支持多核的嵌入式操作系统关键技术研究

何 翔,任晓瑞

(中航工业西安航空计算技术研究所,陕西 西安 710119)

摘 要:以分析当前支持多核的操作系统需实现的关键技术为基础,从多核操作系统的引导和初始化、多核操作系统任务管理、多核中断、核间通信以及核间同步与互斥等方面具体分析、研究支持多核的嵌入式操作系统的实现机制,提出了一种多核领域操作系统关键技术的解决思路。

关键词:多核;嵌入式操作系统;中断机制;核间通信;任务管理

中图分类号:TP316 **文献标识码**:A **文章编号**:1671-654X(2013)04-0086-05

Key Technical Research on Supporting Multi-core Embedded Operating System

HE Xiang, REN Xiao-rui

(*Xi'an Aeronautics Computing Technique Research Institute, AVIC, Xi'an* 710119, *China*)

Abstract: Based on analysis of key technical points supporting multi-core operating system, this paper analyzes the Bootstrap and initialization of multi-core operating system, multi-core OS task management, multi-core OS interrupt mechanism, Inter-Process Communication (IPC) mechanism, Inter-Process synchronization and mutex mechanism, and other specific aspects. This paper studies the realization mechanism of the embedded operating system which supports multi-core architecture. At last, the paper presents the solution of key technology of multi-core OS.

Key words: multi-core; embedded operating system; task management; interrupt mechanism; inter-process communication

引言

近年来,随着电子技术的发展,对嵌入式系统提出了高性能、低功耗、小型化、低成本的要求[1]。通过提高处理器频率来提升系统性能的方法受到功耗、成本以及体积的限制,而单芯片多处理器(Chip Multi-Processors,简称CMP也称多核)能够在不提升处理器频率的情况下提升处理器性能,从而提升系统性能,规避了以上问题。因此多核处理器的研究倍受学术界和工业界的关注。

目前多核处理器已经广泛应用于大规模计算和PC领域,鉴于多核处理器系统的高性能、高集成度、低功耗、低成本的优势[2],多核处理器应用于嵌入式系统已是大势所趋。

嵌入式系统对实时性、安全性、可靠性等方面有较高的要求。为了更好地发挥多核处理器在嵌入式系统中的性能,满足嵌入式系统的要求,需要将系统内部的资源(尤其是计算资源)更加合理、有效地管理起来,提高资源使用效率[3]。这就对支持多核处理器的嵌入式操作系统提出了更高的要求。

1 支持多核的嵌入式操作系统

处理器的发展由单核到双核再到四核、八核以至更多。多核环境下计算资源得到充分的扩充,系统共享资源的竞争访问问题将更加突出,系统的性能将会受制于操作系统对资源的管理和调度,近年来支持多核架构的操作系统的研究已在各大高校和研究机构成为热点。

1.1 支持多核的操作系统研究现状

多核应用于嵌入式领域的研究不断升温。美国麻省理工学院开发的FOS(Factored OS)[4]多核操作系统原型针对多核架构提出了一个解决缓存竞争和进程切换开销问题的新思路,采用空间复用(Space Multiplexing)取代时间复用(Time Multiplexing),将系统的各个服务平均分配且固定到每个计算单元上,从而解决

收稿日期:2013-05-07 **修订日期**:2013-06-13

基金项目:航空科学基金项目资助(2012ZC31006);十二五总装预研课题(51315030101)

作者简介:何 翔(1987-),男,陕西西安人,助理工程师,硕士研究生,主要研究方向为嵌入式实时操作系统。

系统服务相互竞争问题。美国斯坦福大学设计的 Disco 操作系统采用虚拟化技术,由 Disco 将整个系统资源管理起来,并向上层提供多个虚拟机,虚拟机上可运行其他操作系统。该设计的 CPU 核组织形式自由,实现了强可伸缩性。以上两个操作系统原型都提出了新颖的解决多核效率问题的方案。

