一种异构多核处理器嵌入式实时操作系统构架设计

蒋建春[1,2] 汪同庆[1]
(重庆大学光电技术及系统教育部重点实验室 重庆 400044)[1]
(重庆邮电大学自动化学院 重庆 400065)[2]

摘 要 由于异构多核处理器和多处理器系统及同构多核处理器的构架存在很大差别,应用于多处理器系统的分布式结构以及应用于同构多核系统的主从式结构操作系统不能解决异构多核处理器的实时调度和效率问题。对异构多核处理器的特点及发展趋势进行了研究,提出了一种适用异构多核处理器的多主模式实时操作系统构架。这种构架将通信总线中的多主模式引入多核操作系统构架中,采用对称式结构及组件模式设计操作系统模型,使多核处理器中每个内核都可以作为主核实现对资源、任务的实时管理,提高系统性能,同时可以解决主从式操作系统存在的由于处理器核增多而带来的主内核不能满足系统性能要求的瓶颈问题。通过这种单一构架模型可以进行灵活配置,以适应不同结构及功能要求的处理器内核,降低操作系统开发难度。

关键词 异构多核处理器,嵌入式实时操作系统,多主内核,主从式内核
中图法分类号 TP316.2 **文献标识码** A

Architecture Design of Embedded Real-time Operating System for Heterogeneous Multi-core Processor

JIANG Jian-chun[1,2] WANG Tong-qing[1]
(Key Lab of Optoelectronic Technique and System of Ministry of Education, Chongqing University, Chongqing 400044, China)[1]
(College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)[2]

Abstract Because of the different structure between heterogeneous multi-core processor and symmetric multi-core processor, the traditional distributed operating system and multi-core operating system are not suitable for heterogeneous multi-core processor in real-time and efficiency performance. After researching characteristics and trend of developing of the heterogeneous multi-core processor, this paper presented a multi-master mode real-time operating system architecture, in which the multi-master mode in communication bus is introduced, and symmetric structure and modularization frame are used to design operating system architecture. In this operating system, every kernel may be master to manage resources, tasks and other shared devices to improve the real-time and efficiency performance, and to solve the bottleneck problem that the single master can not meet the requirement of performance with increasing of the slave number. This architecture can satisfy different structure processor core with configuring function module. On the other hand, it can also reduce the development difficult of the heterogeneous multi-core processor operating system.

Keywords Heterogeneous multi-core processor, Embedded real-time operating system, Multi-master kernel, Master-slave kernel

1 引言

随着微电子技术的发展,一种将具有不同计算能力的处理器融合到一起的异构多核处理器(HMP-Heterogeneous Multi-core Processor)被广泛应用于航空航天、工业控制、仪器仪表等行业,以满足系统性能需求,降低功耗和成本。HMP由于集成了不同特点和性能的处理器核,可以满足多种应用环境对系统实时性、功耗、可靠性和成本的要求,是未来多核技术的重要发展方向之一。但是,由于其异构特点,使得针对HMP的软件编程技术、软件编译技术以及操作系统等方面的研究还未完全跟上处理器本身的研究水平,从而使其应用推广受到限制和一定的阻力。怎样解决HMP编程过程中存在的效率低、代码可重用性低、处理器利用率低等问题是当前研究热点。当前对于多核处理器应用推广的关键技术之一的操作系统的研究主要针对同构多核,而对HMP的操作系统的研究较少,与之相适应的操作系统研究开发还处于起步阶段,从而阻碍了HMP的应用推广。本文在研究HMP的共同特点的基础上,针对操作系统设计中的操作系统构架设计部分进行讨论,提出一种多主模式的异构多核嵌入式操作系统构架,以解决HMP编程效率低、处理器利用率低及操

到稿日期:2010-08-29 返修日期:2010-12-13 本文受核高基重大专项(2009ZX01038-002-002),重庆市科技攻关计划项目(CSTC, 2009AB2244),重庆市教委科学技术研究项目(KJ090526)资助。
蒋建春(1975—),男,博士生,讲师,主要研究方向为嵌入式系统、智能控制,E-mail:dennis_jjc@yahoo.com.cn;汪同庆(1949—),男,教授,博士生导师,主要研究方向为模式识别、智能控制。

作系统开发难度大等问题。该构架充分考虑 HMP 的构架通用性，设计了一种可配置的多主模式异构多核嵌入式操作系统 MHMOS(Multi-master Heterogeneous Multi-core Operating System)。

