i-1}\geqslant rt_i$，如果满足，任务 i 的重构时间可以隐藏，即 $q_i=0$；如果不满足，任务 i 的重构时间不可隐藏，即 $q_i=1$。

(4)判定为硬件、不相邻且不是同类的任务是否满足此约束 $ht_j+st_{j+1}+\cdots+st_{i-1}\geqslant rt_i(j<i-1)$，如果满足，任务 i 的重构时间可以隐藏，即 $q_i=0$；如果不满足，任务 i 重构时间不可隐藏，即 $q_i=1$。

步骤 3 确定所有任务的 p_i 的值，求得最小程序运行时间 T。在步骤 2 的(2)中已经确定同类且相邻任务的划分方式，但是它们有多种组合情况，所以，每种组合都会求得一个程序运行时间 T，选择其中最小的 T，即得到了运行时间最小的组合；对于步骤 2 没有确定划分方式的任务，任务的参数 p_i 是根据参数 q_i 来确定的，即：如果任务 i 的重构时间可以隐藏，即 $q_i=0$，那么任务 i 为硬件实现，即 $p_i=0$；如果重构时间不能隐藏，即 $q_i=1$，且满足约束 $rt_i+ht_i<st_i$，那么任务 i 为硬件实现即 $p_i=0$；否则，任务 i 为软件实现，即 $p_i=1$。如图 6 和图 7 所示，Task3 和 Task4 是同类且相邻的 2 个任务，它们的组合产生了 2 个程序运行时间 T，最小的 T 为图 6 的组合情况，值得说明的是，图 6 和图 7 中硬件任务只要计算前已经配置好即可，并不要求精确对齐到计算开始时刻。可以看出，模块映射算法针对多重构模块，消除了重构时间对系统的影响，确定了程序中任务的划分方式和程序的最小运行时间。模块映射算法流程如图 8 所示。

