ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, Supplement, December 2009, pp.15−22

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

CPU/FPGA混合架构上的硬件线程加速方法*

陈天洲+, 严力科, 胡 威, 马吉军

(浙江大学 计算机科学与技术学院,浙江 杭州 310027)

Hardware Thread Accelerating Method Based on CPU/FPGA Hybrid Architecture

CHEN Tian-Zhou+, YAN Li-Ke, HU Wei, MA Ji-Jun

(College of Computer Science, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: E-mail: tzchen@zju.edu.cn

Chen TZ, Yan LK, Hu W, Ma JJ. Hardware thread accelerating method based on CPU/FPGA hybrid architecture. *Journal of Software*, 2009,20(Suppl.):15−22. http://www.jos.org.cn/1000-9825/09003.htm

Abstract: The CPU/FPGA hybrid architecture is a popular reconfigurable computing architecture. In order to ease the use of FPGA, a hardware thread approach is proposed, and a hardware thread executing mechanism is designed to make use of the reconfigurable resources. Software thread and hardware thread can be executed in parallel while computation-intensive tasks are assigned to hardware threads and control-intensive tasks are assigned to software threads. Simics simulator is adopted to simulate a hybrid architecture platform, on which software and hardware multithreading DES, MD5SUM and MergeSort algorithms are evaluated. The results show that the average speedup is 2.30, and it proves that the approach explored the performance of CPU/FPGA hybrid architecture efficiently.

Key words: hardware acceleration; hardware thread; multithreading; CPU/FPGA hybrid architecture; reconfigurable computing

摘 要: CPU/FPGA 混合架构是可重构计算的普遍结构,为了简化混合架构上 FPGA 的使用,提出了一种硬件线程方法,并设计了硬件线程的执行机制,以硬件线程的方式使用可重构资源.同时,软硬件线程可以通过共享数据存储方式进行多线程并行执行,将程序中计算密集部分以 FPGA 上的硬件线程方式执行,而控制密集部分则以 CPU 上的软件线程方式执行.在 Simics 仿真软件模拟的混合架构平台上,对 DES,MD5SUM 和归并排序算法进行软硬件多线程改造后的实验结果表明,平均执行加速比达到了 2.30,有效地发挥了 CPU/FPGA 混合架构的计算性能.

关键词: 硬件加速;硬件线程;多线程;CPU/FPGA 混合架构;可重构计算

近年来,随着微电子技术的快速发展,微处理器向着提高架构执行效率、多核心设计、灵活弹性的扩展和深层次功能整合方向发展,并取得了很大进步.尤其是近几年CPU/FPGA可重构混合体系架构的发展,使得微处理器领域达到了一个崭新的发展阶段.可重构计算迎合了未来计算领域的需求,能够提供更高层次的并行运算能力,由于使用了硬件方式执行,那些并行性需求较高的应用程序的计算性能有了很大提升,非常适用于高速数字信号处理[1]、多媒体处理[2,3]以及生物信息处理[4]等.然而,由于针对可重构计算平台的操作系统支持还很不

* Supported by the National Natural Science Foundation of China under Grant No.60673149 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z105 (国家高技术研究发展计划(863))

Received 2008-07-01; Accepted 2009-04-02

完善,要求开发者和用户具备硬件设计的经验,并且开发周期较长、成本过高,因此可重构计算至今没有得到很广泛的应用.

为了简单高效地管理和使用计算机中的可重构资源,一些已有研究分别从软件、硬件和软硬件协同的角度展开.Hayden Kwok-Hay So 提出了针对基于FPGA可重配置计算机设计的操作系统BORPH[5],将可重构资源像CPU运算时间和内存一样封装在操作系统服务中,开发者只需关注高层程序设计,从而简化了可重构资源的使用.Greg Stitt和Frank Vahid 则设计了Warp处理器[6],从硬件设计方面简化开发者使用可重配置资源的难度,该处理器能够动态地将一个程序的耗时区域自动转为可重构硬件实现的电路来执行,从而减少执行时间和能源消耗.Erik Anderson等人通过设计一个叫做HWTI的硬件线程接口实现混合架构软硬件多线程[7].