本文第 2 节介绍当前异构多核操作系统的相关研究进展；第 3 节主要介绍和分析 HMP 的共同特点和发展趋势，为操作系统构架设计提供硬件支撑条件；第 4 节对操作系统构架设计提供可行性分析设计，分别对操作系统中的构架、调度机制、核间通信等几个关键问题进行了设计分析；第 5 节主要进行多主模式和主从式构架的性能对比分析；最后对 HMP 操作系统研究进行了总结与展望。

2 相关工作

针对 HMP 的操作系统相关的研究主要有操作系统构架设计、任务划分、任务分配及调度等方面的研究。在操作系统构架设计方面的研究不多，主要有两种：一种是主从式构架结构，另一种是对称式构架结构。

主从式多核操作系统根据多核处理器的功能把内核分成主核和从核，主核和从核在功能和结构上都存在差异，主核结构和功能都较为复杂，负责全局任务、资源的管理和调度，由于从核在结构和功能上较简单，主要接受主核的管理，负责运行主核分配的任务执行，并具有本地任务调度与管理功能，见文献[1-4]。而对称式构架是采用相同的结构来设计操作系统，每个处理器核中运行一个相同构架的操作系统内核，特别是在内核任务通信与同步、共享资源管理等部分存在相同的结构，每个内核功能可以根据处理器核的功能进行裁减，见文献[5,6]。这种结构的操作系统由于结构较单一，便于系统设计实现，但不能很好地解决异构多核处理器中同构核之间的任务和共享资源的调度管理问题。

文献[4]在单核处理器操作系统的基础上提出了一种主从式操作系统构架，称为 APRIX (Asymmetric Parallel RealtIme KernelS)。在不同的处理器中采用不同结构和功能的操作系统内核。它的主要特点是根据处理器的功能特点采用 AMP 技术将操作系统也分成主从式结构。在主操作系统端，主操作系统的设备管理层上增加一个全局调度器和内核服务处理程序，分别用于负责全局调度和内核服务处理，同时为从操作系统提供任务分配和内核服务，而从操作系统仅保留一小部分功能负责任务管理和内核服务。从内核也增加两个附加模块：本地调度器和内核服务代理，用来负责处理主内核的命令和任务调度。整个系统的管理、任务调度，由于从核中只有局部调度器，需要向主核请求处理，任务的激活与运行需通过主核来进行调控，由主核决定任务的分配。这种结构的操作系统能够很好地利用各个异构核的特性进行工作，但是容易造成从核在等待主核指令的过程中时间的浪费，降低了任务调度的实时性；同时这种结构随着处理器核的增加，要求主核管理的资源、任务调度、事件处理、通信管理等对象也增多，主核负载的增加，将导致主核不能实时有效处理这些事件，成为性能提高的瓶颈。

文献[5]提出了一种对称多核嵌入式操作系统构架，在异构多核处理器中，不同的内核运行一个对称结构的嵌入式微内核拷贝。这个微内核由 4 个部分组成，内部通信模块(IPC-Iner-Processor Communication)、核间接口模块(IPI-Inter-Processor Interface)、硬件中断分配模块和调度模块。IPC 主要提供核间信息交互时收/发功能，IPI 主要提供不同处理器核对贡献存储器的管理与通信功能，硬件中断分配模块主要负责底层硬件产生的中断管理及其分配。调度模块主要根据优先级策略对微内核中的任务进行管理和调度。该操作系统构架很好地处理了异构多核中的通信及其中断管理问题，对解决异构多核处理器的任务静态调度有一定的意义，但不能很好地解决异构多核处理器的动态调度和资源共享问题。

由于多处理器系统和多核处理器系统存在较大的差异，传统的多处理器分布式操作系统不适合嵌入式异构多核处理器。当前针对异构多核处理器的操作系统研究主要是在嵌入式系统领域，大多在微内核操作系统的基础上进行修改和改进，见文献[4,7,8]，没有系统地针对异构多核处理器的特点设计操作系统构架。

3 HMP 构架参考模型

对于 HMP 来说，主要存在两种不同的处理器构架，一种是完全异构的多核处理器，这种处理器的不同核在构架、时钟频率、指令等方面存在差异，在不同的处理器核上程序的运行方式、指令的多少、存储方式、运行时间等可能不同，如：TI 的 OMAP[9] 系列、MS320DM644x SoC[10] 等；另外一种是部分异构的多核处理器，这种处理器中存在多种构架的处理器，每种相同构架的处理器核有多个，一般主处理器核和从处理器核存在异构性，因此在这种多核处理器中只有局部是同构的，而总体上是异构的，如 IBM CELL[11]。如图 1 所示，第二种处理器包括了第一种的特点。随着微电子技术的发展，为了提高各个核的处理能力，多核处理器具有一些共同属性，处理器核除了共享相同的外设和存储空间外，同时还有自己的私有存储空间，如高速缓存、本地内存等[12]。需要进行大量计算时，临时数据一般保存在本地存储空间中，处理器核只需要将计算结果和需要与其它核进行交互的数据保存在共享存储器中，这样就可以减少对高速总线和共享存储访问的冲突，提高数据处理器能力，这种 HMP 是多核处理器的主要发展方向之一。