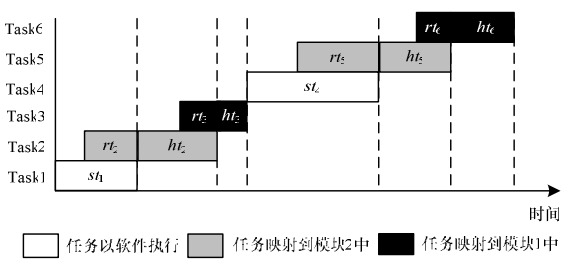

图 6 Task3 为硬件、Task4 为软件的实现方式

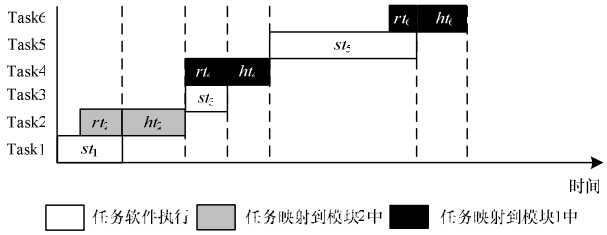

图 7　Task3 为软件、Task4 为硬件的实现方式

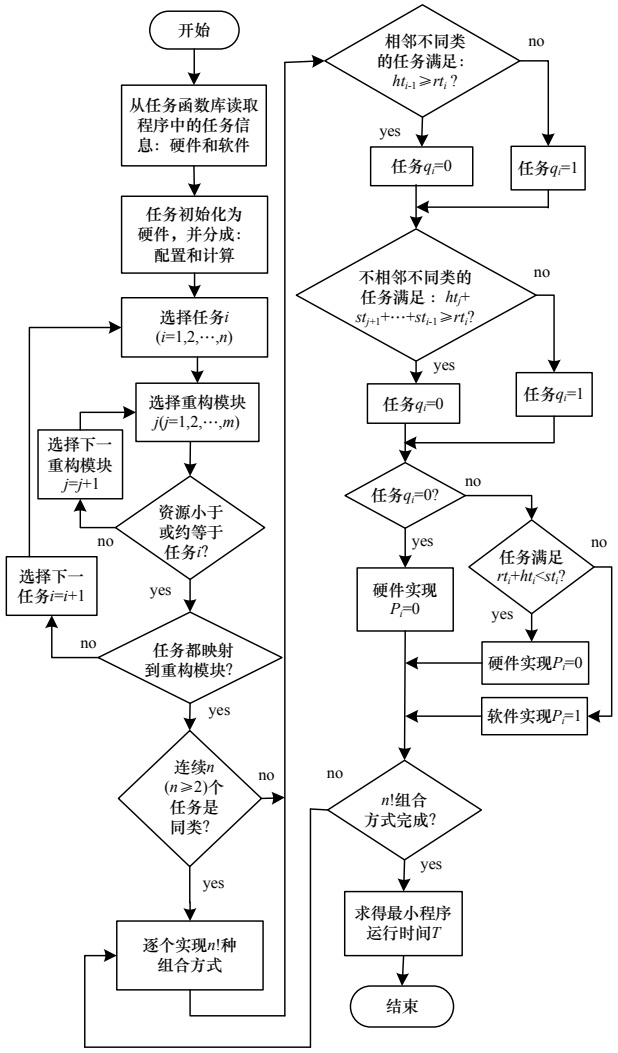

图 8　模块映射算法流程

5　实验结果与分析

实验平台是 Virtex-II Pro XC2VP30 FPGA 开发板，处理器为 FPGA 中内嵌的 PowerPC405，系统的工作时钟频率为 100 MHz。实验是在透明编程模型下的动态部分可重构系统上进行的，以 JPEG 编码，通过模块映射算法调用库中的软件或硬件任务以实现一个运行高效的应用程序。JPEG 编码主要由颜色空间转换(RGB2YCbCr)、离散余弦变换(Discrete Cosine Transform, DCT)、量化(Quantization)和哈夫曼编码(Huffman Encoding)等几部分组成，由于色度 Cb 和 Cr 的处理方式相同，因此它们使用的任务是相同的。JPEG 编码实验所用到库中的任务及其资源(Slice)占用情况为：RGB2YC=153，CbCr_Dct=2 767，CbCr_Qua=1 620，CbCr_Huff=1 668，Y_Dct=2 955，Y_Qua=1 620，Y_ Huff=1 595。动态部分可重构系统上设计了 2 个重构模块，其布局布线如图 9 所示，重构模块上的任务随系统的需要而被动态替换。

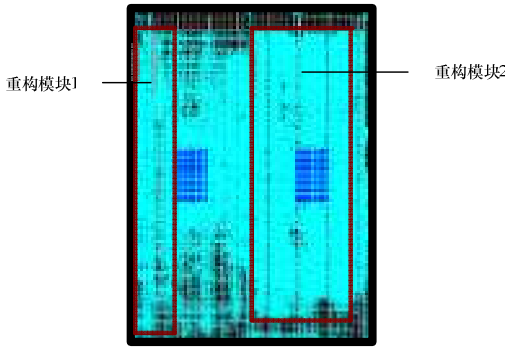

图 9　任务映射到重构模块的布局布线

根据 JPEG 编码中任务的参数，模块映射算法对程序加速的情况如图 10 所示。实验设计了 3 种方式来显示模块映射算法的加速效果，分别为：(1)JPEG 编码用软件实现；(2)JPEG 编码中的所有任务用硬件实现(考虑重构时间的影响)；(3)JPEG 编码引入模块映射算法后，由通过隐藏硬件的重构时间而组成的软硬混合系统实现。从实验结果可知，由于硬件任务重构时间的影响，硬件代替软件的方式没有起到很好的加速效果，并且硬件的高效性也没有得以体现，从而影响了系统性能；JPEG 编码引入模块映射算法后，程序的执行速度是软件方式的 3.31 倍，是硬件方式的 2.59 倍。因此，模块映射算法能加速应用程序的执行，改善系统的性能。

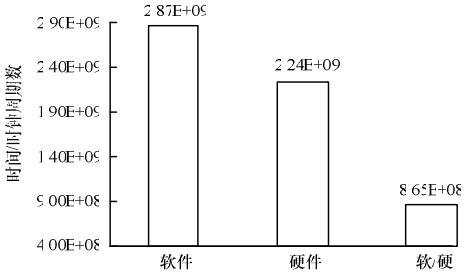

图 10　JPEG 编码 3 种不同实现方式的性能比较

6　结束语

重构时间是制约可重构计算性能的一个重要因素。本文提出的模块映射算法将硬件任务分为配置和计算两部分，利用动态部分可重构的特性，将任务的重构时间隐藏在其他任务的计算过程中，并确定任务映射的模块，从而克服硬件任务的重构时间，达到提高程序执行速度和系统性能的目的。实验结果证明了整体设计方案的可行性。如何改进该算法以使其应用到带有嵌套的应用程序中是下一步研究的重点。

参考文献

[1] Estrin G, Bussel B, Turn R, et al. Parallel Processing in a Restructurable Computer System[J]. IEEE Trans. on Electronic Computers, 1963, EC-12(6): 747-755.

[2] 王志远，王建华，徐　旸. 可重构技术综述[J]. 小型微型计算机系统, 2009, 30(6): 1203-1207.

[3] Hauser J R, Wawrzynek J. Grap: A MIPS Processor with a Reconfigurable Coprocessor[C]//Proc. of IEEE Symposium on FPGAs for Custom Computing Machines. [S. l.]: IEEE Press, 1997.

[4] Mirsky E, Dehon A. MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources[C]//Proc. of IEEE Symposium on FPGAs for Custom Computing Machines. [S. l.]: IEEE Press, 1996.

(下转第 283 页)