BORPH 在操作系统上对可重配置提供支持,而 Warp 处理器在硬件上对可重配置提供支持,这两种方案在利用可重配置资源上都能够达到较好的效率,程序的执行速度有很大提升,能耗也有减少.但是其设计涉及到太多的方面,BORPH 操作系统的设计上对 Linux 有很大的改动,而所实现的功能却相当简单,而 Warp 处理器中嵌入性能分析器和动态软硬件划分模块所涉及的硬件方面的实现过于复杂,因此修改和改进比较困难.而 Erik Anderson 等人设计的硬件线程接口则需要硬件化的操作系统服务支持,通用计算机上很难满足这些条件.

本文提出了一种 CPU/FPGA 混合架构上硬件线程模型,并设计了该架构上的硬件线程执行机制,以硬件线程的方式使用可重构资源.同时,软硬件线程可以通过共享数据存储方式进行多线程并行执行.我们将程序中运算量较大的部分由 FPGA 上的硬件线程执行,而控制部分交给 CPU 上的软件线程执行,有效地利用 CPU 和 FPGA 的特性,从而提高整体计算性能.

本文使用 Simics 仿真软件模拟了一个软硬件混合架构平台,利用软硬件多线程改造后的 DES,MD5SUM 和归并排序算法进行了测试.实验结果表明,平均执行加速比达到了 2.30,说明本文所提出的软硬件线程方案能够充分利用 CPU/FPGA 混合架构的性能优势.

本文第 1 节介绍了本文所基于的 CPU/FPGA 混合体系结构,并提出该架构上的硬件线程模型.第 2 节提出了该混合架构上硬件线程的执行机制,包括硬件线程的控制和执行,以及软硬件线程之间的协同.第 3 节阐述系统的实验平台和在此平台上进行的实验结果与分析.第 4 节是对全文的一个简单总结.

1 CPU/FPGA 混合架构上的硬件线程模型

1.1 CPU/FPGA混合架构

最早的可重构计算机是在 1960 年,Gerald Estrin所提出的“固定+可变”机器[8],这台机器有一个固定的主处理器和一块可重配置硬件作为应用程序加速器.之后所设计开发的可重构计算机大多基于这个模型,因此,可重构计算机通常由零个或多个处理器,一个或者多个可重配置单元,以及内存所组成.

根据 CPU 和可重构逻辑的耦合紧密程度不同,Compton 和 Hauck 提出将可重配置架构分为 4 种类别[9].Todman等人扩展了前者的工作,将可重配置架构分为 5 类.本文所基于的CPU/FPGA混合架构为五类中的一类,如图 1 所示.图 1 表示FPGA通过系统IO总线和CPU连接作为独立设备与CPU协作.CPU通过输入输出系统和FPGA进行数据通信.因为数据通信速度相对比较慢,因此这种耦合类型的架构适合于在FPGA上进行大量计算,而不需要与CPU进行过多的数据交互的应用.虽然这种架构传输速度慢,但是实现比较方便,是普遍被采用的一种架构.

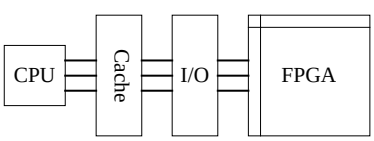

Fig.1 The oriented CPU/FPGA hybrid architecture
图1 本文面向的 CPU/FPGA 混合架构

1.2 硬件线程模型

基于第 1.1 节所述的 CPU/FPGA 混合架构,本文选择以共享内存多线程库的形式来实现软硬件混合多线程,并设计了名为 HwThreads 线程库,为多线程编程提供语言上的支持,以库的方式提供硬件线程编程接口.