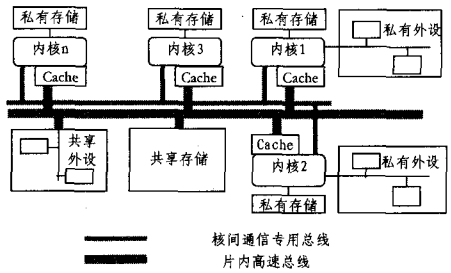


图 1 异构多核处理器基本构架

为了更好地利用各个处理器核和使其协调工作，未来的 HMP 除了采用共享内部高速总线访问共享设备外，一般还设计了用于核间消息传递的专用总线和外设[13,14]，各个处理器配置有核间通信接口模块用于通信，如图 1 所示。这种总线与高速总线结合，对于数据量较大的数据交互通过高速总线进行传递，然后通过核间通信总线发送消息通知相应的处理器核，同时可以采用中断机制，使消息快速响应，从而实现消息数据的实时交互。因此，本文研究的异构多核嵌入式操作系统构架主要是针对第二种异构多核处理器进行的。

4 多主模式操作系统构架模型设计

操作系统设计时，为了充分利用私有资源和共享资源，适应各种不同结构和功能的 HMP 以及嵌入式系统的实时行要求，必须合理设计一个结构简单灵活、功能可配置裁减的操作系统。与单核操作系统相比，在 HMP 异构核中，由于异构核间的指令和构架都可能存在不同，任务不能进行动态迁移，怎样实现资源共享和核间任务通信与同步是设计的重点；而在同构核之间，任务可以实现动态迁移，怎样进行任务有效的动态调度以及共享资源的访问控制是需要解决的关键问题。针对这些问题，本文充分考虑同构与异构核之间的关系，根据处理器核的不同结构与功能需求，设计一种基于统一构架的多主模式的异构多核处理器操作系统。

操作系统的调度和算法问题，在许多文献中已有讨论。本文主要从操作系统构架、调度机制与核间通信等方面进行设计，不涉及具体的调度算法。

4.1 操作系统构架设计

在上节讨论的第二类 HMP 内核中，一般可分为 MCU（Micro-Control Unit）和 MPU（Micro-Process Unit）两类，MCU 主要负责人机交互、文件系统、内存管理以及网络和通信等功能。而 MPU 一般负责编解码、信号处理等计算量较大的功能。在异构核中，如果采用对称式操作系统构架，在不同功能和结构的内核中运行一个相同结构的操作系统内核，这样不仅不能适应异构核的特点和要求，甚至由于额外的功能和不匹配的结构，造成内核运行效率低下，因此，需要根据处理器核的结构和功能特点设计一种可以配置和裁减的操作系统构架以满足特定应用的需要。

而异构多核中的同构核一般以微处理器为主，同构多处理的 OS 策略主要使用主/从模式（master/slave），如通常将一个 OS 放在某个处理核中作为 master，用来负责整个系统诸如中断处理、资源仲裁和调度等工作。主内核在任何时候都需要管理系统的所有共享资源的情况，然后分配和调度资源。使用主/从模式的对称多处理 RTOS 虽然实现起来最简单，但是，随着处理器核的增加，却不能提供较大的性能提升。主内核需要对所有管理从内核进行资源分配和调度，每次只能执行一个操作，这种顺序执行统一调度的方式效率低下，使得多核处理器的并行处理功能大大降低，因此 master 成为了系统瓶颈。

因此，针对这种情况，本文将操作系统内核分为单主内核和多主内核，采用统一构架，只是存在功能不同。每个处理器核运行一个操作系统内核，根据处理器核功能的需要配置内核，异构核中采用单主模式，在同构核中采用对称多主模式。在不同构架的核中的操作系统内核可以在功能上存在差异，以满足处理器核的需要，同构核中的每个内核可以作为主内核来管理资源和任务调度，以消除单主核模式中由于处理器核增多带来的主核负载瓶颈问题。