目前相对成熟的支持多核处理器的嵌入式操作系统主要有 WindRiver 公司的 Vxworks, QNX 公司的 Neutrino 以及由 Linux 发展而来的 Monta Vista Linux 等操作系统。其中 WindRiver 公司的 Vxworks 系列的应用较为广泛。

1.2　支持多核的嵌入式操作系统的特点

多核系统在性能功耗比方面相比于传统单核系统有很大的优势。嵌入式多核操作系统能够保证系统的实时性、安全性,合理分配竞争资源,保证任务执行的正确性。多核系统设计的复杂性不能转嫁给用户,保持系统的简单易用[5]。除对多核系统有特殊要求外,多核对用户是透明的,这需要操作系统在用户界面和编程接口方面与单核系统保持一致或相似。

2　支持多核的嵌入式操作系统关键技术

在传统单核操作系统的基础上,多核操作系统还需要解决以下关键技术:多核操作系统的引导和初始化、多核操作系统任务管理、多核中断机制、核间通信机制以及核间同步与互斥机制。

2.1　多核操作系统的引导和初始化

多核系统在运行时,每个核是对称的,没有主次关系,但是在系统启动和初始化阶段,核心的启动顺序和引导方式是不对称的。操作系统内核的初始化只能由一个 CPU 核心来完成。系统上电后引导 CPU(Bootstrap CPU)被激活,而其他 CPU 处于未激活态。引导 CPU 完成自身资源以及运行时所需要的资源等必须资源的初始化,完成系统硬件平台初始化,将操作系统镜像复制到 RAM 中,初始化操作系统内核。完成以上工作后引导 CPU 开始为其他 CPU 的启动做准备。在激活其他 CPU 之前将非引导 CPU 在 RAM 中执行代码的入口地址写入 ROM 中非引导 CPU 执行代码指向的位置。激活非引导 CPU,非引导 CPU 得到激活信号执行 ROM 中代码,完成自身私有资源初始化,读取引导 CPU 写入的 RAM 中代码的入口地址,并完成剩余初始化工作,设置自身为在线状态并进行调度,引导 CPU 查询到非引导 CPU 为在线状态后结束非引导 CPU 启