基于 HwThreads 库的硬件多线程结构如图 2 所示,以内核模式和用户模式混合方式实现.线程的创建/结束,信号机制,上下文切换机制通过操作系统内核完成;而线程的管理、同步、线程私有数据管理等则在用户模式下完成.HwThreads 库提供如下类接口:

- 基本线程以及线程属性接口
- 线程优先级,调度接口
- 互斥锁以及互斥锁属性接口
- 条件变量以及条件变量属性接口

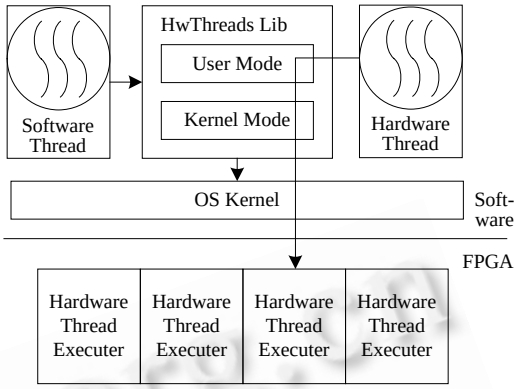

Fig.2 The software and hardware threads structure based on HwThreads Lib.

图1 基于 HwThreads 库的软硬件线程结构

该多线程结构中的线程类型分为软件线程和硬件线程,其中硬件线程又分为两个部分:第 1 部分是在软件层面的一个硬件线程控制块;第 2 部分是配置在 FPGA 上的一个规格一致的硬件线程执行器(hardware thread executer).硬件线程控制块遵循线程的基本接口行为和其他线程以及线程管理器进行交互通信.硬件线程执行器是一个硬件执行单元,负责执行线程函数的计算任务,操作系统通过软件层的线程控制块对其进行管理.线程执行器以有限状态机控制执行,其状态包括:初始化、就绪、运行、等待、结束.

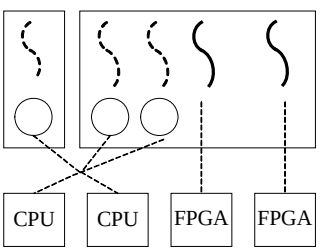

Fig.3 The software and hardware threads mapping mode.

图2 软硬件线程映射方式

HwThreads 库建立在用户模式和内核模式上.应用程序调用 HwThreads 线程库接口进行线程创建、线程管理、线程间同步和信号传递.对于不同类型的线程,HwThreads 中线程创建函数将区别对待:针对软件线程,创建线程控制块,分配线程寄存器和线程堆栈;针对硬件线程,创建线程控制块,分配寄存器,配置 FPGA 执行模块.

图 3 所示的是系统中运行在两个进程中的多个软硬件线程执行时状态.其中黑色虚曲线表示软件线程,黑色实心曲线表示硬件线程,圆形表示一个软件线程对应的内核轻量级进程,这些内核轻量级进程在 CPU 上独立执行,如果有多个处理器和处理器核的话,每个轻量级进程也可以并行运行在多个处理器或者处理器核上.而硬件线程则运行在针对特定硬件线程函数逻辑配置好的线程执行器上.

2 硬件线程执行机制

2.1 硬件线程控制块

如前文所述,软件层面的硬件线程控制块遵循线程的基本接口行为和其他线程以及线程管理器进行交互通讯,系统通过硬件线程控制块管理硬件线程.通常的一个进程中包含了多个线程,其中有软件线程和硬件线程,以及进程号、用户号、组号、文件句柄、锁等共享资源.软件线程都有自己的线程号,线程参数,线程函数体,该函数的参数,栈指针,寄存器,PC,线程状态,线程堆栈等;而一个硬件线程则有自己的线程号,控制寄存器,线程状态寄存器,数据寄存器以及一个配置了线程执行逻辑的线程执行器.

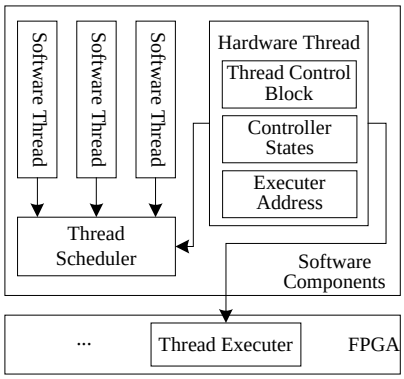

Fig.4 The structure of hardware thread control block.

图2 硬件线程控制块结构

图 4 中描述了软件线程(software thread)、硬件线程控制块(hardware thread control block),以及硬件线程执行器(hardware thread executer)之间的关系.每一个硬件线程由一组线程控制块和线程执行器组成.操作系统可以通过硬件线程控制块获取硬件线程执行状态,硬件线程执行结构,并对线程进行控制.