在单核操作系统中，内核负责任务管理、资源管理、中断管理、事件处理及通信管理等工作。在多核操作系统中，这些功能同样具有，除此之外，还包括核间通信与同步、核间任务调度管理、共享资源及设备管理等部分。异构主核除了一般的任务管理、设备管理、核间通信等功能以外，还具有其它与人机交互相关的功能，如文件系统、GUI、网络管理等功能，而同构核一般以计算和处理功能为主，只需要包含如任务管理与调度、设备管理、核间通信等一些基本功能。因此，本文采用组件化方法来设计操作系统构架，将功能相对独立的部分进行模块化，设计成可以配置的组件。

如图 2 所示，构架根据功能将操作系统分为 4 个层次：应用层（Application Layer）、操作系统内核（OS Kernel）、本地硬件抽象层（LHAL-Local Hardware Abstract Layer）和全局板级支持包（GBSP-Global board support package）。应用层是各种用户程序集合。操作系统内核是操作系统核心部分，提供应用层编程调用接口、任务管理、任务调度、设备管理、通信与同步、内存管理等功能，在内核中，为了便于本地设备和共享设备统一管理，通过虚拟设备管理模块将本地设备及共享设备、本地任务及全局任务等部分统一映射到本地进行访问，如内存映射、外设映射、I/O 映射、任务映射等，内核通过虚拟设备层来访问各种资源时就像访问本地资源一样，便于操作系统实现对全局和本地资源的统一。管理访问共享设备与本地设备、核间通信与核内通信及任务调度是虚拟设备层的主要功能。而 LHAL 主要是各个处理器核本地硬件及其外设的抽象层，可屏蔽本地通用硬件特性。异构核虽然存在结构上的不同，但可共享一些资源，如内存、外设等，因此，设计一个 GBSP 抽象通用共享设备、系统启动以及根据对应的内核配置信息将 OS 拷贝到各个处理器核中。

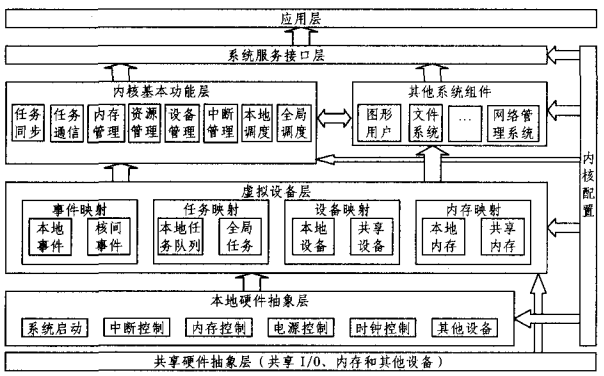


图 2 多主模式操作系统统一功能结构模型图

单主内核和多主内核采用相同的结构，只是在功能构成上存在差别。单主内核主要运行在异构核上，与多主内核相比结构相同但功能存在差异，除了基本功能以外，还包括人机交互、控制等功能，如文件系统、网络管理、GUI 等功能。由于单主内核和多主内核之间的处理器核构架不同，因此，它们之间的任务不能实现动态调度，所以单主内核需要在运行前事先分配和加载指定的任务，在本地进行任务调度。除此之外，当单主内核和多主内核任务进行同步时，还需要对多主内核中的任务进行调度管理，因此还需设计一个全局调度模块。如果单主内核需要与多主内核的任务进行通信和同步，可以通过通信与同步功能模块与虚拟设备管理共同实现，关于任务的同步与通信见 4.2 节。

这种操作系统模型采用统一结构的内核，通过功能配置为单主内核和多主内核，便于操作系统设计和应用编程。多主内核一般运行在异构多核处理器的同构核中，单主内核相比少了人机交互和控制等功能模块，主要负责同构核间的任务调度与管理、共享设备管理、共享内存管理以及核间通信与事件处理等功能。多主内核的每个核中都有这些相同的模

块，相当于将传统的一个操作系统内核简化后分布在各个同构核中实现，以实现内核功能的分布式处理，降低对某一个处理器核的依赖性，提高任务调度的实时性。

4.2 操作系统调度机制及核间通信

上节对操作系统的基本构架进行了设计，在构架中，单主内核内存在本地任务调度，多主内核之间存在核间任务调度，同时所有内核之间任务可能存在通信和同步，要实现这些功能还必须设计异构多核操作系统的调度策略和核间通信机制。

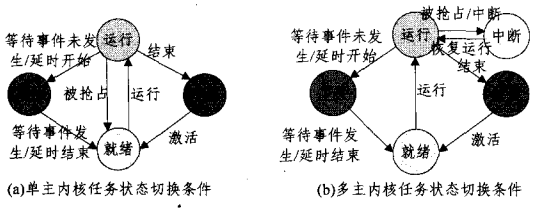
4.2.1 任务优先级

在本操作系统构架中，主要是针对实时系统而进行设计的，为了更好地适应异构多核处理器中的同构核的任务调度，每个任务都分配一个优先级，同时允许一个优先级对应多个任务，形成优先级队列，对于同一优先级队列中的任务具有相同的优先级，在任务调度时，优先级高的任务可以优先于低优先级的任务被调度执行。