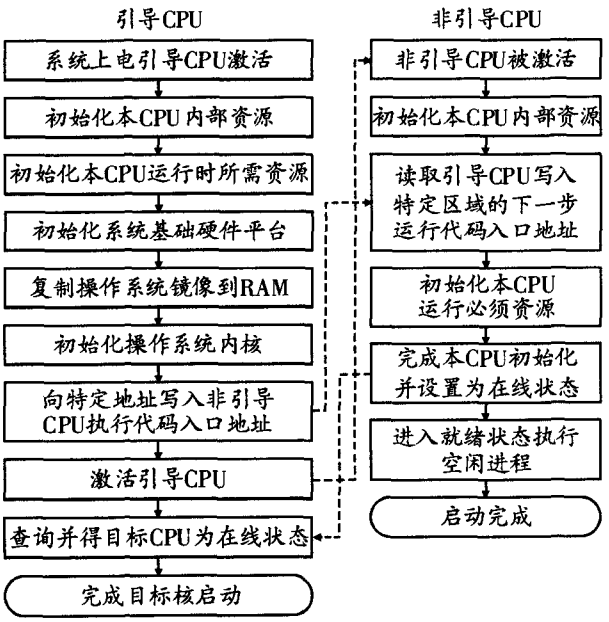

图 1　多核操作系统非引导 CPU 启动过程

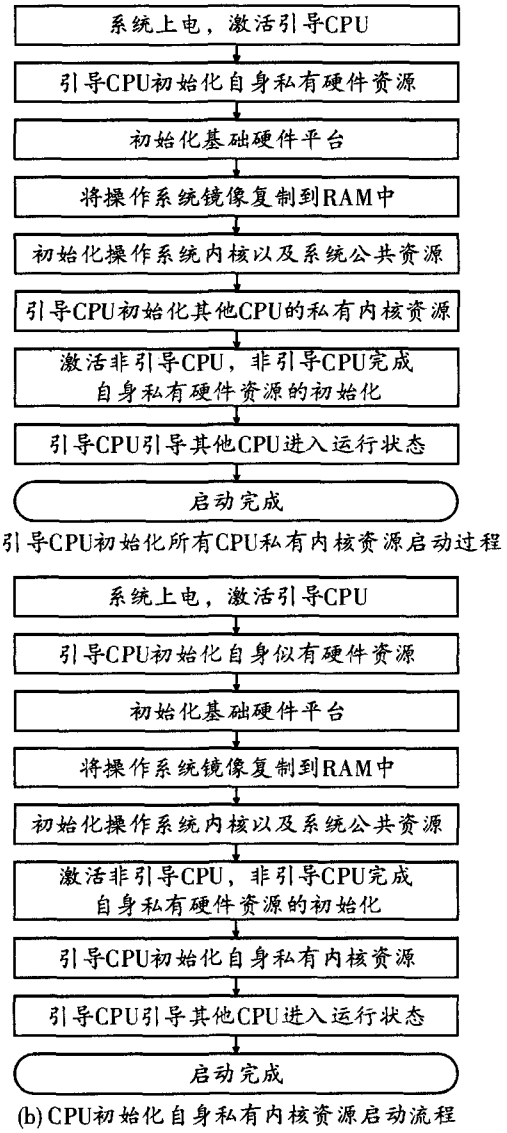

图 2　多核系统 CPU 私有内核资源初始化方式

动的引导工作(如图1)。

多核系统中,每个 CPU 在被激活后都是从 ROM 中的入口地址开始执行代码。ROM 中的系统启动引导程序需要根据 CPUID 判断本地 CPU 是否为引导 CPU,引导 CPU 启动过程中需要完成整个系统初始化,并引导其他 CPU 完成自身的启动和初始化过程,在这一启动过程中,CPU 所做的工作是不对称的。

图1所示的操作系统启动过程中,空闲进程以及 CPU 私有栈等 CPU 内核私有资源的初始化可以由引导 CPU 统一完成,也可以由 CPU 自身完成(如图2)。

采用非引导 CPU 初始化自身私有内核资源,CPU 私有资源使用灵活,扩展性强。但缺点也很明显,每个 CPU 的私有资源组织结构松散,不方便集中管理。

采用引导 CPU 统一初始化每个 CPU 的系统私有内核资源,CPU 私有内核资源组织结构统一,方便管理,结构简单,资源使用速率和效率高,空间开销小,能够满足嵌入式系统对功能、性能的要求。

2.2 嵌入式多核任务管理

如何通过恰当的调度机制和策略保证多核的负载均衡,并充分发挥各个内核的性能特点、提高整个系统的吞吐量和实时性,是支持多核的嵌入式系统研究中的核心问题。

就绪队列的结构和任务调度的方法决定了操作系统调度效率。

2.2.1 就绪队列结构

当前多核操作系统任务就绪队列主要有全局队列、局部队列和共生队列三种结构。

1)全局队列是指操作系统维护一个全局的任务就绪队列,当 CPU 空闲,就从该任务就绪队列中取出任务执行,这种队列结构能够有效地解决多核系统负载平衡问题,并且实现相对简单。缺点也是显而易见的,当系统 CPU 数量增加或者各 CPU 频繁进行任务调度时,CPU 对全局就绪队列的竞争使用会降低系统运行效率。

2)局部队列是指操作系统为每个 CPU 内核都维护一个私有的任务就绪队列,每个内核都从自己的任务就绪队列中取出任务执行。局部队列调度中任务无须在各个内核之间频繁切换,提高缓存命中率,同时解决多核负载平衡问题调度器会占用一部分系统资源,使系统效率降低。

3)共生队列结构是全局队列和局部队列的综合体,在操作系统的就绪队列中既有全局队列也有局部私有队列,该方法具有以上两种队列结构的优点,同时可以通过调度规则避免以上两种队列结构的缺点。