系统将 FPGA 线程执行器的状态位和控制位通过内存映射 mmap 将其映射到硬件线程控制器地址空间.如图 4 所示,当一个进程中调用 hwthread_create 函数创建一个硬件线程的时候,系统分配一块内存空间给硬件线程控制器,同时判断硬件线程执行器是否已经配置好,否则将新的硬件线程执行逻辑装载到 FPGA 设备上,FPGA 设备上的线程执行器状态由 FREE 变成 USED,然后通过系统调用将该硬件执行器的控制,状态和数据寄存器映射到线程控制器地址空间.最后将该线程的控制块加入到线程调度器就绪队列等待分发.一旦一个硬件线程获得了执行机会,线程控制器寄存器读写传递一个 EXECUTE 命令激活线程执行器,线程的执行逻辑就开始执行.

这种线程控制器的优点是将硬件线程的控制封装在控制结构中,传统的线程调度器只需要做很少的修改就可以调度硬件线程.

2.2 硬件线程执行器

硬件线程模块设计为规格一致的线程执行器形式,属性有如下 3 点:

1) 线程逻辑面积大小.线程逻辑面积是硬件线程在可重构设备上所占据的面积尺寸,包括所使用到的可重构逻辑单元、可编程 I/O,以及可重构网络的面积;
2) 时钟频率.时钟频率是硬件线程所运行的频率.时钟频率的作用是为了使硬件线程能够和系统中的内存、I/O 设备有一个共通的通讯时钟节拍;
3) 线程模块接口.为了和可重构设备外部进行通信,因此必须有一套统一的外部接口,线程模块接口的目的就是为了提供一套标准接口以支持硬件线程和外部设备通信.

为了能够充分利用可重配置设备上的面积资源,本文设计所使用的线程逻辑面积是均匀划分的,所以硬件线程可以配置到任意块单元面积上.这就要求每个面积单元能够提供一套标准的接口给硬件线程使用.如图 5 所示,每个线程硬件函数模块有一组输入输出接口.这组接口控制硬件线程执行器的执行、输入、输出和同步以及获取硬件线程执行器的状态.表 1 描述了线程执行器的各个端口功能.一个 FPGA 设备上均匀的划分出多个硬件线程,每个线程模块都遵循这个规格,目的是为了让多个硬件线程可以同时执行,并且减少单个线程执行器的配置时间,为硬件线程上下文切换提供便利.

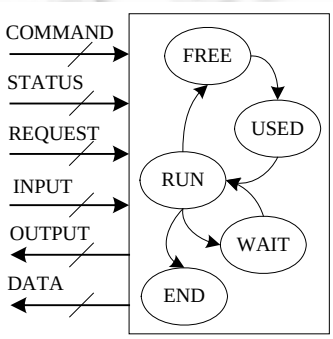

Fig.5 The hardware thread executor module

图3 线程硬件执行器模块

线程执行器上的 COMMAND 端口用来改变线程执行器的状态,命令信号包括 EXECUTE(0000 0001),RESET(0000 0010),STOP(0000 0100).EXECUTE 命令有两个作用:(1) 通知已配置的线程执行器开始执行;(2) 如果线程执行器状态为等待状态,那么 EXECUTE 命令唤醒线程执行器开始执行.RESET 命令通知线程执行器进行重置,并将状态位 STATUS 改成 FREE.STOP 命令信号的作用是暂停线程执行器当前的执行任务,将锁存器中的值保存到线程寄存器中.如果需要将该线程临时换出,那么进一步将线程执行器当前的执行状态保存到缓冲 RAM 中.

只有当线程执行器的状态 STATUS 为 USED 或者 WAIT 时候才能发送 EXECUTE 命令给线程执行器,在其他状态时该命令无效.当线程执行器状态为 USED 时,该执行器已经配置完成,可以开始运行,所以收到 EXECUTE 命令后线程执行器开始执行计算任务,线程执行器状态变成 RUN.当线程执行器状态为 WAIT 时,该线程执行器在等待其他线程所占据的互斥锁或某个条件变量.如果执行器成功获得被阻塞的资源,那么线程执行器响应 EXECUTE 命令,重新开始执行.