4.2.2 任务状态与调度机制

本文中，在多主内核中采用基于全局队列的任务调度算法，在单主内核中采用局部队列调度算法，整个系统采用抢占式与非抢占混合调度策略来进行任务调度，每个任务可以根据任务特性配置为抢占式调度或非抢占式调度。为了便于统一管理核间任务通信及同步，需要对所有任务进行编号，并建立任务/事件映射表和任务/内核映射表存放在共享存储中。在核间通信时，通过这些映射表找到需要通信的任务及其执行该任务的内核，详见下节。

为了实现多核调度的实时响应，本文将任务分为运行、就绪、挂起、等待和中断 5 个状态。对于单主内核的任务而言，任务只在本地存储空间运行，不能被动态调度到其他核中执行，因此，任务的状态和单核操作系统中的任务状态基本一致，任务设置为前 4 种状态，如图 3(a)所示。



(a)单主内核任务状态切换条件　　(b)多主内核任务状态切换条件

图 3　任务状态切换图

对于多主内核，任务代码在执行前存储在共享存储空间中，各个内核根据任务的状态、优先级和各个多主内核的负载平衡情况进行动态调度执行。为了降低对共享存储空间的访问频率以提高任务调度和执行效率，任务设置为 5 种状态，如图 3(b)所示。任务在执行过程中，可能被中断或被其它高优先级任务所抢占，被抢占的任务还未执行完成，如果此时将该任务在共享存储中的状态改变为就绪，需要将任务的相关数据及状态信息写入共享存储的对应位置，与将数据保存在本地的存储空间相比，时间花费更多，并且可能引起共享存储的访问冲突。因此，在本文中，将任务被中断或高优先级任务抢占的状态定义为中断状态。当该任务被中断或抢占时，对应的现场数据被压栈到本地存储器中，不改变共享存储中的任务状态标志，此时对于其它内核而言，任务还处于执行状态，不需要对该任务重新调度，任务暂时保留在内核本地存储空间中，当中断服务例程或高优先级任务执行完成后，从本地存储空间中快速恢复执行。如果任务由于延时或等待事件而挂起，需要进入全局挂起状态。

在内核中，任务的调度和单核操作系统一样，只是在内核中增加了一个虚拟设备层模块，用于区分核间和核内的资源管理、消息传递和任务调度等功能。当涉及核间任务同步时，如果与之通信的任务还处于非运行状态，即此时还没有运行该任务的内核，需要激活该任务并根据调度算法向某个多主内核发送调度命令，因此需要对多主内核间通信。

在多主内核中，所有运行任务的状态都保存在共享存储空间中，任务的调度都是根据共享内存中的任务状态以及内核的状态与内核的负载来进行的。为了负载平衡，在每个内核中都有一个任务调度算法来计算和调度任务到相应的内核上去。为了解决多个任务在一个多主内核中的被抢占问题，采用 FILO(First In Last out)队列来进行管理本地任务调度。

在本文中为了提高内核的效率，将调度过程分为主动调度和被动调度。主动调度，就是内核在检测到高优先级任务根据调度算法抢占正在执行的任务或者内核在空闲时主动调度就绪任务执行的过程。在这种调度过程中，调度指令是内核本身发出的，当内核处于空闲状态时，可以设定一个扫描周期对就绪队列进行检测，一旦存在就绪任务，就按照调度算法进行任务调度。被动调度是指内核在任务执行过程中或者当内核空闲时，接收到其他内核根据调度算法发送任务调度命令消息后，内核调度器调用就绪任务执行的过程。在这种情况中，任务的调度命令是其他内核发出的。

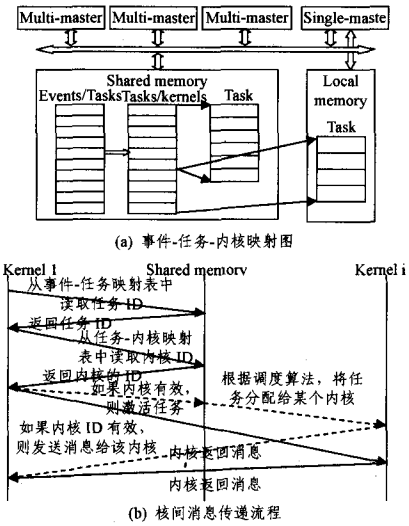
由于在多主内核管理的任务中，可能存在大量任务需要管理，如果每个内核都对所有任务进行管理，不仅不会提高内核的执行效率，反而会因为扫描大量任务花费更多的时间，从而降低执行效率。为了解决这个问题，本文将任务按照内核的数量进行分段，让不同内核分别管理不同任务共同具有的操作，如任务的延时管理，这样可以降低延时的处理时间，提高系统实时性。