2.2.2 任务调度方法

任务调度是合理、高效地将任务分配到给每个 CPU,同时满足任务的执行要求。为了达到该目的,需要针对操作系统所要执行任务的特点选择合适的调度方法。目前主流的调度方法分为静态优先级调度方法和动态优先级调度方法。静态优先级在任务创建时被调度器分配一个优先级,直到任务结束都不会变化。动态优先级在调度时根据制定的策略,从就绪队列中选择最合适的任务运行,调度中任务的优先级会发生变化。动态优先级调度方法对系统资源的分配效率更高,但同时也存在调度算法复杂,系统开销大的缺点。静态优先级调度算法结构简单、功能易实现、系统开销小,并且能够保证高优先级的任务在第一时间完成。鉴于静态优先级调度算法的特点能够很好地满足嵌入式系统的功能、性能要求,所以选择静态优先级调度算法是嵌入式操作系统调度算法的首选。

支持多核的嵌入式操作系统为了保证任务的实时性,采用优先级抢占调度算法,时刻保证高优先级的任务能够占有 CPU。多核结构下,同一时刻有多个任务在运行,用 RunTasks[MAX_CPU]数组表示各个 CPU 中的运行任务,当 RunTasks[MAX_CPU]中存在优先级低于就绪队列中某个任务的优先级的任务时系统进行任务调度。假设目前所有 CPU 运行任务的优先级都高于就绪队列中的任务,则系统在以下情况执行调度函数并切换任务上下文能够满足优先级抢占策略:

1)有新任务添加到就绪队列并且就绪队列中最高优先级高于运行任务的最低优先级;

2)占有 CPU 的运行态任务释放 CPU。

在优先级抢占调度方法的基础上,使用一些调度优化方法可以提高多核系统的运行效率。

根据 CPU 缓存的结构以及系统执行任务的特点对所有 CPU 进行划分,形成"调度域"(如图3),任务可以映射到一个域中,同一个"调度域"可以被多个任务映射,任务与"调度域"多对一的关系。操作系统设计者可以将相关性强的任务映射到一个"调度域",这样做能够提高 CPU 本地缓存的命中率从而提高多核系统的运行效率。

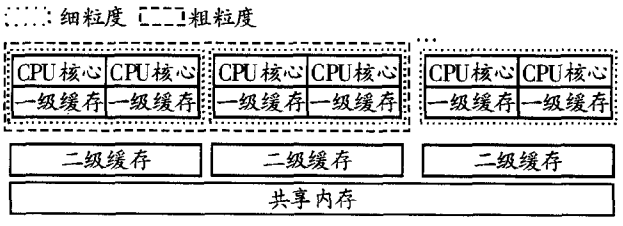

图3 调度域划分

处理器亲和性(affinity)技术也是一种提高系统效率的方法。CPU 亲和性是进程要在某个给定的核上尽量长时间地运行而不被迁移到其他核的倾向性。亲和性分为软亲和性和硬亲和性,软亲和性中任务尽可能地在一个特定 CPU 上运行而不会频繁地在核间迁移,而硬亲和性中任务只能在指定 CPU 上运行。

针对嵌入式操作系统的实时性特点,本文在优先级抢占调度方法的基础上提出采用混合就绪队列调度结构结合硬亲和性技术的调度结构。每个 CPU 核心维护一个私有硬亲和性就绪队列,系统维护一个全局就绪队列。在任务创建时系统通过给任务控制块(TCB)的 affinity 属性赋值来绑定 CPU,任务变为就绪态时 affinity 属性为空则该任务进入全局就绪队列,affinity 为某个 CPU 的 ID 则说明该任务被绑定在一个特定的 CPU 上,该任务进入相应 CPU 的私有就绪队列。该调度结构和方法在提高了系统的运行效率的同时也解决了负载平衡问题。

2.3　多核中断系统

传统单核处理器通常采用一个外部中断控制器来处理整个计算机系统的中断。多核处理器系统中,情况有所不同,每个处理器都必须有各自本地中断控制器,并与 CPU 内部的一个全局的中断控制器连接。全局中断控制器负责接收外部中断并分发给每个核的本地中断控制器,中断控制器之间也可以相互传递或接收中断。