Table 1 IO ports of thread executor
表1 线程执行器的输入输出端口

端口	功能
COMMAND	命令信号改变线程执行器状态
STATUS	状态信号，告知线程封装器线程执行器的当前状态
REQUEST	请求访问信号，该信号指定需要访问通过 OUTPUT 指定的外部数据
INPUT	函数组件的输入信号，一个 32 位的整型，指定输入数据的地址
OUTPUT	输出一个 32 位的地址
DATA	如果是一个整型结果，则输出结果，否则输出该结果的地址

RESET 命令在任何时候都能初始化线程执行器,并改变线程执行器的状态为 FREE.

为了保证线程执行器在系统启动时的状态为 FREE,线程执行器在系统启动时将 COMMAND 信号置为 RESET.线程终止之后,如果该硬件线程是一个分离线程,那么线程封装器的 COMMAND 信号上给以个 RESET 命令,线程执行器变为 FREE 状态,这个线程执行器能在下次给一个新的线程使用.如果该硬件线程被其他线程联合等待,那么线程将执行结果写入线程控制块中的 result 变量中,完成这一步操作,线程执行器才能接受 RESET 命令重新进入 FREE 状态.

STATUS 信号端口是一个只读端口,用来获取线程执行器当前的状态,包括:FREE (0000),USED(0001),RUN (0010),WAIT(0011),END(0100).硬件线程执行器的状态转换如图 6 所示,当系统启动时或 RESET 命令之后,线程执行器的 STATUS 处于 FREE 状态.当一个线程在线程执行器上配置完之后,线程执行器的 STATUS 变为 USED 状态.

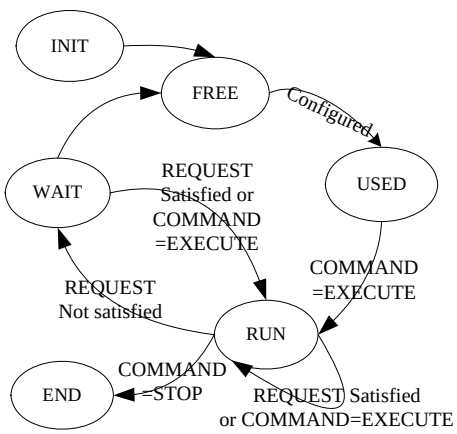

Fig.6 The status transfer of hardware thread executor
图4 硬件线程执行器状态转换图

在线程执行器配置完成后,其状态为 USED 时,COMMAND 命令置为 EXECUTE 后,硬件线程执行器开始执行,其状态变为 RUN.这个时候如果没有阻塞和退出操作硬件线程一直执行,直到线程执行器结束运行或者遇到等待互斥锁,条件变量或其他阻塞操作.

如果遇到线程等待互斥锁,条件变量或者其他阻塞操作时候,线程执行器在 REQUEST 上发送一个请求互斥锁,条件变量或者其他资源的请求,线程开始等待,进入 WAIT 状态,直到所请求的资源可用.当线程执行器请求的资源可用时,线程执行器恢复执行,其状态改为 RUN.

当线程执行结束后进入 END 状态,如果有运算结果,线程执行器将运算结果发送到 DATA 端口.并且根据 OUTPUT 所指定的结果类型确定 DATA 上的数据读取方式.

REQUEST 请求信号用来发送线程执行器对互斥锁,条件变量,以及输入输出等的请求. REQUEST 信号包括 :MUTEX_LOCK(0000 0001) 请求一个互斥锁 ;MUTEX_UNLOCK(0000 0010) 请求释放一个互斥锁;COND_WAIT(0000 0100)等待一个条件变量信号;COND_SIGNAL(0000 0101)发出一个条件变量信号;COND_BROAD(0000 0110)广播一个条件变量信号;WRITE(0000 1001)一个写数据请求;READ(0000 1010)一个读数据请求.