4.2.3 任务的通信与同步

要实现操作系统的正常工作，多核操作系统除了必须具备单核操作系统的基本功能以外，还必须具备内核之间的通信与同步、核间调度功能。根据上节设计的操作系统框架，核间通信与同步是通过通信事件管理模块来实现的(COM & EVENT)，将消息、信号量、事件等用于任务间同步和通信的方式都看成事件进行统一管理。实现单主内核与多主内核、多主内核相互之间的任务同步于通信，主要通过操作系统内核中的虚拟设备层进行本地和核间事件映射，同时需要设计全局任务状态表、事件/任务映射表、任务/内核映射表来实现事件与任务的关联、任务与处理器核的关联，以达到任务之间的通信与同步的目的，如图 4(a)所示。全局任务状态表用于记录全局任务的状态(运行、就绪、挂起、等待)，只有处于运行态和在单主内核中的任务才有对应的内核编号。事件/任务映射表用于建立核间事件与任务之间的联系，在任务创建时需要将新创建的任务和事件进行关联，便于进行任务通信与同步。任务/内核映射表用来记录任务与运行该任务的内核编号的关联关系，在任务被调度时由执行调度的内核调度器负责进行修改。

内核间的任务在进行通信和同步时，发起通信的内核可以通过事件根据事件/任务映射表找到对应的任务，如图 4(b)所示，然后通过任务/内核映射表可以找到执行任务的内核。如果对应该任务的内核无效，则表明该任务处于非运行

态，这时，发送消息或事件的内核调度器负责激活该任务为就绪态，按照优先级方式和负载平衡要求，根据相应算法查找低于该任务优先级的最低优先级正在执行的任务的内核，通过发送消息或事件给该内核，抢占该任务执行，被抢占任务进入中断状态。



(a) 事件-任务-内核映射图

(b) 核间消息传递流程

图 4　核间同步与通信过程

同样，对于单主内核而言，如果需要向多主内核发送数据或消息，其执行步骤和同构核之间的通信一样。

5　性能分析

这种多主模式操作系统采用对称结构来设计操作系统构架，它们的主要特点是各个处理器核可以运行相同构架的操作系统内核，但可以根据处理器本身的特点进行功能裁减，实现对资源、任务调度的功能管理。这种构架降低了操作系统实现的难度，提高了编程效率和操作系统的灵活性。操作系统不同内核在系统启动时，启动代码根据处理器核的操作系统配置表选择性地将操作系统内核的不同功能拷贝到不同核中运行。

5.1　调度步骤对比分析

为了更好地说明本操作系统构架的优越性，本文采用文献[4]的主从式结构进行对比分析。在文献[4]中，所有从核的任务都是由主核负责管理和调度，从核通过核间通信消息来进行任务的调度管理。从核的任务调度过程经历以下几个，步骤如图 5(a)所示：

