

嵌入式非对称多核并行软件设计

李志远, 赵元富, 兰利东

(北京微电子技术研究所, 北京 100076)

摘 要: 多核处理器结构已经从通用计算领域延伸至嵌入式计算领域, 并成为嵌入式处理器主流发展形式. 为了解决传统串行软件无法有效利用嵌入式非对称多核处理器计算资源的问题, 提高嵌入式多核处理器的性能表现, 对非对称多核结构和相应的并行方式进行研究. 针对嵌入式非对称多核处理器的特殊结构, 提出了组件化设计的混合并行软件, 建立了非对称多核处理器的并行执行环境, 可以充分利用系统的计算资源, 提升系统计算性能.

关键词: 非对称多核处理器; 并行软件; 嵌入式系统; 组件化软件

中图分类号: TP338

文献标识码: A

文章编号: 1000-7180(2013)08-0107-05

Parallel Software for Embedded Asymmetric Multi-core Processor

LI Zhi-yuan, ZHAO Yuan-fu, LAN Li-dong

(Beijing Microelectronics Technology Institute, Beijing 100076, China)

Abstract: Multi-core processors have emerged as the major trend in computing systems and become a hot area of embedded system. A parallel software model for embedded multi-core processor is proposed to solve the problem that traditional serial software cannot meet the requirements of multi-core processor very well. According to the researches on architecture of embedded asymmetric multi-core processors and parallel software models, a hybrid parallel component-based software model is presented to make full use of computing resources of the embedded asymmetric multi-core processor to gain higher performance.

Key words: asymmetric multi-core processor; parallel program; embedded systems; component-based software

1 引言

进入 21 世纪以来, 随着集成电路工艺水平和集成电路设计方法学不断进步, 多核处理器凭借其高性能、低功耗的特点, 逐渐取代传统单核处理器成为当今处理器发展的主流形式, 并且从通用计算领域延伸至嵌入式计算领域^[1]. 越来越多的嵌入式多核处理器在业内出现并获得广泛应用. 嵌入式软件设计必须和硬件设计很好地匹配, 当硬件结构出现改变时, 软件设计必须做出相应改变, 针对目标硬件进行有的地改进和优化, 甚至是提出全新的软件模型, 才能有效利用硬件计算资源, 实现计算能力的最大化. 多核硬件结构的出现, 掀起了软件并行化革命浪潮. 多核处理器硬件结构的多样性使得并行软件的研究成为当今的热点和难点.

本文通过对新型嵌入式非对称多核处理器结构的研究, 提取系统并行性并设计了基于该非对称多核处理器的并行软件. 该并行软件可以有效提升整体系统的计算性能, 丰富系统功能, 提升整体效率及系统灵活性.

2 多核结构和并行模型

自从摩尔定律提出以来, 微处理器的发展一直遵循着摩尔定律, 运算速度得到极大提升. 进入 21 世纪之后, 功耗问题和散热问题成为传统处理器性能提升的极大阻碍^[2]. 微处理器设计者的注意力开始向多核处理器转移. 相比于传统的单核处理器, 多核处理器充分利用芯片内晶体管, 在提升处理器性能之余降低了功耗, 这种结构已经成为处理器发展的主流趋势^[3]. 业内厂商已经推出多款多核处理器,

收稿日期: 2012-10-23; 修回日期: 2012-12-10

如: Intel 的 Core 系列, IBM 的 PowerPC 系列, Cell BE 系列和 nVidia 的 Tegra 系列等多核处理器。

2.1 多核结构

由于设计思想和硬件结构的不同, 嵌入式多核处理器结构呈现多种多样的形式。根据核心之间的逻辑关系划分, 多核处理器可分为对称多核和非对称多核。对称多核 (Symmetric Multi-core Processor, SMP) 指处理器内部各个核心体系结构相同, 性能参数相同, 核心地位相同, 访问外设权限相同。非对称多核 (Asymmetric Multi-core Processor, AMP) 中, 各个核心可以相同也可以不同, 一般由一个功能强大的核心作为主处理核心, 其余核心作为辅助处理核心^[4]。对称多核和非对称多核的典型结构如图 1 所示。

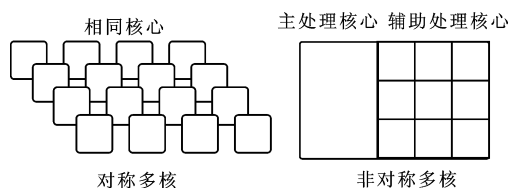


图 1 多核结构示意图

典型的嵌入式非对称多核结构中各个核心具有相同的 ISA (Instruction Set Architecture), 但是性能、尺寸、速度和功耗各不相同^[5]。非对称多核处理器可以显著提高系统性能, 降低系统功耗, 具有良好的系统灵活性和巨大的性能潜力, 可以突破制约对称多核处理器性能的阿姆达尔定律, 提高系统执行效率, 非对称多核结构已经成为多核研究的热点领域^[6]。