多核系统中的中断可分为三类:内部错误中断、软件中断,I/O 中断以及核间中断(IPI)。同时内部错误中断,软件中断属于内部中断。I/O 中断属于外部中断。核间中断是指不同 CPU 之间的中断(如图 4)。

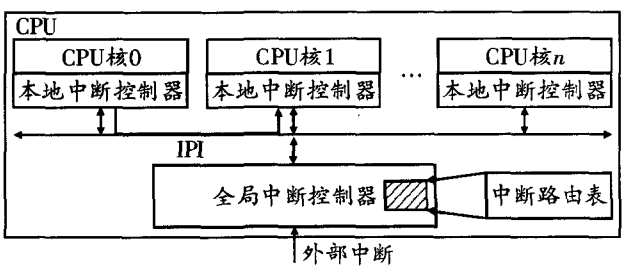

图 4　多核中断系统

全局中断控制器负责将来自外部设备的中断请求分配给处理器内各 CPU 核。对于每一个中断路由向量,可以采用动态和静态两种分配方式。采用中断路由向量静态分配模式,全局中断控制器将该中断请求分配给预设的一个或多个核;采用中断路由向量动态分配模式,可以按照一定的策略发送给特定核。以上两种方式都必须要考虑 ISR 的负载均衡问题,在保证系统要求的前提下,提高系统 ISR 的并行度从而提高系统效率。

多核中断系统中核间中断也是必不可少的。按照完成的任务不同,可将核间中断分为以下两种:

1)重新调度中断:源核向目标核发送中断请求,告知目标核需要进行调度,同时向目标核提供所需参数,目标核收到该中断后会根据操作系统调度策略进行任务调度。

2)请求执行中断:源核向目标核发送中断请求,请求其执行某个特定功能的函数。该中断在多核系统中是必不可少的。例如通过该中断使两个核心协作使用或修改核间私有资源。该中断为核间中断类型的扩充提供了基础。

为了提高系统执行效率和实时性,需要在以上两种核间中断的基础上进行核间中断种类的扩充。将实时性要求高且使用频率高的核间中断任务分离出来,设计为新的核间中断向量从而缩短中断响应时间。

基于以上设计思路,本文针对嵌入式操作系统的特点对核间中断划分为四种:请求执行中断、多核调试中断、重调度中断、进程间通信中断。该核间中断向量的划分保证了核间中断功能性的全覆盖,将相对使用较多的多核调试和进程间通信的相关中断分离出来形成专用的核间中断向量,提高了中断系统的执行效率。

2.4　核间通信机制

处理器间通信主要有两种方式:采用核间中断方式和基于共享内存的同步互斥访问方式。

核间中断方式中,处理器通过向其他处理器核发送处理器间中断,目标处理器核接收中断并相应,进入中断服务程序,执行通信函数与源处理器核进行通信。

基于共享内存的同步互斥访问方式中,处理器并发执行发送请求后等待被请求处理器核响应后才能继续执行。或者采用信号量的方式互斥访问临界资源,从而达到通信的目的。

核间中断方式能够快速响应,但是传输的数据量有限,基于共享内存的同步互斥访问方式,通过内存读写完成通信,能够进行大数据量通信。通过以上两种核间通信方式的组合,可以满足操作系统对于处理器核间通信的要求。

2.5　核间同步与互斥机制

传统支持单核处理器的嵌入式操作系统提供了可抢占锁和关中断两种基本的互斥机制,在这个基础上实现了信号量(Semaphore)、栅栏(Barrier)、事件(Event)等同步互斥与通信机制。然而在多核环境下同步和互斥需要考虑的因素更多、更复杂,传统的抢占锁和关中断已经不能满足对多核系统同步互斥的要求。

关中断在多核系统中可分为关本核中断和关所有核中断。关闭本核中断只能满足在本核心内互斥的要求,关闭全部中断可以满足系统互斥的要求,但同时也降低了系统效率,所以在关中断的使用中除必须情况,须减少关闭全部中断。