2.3 软硬件线程协同执行

混合架构上软硬件线程协同执行的装载、运行过程如图 7 所示,父线程(father thread)执行时遇到一个线程

构造指令(thread construct instruction),那么其本身在运行的同时,调用线程管理模块中的线程构建函数,构造新的线程.由于软硬件线程的内在的差异性,线程构造函数分为两种类型,即软件线程构造函数和硬件线程构造函数.

软件线程调用线程函数库中的 hwthread_create 函数构造需以参数传入的函数为线程执行体的一个软件线程.函数在参数中设定所调用的线程体是硬件部分还是软件部分.如果所调用的是函数体是 CPU 上执行的软件部分,那么这个线程创建函数创建一个软件线程.如果参数中设定线程是一个 FPGA 上配置的硬件线程,那么系统将创建硬件线程.无论硬件线程还是软件线程,其在系统中都有一个线程控制块,其结构核心数据定义如下:

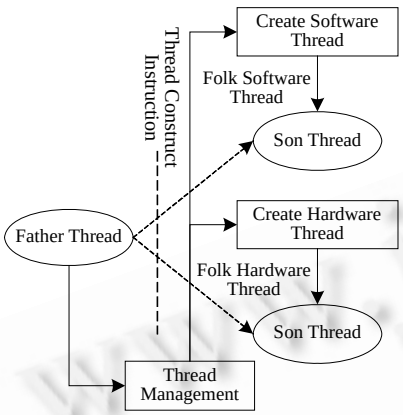

Fig.7 The process of software and hardware thread creation and execution.

图4 软硬件线程协同执行的创建运行过程

```
struct hwthread
 {
    pid_t tid;
    pid_t pid;
    bool hwsw_flag;
    struct hwthread_key_data *specific[KEY_SIZE];
    bool specific_used;
    bool stopped_start;
    struct hwthread *join;
    void *result;
    int schedpolicy;
    union {
       void *(*start_routine)(void *);
       void *controller;
    };
    void *arg;
    int status;
 }
```

该结构中,*pid*,*tid* 分别对应线程所在的进程号和线程本身的线程号.*hwsw_flag* 用来判断该线程是软件线程还是硬件线程,如果该线程是硬件线程,即 *hwsw_flag* 的值为 1,那么通过 *controller* 地址开始的内存映射寄存器访问控制硬件线程执行器,线程执行器的状态寄存器的值将映射到 *status* 和 *stopped_start* 上.如果该线程为软件线程,即 *hwsw_flag* 为 0,那么调用 *start_routine* 作为线程体函数执行.

3 实验结果与分析

3.1 实验平台设置

本文选用 Virtutech Simics 作为仿真实验平台,Simics 是一个商业和科研上应用比较主流的全系统仿真软件,能够完整的对一个系统进行仿真,并提供了一套完全的设备描述接口,开发者可以通过 DML,C,或者 Python 语言调用设备接口开发自己的硬件设备供系统使用.

在 Simics 上可以有两种模式对设备进行仿真,一种方式是基于事件模式对设备进行仿真;另一种方式是基于数据流模式对设备进行仿真.

本文实验中,FPGA 设备提供给系统一个基本的硬件配置和数据访问接口,而内部的实现则基于事件模式用 DML 语言进行仿真,其规模可以通过计算逻辑功能换算成门电路.

Simics 配置了一个双处理器核的 CPU,每个处理核频率为 200MHz,存储器大小为 256M,可重配置设备频率为 66MHz,门电路为 300 万规模.另外每个处理器核的一级 cache 和二级 cache 分别为 32kB 和 512kB.硬件线程输入数据缓存的大小分别设置为 128KB.

3.2 实验结果和分析

由于仿真平台的性能比较简单,因此本文选择了相对简单的多线程程序,包括归并排序,DES 算法,以及 MD5SUM 算法多线程实现,其中:

- 使用 DES 算法对 100KB 数据进行加密;
- 使用 MD5SUM 程序对 50KB 数据进行验证计算;
- 使用归并排序算法对一个包含 80K 个不同 32 位乱序整数数组进行排序.