2.2 并行软件模型

多核结构给软件开发者提出了新的挑战, 软件模型必须和硬件设计紧密结合, 才能发挥多核处理器的硬件结构优势, 提高系统的性能表现。业内对于多核并行软件设计进行了大量的研究, 提出了多种并行软件模型, 其中以共享存储编程模型和消息传递编程模型为主流编程模型。

(1) 共享存储编程模型

共享内存编程模型 (Shared-memory Model) 适合对称多核结构, 多个硬件核心对于全部资源有相同的访问权限, 所有内存空间对于各个核心来讲都是透明的^[7]。共享存储编程模型以 OpenMP 编程模型为工业标准。OpenMP 编程优点是编程简单, 可以充分利用共享存储体系结构的特点, 提供细粒度和粗粒度的并行机制。缺点是它的可扩展性比较差, 执行效率受系统可用计算单元的限制^[8]。

(2) 消息传递编程模型

消息传递编程模型 (Message Passing Model) 对应的硬件模型中, 多个核心具有自己的私有内存, 核心之间通过邮箱等机制进行消息传递。基于消息传递编程模型设计的软件适合运行在分布式非对称多核处理器上。MPI 是消息传递编程模型的实现标准, 是目前分布式存储系统上的主流并行编程模型。MPI 编程模型的执行效率高, 可移植性强; 缺点是基于 MPI 编程模型的应用程序开发和调试相对困难, 核心之间的通信开销较大, 只能提供粗粒度的并行机制。

针对多种多样的多核硬件模型, 人们还研究设计了其他多种并行编程模型, 以期充分利用多核硬件结构的计算能力。

3 硬件结构

多核片上系统 (Multi-Processor System on Chip, MPSoC) 以其高性能、低功耗、高集成度和高可靠性的特点很好地满足了嵌入式系统的要求, 从而成为嵌入式处理器设计的重点研究内容^[9]。本文的实验平台基于一款代号为 Virgo 的嵌入式 MP-SoC 处理器芯片。

3.1 硬件结构

Virgo 整体硬件结构如图 2 所示。使用一个高性能 32 位 RISC 处理核心作为主处理核心, 采用哈佛结构设计, 具有两级缓存。一个性能稍低的 32 位 RISC 处理核心作为辅助处理核心, 同样采用哈佛结构, 具有一级缓存; 内部还具有众多控制器和加速器, 可以实现多种外围器件的扩展和不同应用的加速计算。

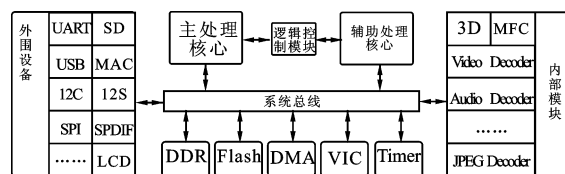


图 2 Virgo 硬件结构

3.2 通信方式

主核在整体结构中占据主要地位, 对所有外设具有控制和访问权限。辅核位于次要地位, 对外设的控制和访问权限受主核控制, 控制机制通过内部逻辑控制模块来实现。逻辑控制模块为辅核添加逻辑控制, 使得辅核的权限受主核限制。主核可以访问全部内存空间, 辅核只能访问有限的内存空间, 这就使得我们可以利用部分内存实现大数据量的指令和数

据的交互。

逻辑控制模块的另一个功能是实现主核与辅核之间的消息传递机制。该模块中存在小规模寄存器堆,可以实现小数据量的交互。另外主核和辅核可以通过配置逻辑控制模块产生对另一个核心的中断信号来实现对另一核心工作状态的控制。通过这种方式实现双核心之间的消息传递机制。

4 多核结构和并行模型

嵌入式系统计算资源有限,硬件相对简单,但软件的复杂度较高,嵌入式系统对于高性能、低功耗、高集成度和高可靠性的要求,使得嵌入式系统的整体设计较为复杂。由于嵌入式多核系统的特殊性,嵌入式软件和嵌入式硬件必须紧密结合才能发挥整体系统的性能。

Virgo 的设计采用了新颖的设计方式,模糊了传统的共享内存结构和分布式结构的分界线。软件设计方面必须针对硬件的设计做出优化和改变,才能有效的利用硬件计算资源。

4.1 混合并行模型

根据 Virgo 的硬件设计,对该系统的并行性进行深入的研究,提出基于该处理器的混合并行模型,模型如图3所示。

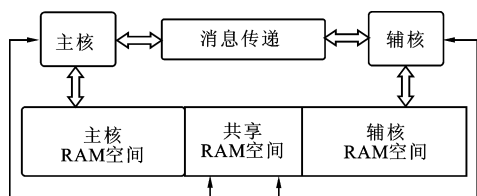


图3 混合并行模型

在混合并行模型中,主核对于全部内存空间具有绝对的访问权限,而辅核只具有有限的访问权限。在软件设计中设计了一块可以被主核和辅核共同访问的内存空间作为共享空间,以实现双核心之间大数据量的指令和数据传递。另外使用处理器内部的逻辑控制模块实现双核心之间的消息传递机制。