(1)某个从核完成一个任务，从工作状态进入空闲状态，通过核间通信模块向主核发送一个任务完成消息；

(2)主核收到消息并记录该从核的状态；

(3)某个从核在执行任务的过程中需要激活一个任务，向主核发送激活任务命令；

(4)主核接收到命令后，设置该任务为就绪态；

(5)主核查找空闲从核；

(6)主核发送调度命令消息给空闲从核；

(7)空闲从核接收到调度命令后调度或创建任务执行。

如果采用本文的多主模式来进行任务调度，则按照上节设计的两种调度方式进行分析，一种是主动调度，一种是被动调度。

在主动调度中，任务的调度过程只需经历以下几个步骤，如图 5(b)所示：

(1)某个内核完成一个任务，从工作状态进入空闲状态时，先查找就绪队列是否有任务等待执行，如果有任务，则直接调度任务执行，并将该任务状态设置为执行；

(2)设置内核状态为工作状态。

在被动调度中，任务调度过程经历以下几个步骤，如图 5(c)所示：

(1)某个内核完成一个任务，从工作状态进入空闲状态，内核设备管理器设置内核状态为空闲；

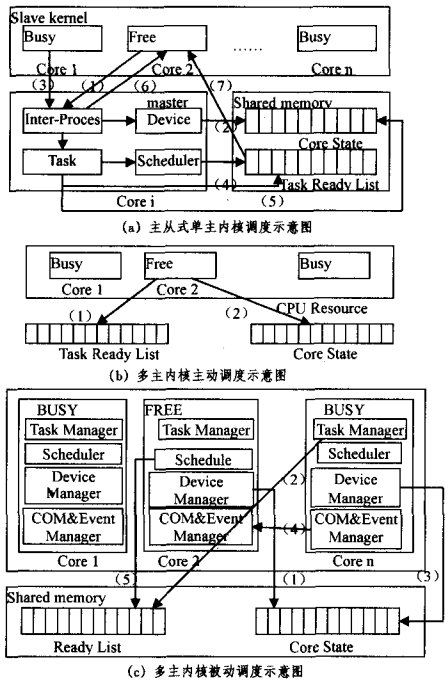
(2)当某个内核执行任务过程中激活了一个任务，设置该任务为就绪态；

(3)该内核调度器查找空闲内核；

(4)向空闲内核发送一个任务调度消息；

(5)空闲内核接收到消息后，从就绪队列中调度该任务，并设置任务的状态为运行态。

从上面的分析可以看出，采用本文设计的操作系统构架比文献[4]中的构架在执行单任务调度时步骤更少，在主动调度中，任务调度工作效率远高于文献中[4]的内核。同时，由于主从式内核中的主核负责所有的从核的任务管理与调度工作，如果同时存在多个任务需要调度，主核只能依次顺序进行调度，这样势必造成主核成为提高性能的瓶颈。而对于本文提出多主内核，可以通过细粒度同步[15]方法来对需要访问的队列和链表加锁，实现共享数据结构不同单元的并行访问，从而提高调度效率。



(a) 主从式单主内核调度示意图

(b) 多主内核主动调度示意图

(c) 多主内核被动调度示意图

图 5　主从式内核和多主模式内核任务调度步骤对比

由于操作系统的性能，如任务调度、消息处理、数据读写操作等，除了与操作系统构架相关，还与调度算法、消息处理机制、读写控制操作等部分相关，因此本文主要从操作系统构架方面进行设计，但未对这些性能做出比较。

5.2　调度时间测试

为了验证这种构架的合理性，本文以 ucos-ii 操作系统为原型，在该操作系统基础上，添加虚拟设备层和修改内核全局调度器功能。实验平台以 Xilinx 的 Virtex-5 FPGA 为基础搭建一个由 PowerPC 405 核和 4 个 MicroBlaze 4.00 软处理器核组成的异构多核 SOC 系统平台，在该平台上分别对同构核

之间的主动调度、被动调度时间、异构核与同构核之间的通信时间进行了测试。在测试过程中，FPGA 的工作频率设置为200MHz，内核间的消息传递采用带中断的消息邮箱机制。表 1 是多主内核任务调度的花费时间，在主动调度过程中，访问共享数据时等待的时间是影响调度时间波动的主要原因。在被动调度中，除了访问共享数据的时间波动外，调度时间还与任务是否是抢占调度有关，在抢占调度中，压栈当前任务状态与数据需要花费时间，因此比空闲时调度需要更多时间。

表 1　多主内核任务调度时间

调度类型	平均时间	最大时间	最小时间
主动调度	0.35us	0.35us	0.35us
被动调度(内核忙时)	0.35us	0.35us	0.35us
被动调度(内核空闲时)	0.35us	0.35us	0.35us

表 2 是核间任务通信所花费的时间，这些时间主要与核间通信接口的中断等待时间、内核是否空闲以及被中断任务现场保护等因素有关。

表 2　核间任务通信时间

通信类型	平均时间	最大时间	最小时间
异构核与同构核	0.35us	0.35us	0.35us
同构核与同构核	0.35us	0.35us	0.35us

结束语　这种操作系统构架由于具有可配置的统一结构，使得在实现上大大降低了难度，同时由于多主模式使共享任务及其设备的调度管理效率更高，系统内核实时性更强。但是这种构架带来的不足是需要每个处理器内核有较大的本地存储空间以存储操作系统内核代码及数据，但随着微电子技术的发展，为了满足系统的整体性能要求，增大存储空间应该不是问题。在多核系统中，任务的划分、分配及调度问题也是需要解决的关键问题，这些在同构多核系统中已经进行了大量的研究，对于异构多核处理器而言还需要进行大量工作。

参 考 文 献

[1] Goble G H, Marsh M H. A Dual Processor VAX 11/780[C]// Proceedings of the 9th Annual Symposium on Computer Architecture. 1982:291-298

[2] Kagstrom S, Lundberg L, Grahn H. A Novel Method for Adding Multiprocessor Support to a Large and Complex Uniprocessor Kernel[C]// Proceedings of 18th International Parallel and Distributed Processing Symposium. 2004