实验中线程执行器的端口通过内存映射到内存地址 0x1000,例如第一个线程执行器的 COMMAND 端口映射到 0x1000,STATUS 端口映射到 0x1004,REQUEST 端口映射到 0x1008,INPUT 端口映射到 0x100C,OUTPUT 端口映射到 0x1010, DATA 端口映射到 0x1014. 本文中的线程执行器设计为大小规格一致的形式,其规格如表 2 所示,可以看出来,线程执行器中控制接口、线程库系统函数占用了线程执行器单元资源的 30%左右.

Table 2 Resource usage of hardware thread executor.
表 2 硬件线程执行器资源使用率

资源	线程系统	执行器	百分比
slice	2534	7869	32.2%
锁存器	2378	9666	24.6%
查找表	4320	13471	31.4%

Table 3 Speedups of 3 algorithms with different number of software and hardware thread combination.
表 3 三种算法在不同线程数目下的执行加速比

线程组合	DES	MD5SUM	MergeSort	平均
2 SW/1 HW+1 SW	1.80	2.01	1.69	1.84
3 SW/1 HW + 2 SW	1.27	1.38	1.49	1.38
3 SW/1 HW + 2 SW	2.77	2.46	2.23	2.49
4 SW/2 HW + 2 SW	1.65	2.27	3.19	2.37
4 SW/3 HW + 1 SW	2.64	3.46	4.12	3.41
平均加速比	2.03	2.32	2.54	2.30

3 种算法不同软硬件线程配置下运行的实验结果如图 8 所示.3 种算法在不同线程数目下的执行加速比见表 3,DES 算法中,如果使用相同数目线程执行程序,一个硬件线程加一个软件线程比两个软件线程加速比为 1.80.一个硬件线程加两个软件线程比 3 个软件线程的加速比为 1.27;两个硬件线程加一个软件线程比 3 个软件线程的加速比为 2.77.两个硬件线程加两个软件线程比 4 个软件线程的加速比为 1.65;3 个硬件线程加一个软件线程比 4 个软件线程的加速比为 2.64.通过硬件线程加速后的平均加速比为 2.03.

MD5SUM 算法中,如果使用相同数目线程执行程序,一个硬件线程加一个软件线程比两个软件线程加速比为 2.01.一个硬件线程加两个软件线程比 3 个软件线程的加速比为 1.38;两个硬件线程加一个软件线程比 3 个软件线程的加速比为 2.46.两个硬件线程加两个软件线程比 4 个软件线程的加速比为 2.27;3 个硬件线程加一个软件线程比 4 个软件线程的加速比为 3.46.通过硬件线程加速后的平均加速比为 2.32.

MergeSort 算法中,如果使用相同数目线程执行程序,一个硬件线程加一个软件线程比两个软件线程加速比为 1.69.一个硬件线程加两个软件线程比 3 个软件线程的加速比为 1.49;两个硬件线程加一个软件线程比 3 个软件线程的加速比为 2.23.两个硬件线程加两个软件线程比 4 个软件线程的加速比为 3.19;3 个硬件线程加一个软件线程比 4 个软件线程的加速比为 4.12.通过硬件线程加速后的平均加速比为 2.54.

另外可知,由 3 个算法加速比的平均可得:使用相同数目线程执行程序,一个硬件线程加一个软件线程比两个软件线程加速比为 1.84.一个硬件线程加两个软件线程比 3 个软件线程的加速比为 1.38;两个硬件线程加一

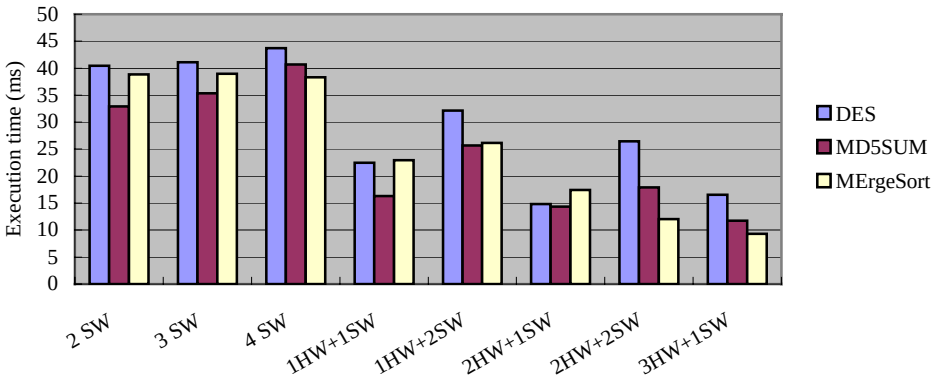

Fig.8 Execution time of 3 algorithms with different number of hardware and software threads
图5 3 种算法在不同线程数目下的执行速度