自旋锁是一个轻量级的锁,在解决核间同步互斥方面有很高的效率,因此多核系统都是以自旋锁为基础重新定义传统单核同步互斥机制。

原子内存操作为系统提供了原子的读、写或者修改某特定区域内存的能力,该操作能够方便程序在不考虑其他处理器核心的情况下安全完成内存读写操作。

通过为系统经常使用的内核资源设计专门的锁机制——内核锁,能够在保证系统资源安全使用的同时提高系统效率。

为了提高系统的运行效率,在对系统公共资源竞争使用的加锁中,需保证获取互斥锁的处理器在安全使用公共资源的同时,不占用与本处理器核不相关的资源,做到细粒度加锁。

3 结束语

随着国防、通信、航空、工业控制等领域对于嵌入式系统的高性能、低功耗、小型化提出更高的要求,以往基于单核的系统架构已经不能满足,而多核架构的产生有效地解决了以上问题。本文总结并分析了支持多核的嵌入式操作系统在实现中会遇到的问题及其解决思路,为后续开展支持多核的嵌入式操作系统研究和设提供了参考。

参考文献:

[1] Kunle Olukotun, Basem A Nayfeh, Lance Hammond, et al. The Case for a Single Chip Multiprocessor[C]. Proc 7th Int' l Conf Architectural Support for Programming Languages and Operating systems. New York: ACM Press, 2011.

[2] Krishna K Rangan, Wei Gu - Yeon. Thread Motion: Fine - grained Power Management for Multi - core Systems [J]. ACM SIGARCH Computer Architecture News, 2009, 37 (3): 302 - 313.

[3] Schneider S. Multiprocessor Support for the Fiasco Micro - kernel [D]. Chemnitz, Germany: Technical University of Chemnitz, 2010.

[4] David Wentzlaff, Anant Agarwal. Factored Operating Systems (Fos): the Case for a Scalable Operating System for Multicores[J]. ACM SIGOPS Operating Systems Review, 2009, 32(2): 76 - 85.

[5] 任晓瑞,时磊. 支持对称多处理器结构的操作系统设计[J]. 航空计算技术,2008,38(2):53 - 57.

(上接第85页)

迭代次数到50以后,重量基本没有变化,这是因为迭代到50次时,优化设计变量 *beam*11 没有达到收敛条件,而 *beam*11 对重量的影响较小。

采用本文的计算方法,经过96次迭代后,从图3和图4可以看出,结构重量在满足约束条件下获得最优解,结构重量由427.44 kg下降到309.43 kg,与依据经验布置梁位置的飞翼布局机翼的重量相比,优化后的重量下降了27.6%。并且,最后使模型的最大应力由87 MPa提高为113 MPa,没有超过许用应力值,翼尖位移由248 mm上升为364 mm,在约束范围内。

4 结语

本文对机翼优化时,分离位置和尺寸参数,将尺寸参数作为内部优化设计参数,把位置参量作为外部优化设计参数,基于iSIGHT的框架,可以实现不同求解器的连接,并可监测结果的变化,这样对研究的结果有更好的把握,从而可以对不同学科的耦合问题实现求解,在工程上具有一定适用性。从算例也可以看出,优化的结果达到预期要求。

参考文献:

[1] 陶梅贞. 现代飞机结构综合设计[M]. 西安:西北工业大学出版社,2001.

[2] 王伟,赵美英,常楠. 某型机翼内翼结构几何优化设计[J]. 强度与环境,2006,33(3):56 - 61.

[3] 王伟,赵美英,赵锋,等. 基于人工神经网络技术的结构布局优化设计[J]. 机械设计,2006,23(12):7 - 10.

[4] 王栋,张卫红,姜节胜. 桁架结构形状与尺寸组合优化[J]. 应用力学学报,2002,19(3):72 - 76.

[5] 解秋红,刘保华,李西双,等. 基于Hook - Jeeves算法的渤海现今应力场优化反演[J]. 工程力学,2012,27(6):279 - 284.

[6] 叶天麒,周天孝. 航空结构有限元分析指南[M]. 北京:航空工业出版社,1996.