[3] Muir S, Smith J. AsyMOS-an Asymmetric Multiprocessor Operating System[C]// Proceedings of Open Architectures and Network Programming. 1998:25-34

[4] Seo M, Kim H S, Maeng J C, et al. An Effective Design of Master-Slave Operating System Architecture for Multiprocessor Embedded Systems[C]// Proceedings of Lecture Notes in Computer Science, 12th Asia-Pacific Conference. Seoul, Korea, 2007: 114-125

[5] Chen Jing, Liu Jian-hong. Developing Embedded Kernel for System-On-a-Chip Platform of Heterogeneous Multiprocessor Architecture[C]// rtcsa, 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06). 2006:246-250

[6] 蒋建春，曾素华，岑明. 一种基于异构双核处理器的嵌入式操作系统构架设计[J]. 计算机应用，2008，28(10)：2686-2689

[7] 谢铖，多内核构件化嵌入式操作系统的研究[D]. 杭州：浙江大学，2006

[8] 章承科. 基于多核处理器的实时操作系统的扩展[D]. 成都：电子科技大学，2006

[9] Intel. Intel@CoreTM2 Duo processor [EB/OL]. http://www.intel.com /products/ processor_number/eng/chart/core2duo.htm.

[10] 达芬奇 (DaVinci™) 数字媒体处理器[EB/OL]. http://focus.ti.com.cn/cn/paramsearch/docs/parametricsearch.tsp? family=dsp&sectionId=2&tabId=1852&familyId=1300

[11] Kahle J A, et al. Introduction to the cell multiprocessor [J]. Ibm Journal of Research and Development, 2005, 49(4/5): 589-604

[12] 邓让钰，陈海燕，等. 一种异构多核处理器的并行流存储结构[J]. 电子学报，2009，37(2)：312-317

[13] 迎九，多核/虚拟化/多操作系统的软件趋势[J]电子产品世界，2010，17(1)：58-60

[14] Guérin X, Pétrot F. A System Framework for the Design of Embedded Software Targeting Heterogeneous Multi-core SoCs [C]// asap, 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors. 2009:153-160

[15] Herlihy M, Shavit N. 多处理器编程的艺术[M]. 金海，胡侃，译. 北京：机械工业出版社，2009：141-145

(上接第 285 页)

[3] 王蕴红，谭铁牛，朱勇. 基于奇异值分解和数据融合的脸像鉴别[J]. 计算机学报，2000，23(6)：649-653

[4] 甘俊英，张有为. 一种基于奇异值特征的神经网络人脸识别新途径 [J]. 电子学报，2004，32(1)：170-173

[5] Liu Ke, Cheg Yong-ning, Yang Jing-yu. Algebraic feature extraction for image recognition based on an optimal discriminant criterion [J]. Pattern Recognition, 1993, 26(6): 903-911

[6] Swiniarski R W, Hargis L. Rough sets as a front end of neural-networks texture classifiers [J]. Neurocomputing, 2001, 36(1-4): 85-102

[7] Swiniarski R W, Skowron A. Rough set methods in feature selection and recognition [J]. Pattern Recognition Letters, 2003, 24(6): 833-849

[8] 王敏，王彦杰. 基于小波变换和改进的奇异值分解的人脸识别[J]. 计算技术与自动化，2008，27(4)：91-94

[9] 谢永华，陈伏兵，张生亮，等. 基于分块小波变换与奇异值阈值压缩的人脸特征提取与识别算法[J]. 计算机应用与软件，2008，25(1)：30-32，78

[10] 闫荣华，彭进业，李岩，等. 基于小波域奇异值分解的人脸识别方法 [J]. 计算机工程，2007，33(4)：212-214，217

[11] 蒋巍，王伟. 基于小波和奇异值分解的人脸识别方法 [J]. 计算机仿真，2006，23(4)：181-183

[12] 王敏，尹政兴，李汉强. 基于小波变换和奇异值的人脸特征提取[J]. 中国水运：理论版，2006，43(3)：163-163

[13] Tian Yuan, Tan Tie-niu, Wang Yun-hong, et al. Do singular values contain adequate information for face recognition? [J]. Pattern Recognition, 2003, 36(3): 649-655

[14] Liu Jun, Chen Song-can, Tan Xiao-yang. Fractional order singular value decomposition representation for face recognition [J]. Pattern Recognition, 2008, 41(1): 378-395

[15] Lu Ji-wen, Zhao Yong-wei. Dominant singular value decomposition representation for face recognition [J]. Signal Processing, 2010, 90(6): 2087-2093