个软件线程比 3 个软件线程的加速比为 2.49.两个硬件线程加两个软件线程比 4 个软件线程的加速比为 2.37;3 个硬件线程加一个软件线程比 4 个软件线程的加速比为 3.41.平均加速比为 2.30.

4 结论和进一步研究

本文在最普遍的 CPU/FPGA 混合架构上提出了一个硬件线程模型,并设计了硬件线程执行机制,以硬件线程的方式使用可重构资源.同时,软硬件线程可以通过共享数据存储方式进行多线程并行执行.通过将程序中运算量较大的部分以 FPGA 上的硬件线程方式,而控制部分以 CPU 上的软件线程方式执行,有效地利用 CPU 和 FPGA 的特性,从而提高整体计算性能.

本文使用 Simics 仿真软件模拟了一个软硬件混合架构平台,利用软硬件多线程改造后的 DES,MD5SUM 和归并排序算法进行了测试.实验数据表明,平均执行性能加速比达到了 2.30,验证了本文所提出的软硬件线程方案能够有效发挥 CPU/FPGA 混合架构的性能优势.本文针对硬件线程的执行机制进行了深入讨论,但是对线程间的同步机制、信号机制,以及线程调度策略方面还没有进行深入,有待进一步地研究.

References:

[1] Mishra SM, Cabric D, Chang C, Willkomm D, van Schewick B, Wolisz A, Brodersen RW. A real time cognitive radio testbed for physical and link layer experiments. In: Proc. of the 2005 IEEE Symp. on New Frontiers in Dynamic Spectrum Access Networks. 2005. 562–567.

[2] Lin EC, Yu K, Rutenbar RA, Chen T. A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA. In: Proc. of the 15th ACM/SIGDA Int'l Symp. on Field Programmable Gate Arrays. 2007. 60–68.

[3] Ortigosa EM, Ortigosa PM, Cañas A, Ros E, Agís R, Ortega J. FPGA implementation of multi-layer perceptrons for speech recognition. In: Field-Programmable Logic and Applications. Berlin: Springer-Verlag, 2003. 1048–1052.

[4] Dydel S, Bala P. Large scale protein sequence alignment using fpga reprogrammable logic devices. In: Field Programmable Logic and Application. Berlin: Springer-Verlag, 2004. 23–32.

[5] So HK. BORPH: An Operating System for FPGA-Based Reconfigurable Computers [Ph.D. Thesis]. Berkeley: University of California, 2007.

[6] Stitt G, Vahid F. Thread warping: A framework for dynamic synthesis of thread accelerators. In: Proc. of the 5th IEEE/ACM Int'l Conf. on Hardware/Software Codesign and System Synthesis. 2007. 93–98.

[7] Anderson E, Agron J, Peck W, Stevens J, Baijot F, Komp E, Sass R, Andrews D. Enabling a uniform programming model across the software/hardware boundary. In: Proc. of the 14th Annual IEEE Symp. on Field-Programmable Custom Computing Machines. 2006. 89–98.

[8] Estrin G. Reconfigurable computer origins: The ucla fixed-plusvariable (f+v) structure computer. IEEE Annals of the History of Computing, 2002,24(4):3–9.

[9] Compton K, Hauck S. Reconfigurable computing: A survey of systems and software. ACM Computer Surveys, 2002,34(2): 171–210.

陈天洲(1970－),男,浙江丽水人,博士,教授,博士生导师,主要研究领域为计算机体系结构,多核计算,嵌入式系统,节能计算.

胡威(1979－),男,博士,讲师,主要研究领域为计算机体系结构,嵌入式系统.

严力科(1982－),男,博士生,主要研究领域为可重构计算.

马吉军(1983－),男,硕士,主要研究领域为可重构计算.