4.2 工作模式

Virgo 中两个核心都是完整而独立的核心,可以独立执行任务也可以协同执行任务。根据此特点,为 Virgo 设计了两种工作模式:分立模式和协同模式。

(1)分立模式

分立模式下,双核心的工作方式与多计算机系统类似,每个核心独立运行任务,不影响彼此的工作状态。我们强化了共享内存机制,主要使用该部分内

存空间进行大数据量的指令和数据交互;弱化了双核心之间的消息传递机制,屏蔽了双核心之间的中断通路,只保留双核心之间相关寄存器存储辅核的工作状态。主核根据寄存器中的信息判断辅核工作状态,当辅核工作状态出现问题时,主核及时获取消息,摒弃错误数据并重启辅核,以保证运算结果的正确性。

(2)协同模式

在协同模式下,共享内存机制和消息传递机制都激活并协同工作,主核分配任务给辅核,辅核接受任务并计算,计算完成后发送消息给主核,主核读取处理完成后的数据并进行数据的整合和处理。两个核心协同并行处理,利用辅核的计算能力,提升系统计算性能,利用辅核的通用特性,提升系统整体灵活性。

4.3 协同模式下的并行机制

Virgo 中的辅核,是完整而独立的处理核心,具有完整的指令集、独立的工作方式和私有内存空间,理论上只要使用相应的指令集编程,就可以有效使用辅核工作。

协同模式下,我们将主核上可并行化的任务划分为两个子任务,一个在主核上运行,一个发送给辅核,由辅核进行协同执行,计算结束后由主核进行数据的整合和处理,采用这种机制,实现任务的并行化处理。

4.3.1 单任务模式

辅核的通用特性,决定了辅核可以执行不同类型的任务,以达到提升系统灵活性的目的。执行任务类型不同,任务的数据量也不同,针对这种情况,为系统设计了两种任务模式:小任务模式和大任务模式。

(1)小任务模式

小任务模式下,系统主要处理数据量较小的任务,通过 DMA 操作,将任务加载到辅核的私有空间上,将数据加载到共享空间,启动辅核执行任务。当辅核执行结束后,数据被存储在共享空间内供主核读取并进行后续处理。这种情况下,辅核上任务数据量较小,由此带来的通信消耗也较小,辅核的运行效率较高,可以实现较好的并行效果。

(2)大任务模式

多媒体等任务往往数据量比较大,可能大于共享内存的数据容量,为此设计了大任务模式。系统对大数据量任务进行多次分割,将任务分割为大小合适的子任务队列。队列中的子任务被依次加载和处

理,在计算完成后主核将计算完成后的子任务队列进行整合和处理.这种模式优点是任务的分割和整合可以通过软件自动完成,缺点是多子任务的多次加载,会引起额外的加载消耗和通信消耗.

4.3.2 多任务模式

辅核的通用特性决定了辅核上可以运行多种任务,当多个任务同时存在于辅核内存空间时,必须有相应的任务调度机制进行有效的任务切换和处理,考虑到辅核上任务类型的多种多样,我们设计了多种调度方式对辅核上的任务进行调度处理.

(1) 辅核自我调度

当存在多个任务时,辅核可以直接进行任务的调度和切换.辅核对于外围设备具有访问权限,在主核系统软件部分对辅核可以访问的外围设备和中断屏蔽方式进行设定,辅核在运行时,可以通过对不同设备的中断响应,进行任务的切换.此外,当辅核软件上存在任务管理服务(TMS)或者微型嵌入式操作系统(EOS)时,辅核具有自主处理任务的能力,任务的切换和调度将会更有效.这种方法的优点是辅核直接调度任务,可以有效扩展辅核功能,提高系统灵活性.缺点是会增加辅核上软件复杂度,需要移植 TMS 或 EOS,使得辅核上软件的编写和调试比较复杂.

(2) 主核调度

Virgo 结构中,辅核的工作状态受主核控制,可以利用主核对辅核上的任务进行调度.主核可以利用消息传递机制对辅核的工作状态和执行的任务进行控制,实现任务调度.

在这种模式下,主核需要对于辅核当前执行的任务具有准确地控制.当需要任务切换时,主核向辅核传递消息,辅核根据主核发送的消息切换任务,并且将当前的任务信息和辅核工作状态保存在寄存器中供主核读取,以保证辅核运行的正确性.这种模型的优点是辅核上的软件设计简单,可靠性高,无需移植 TMS 或 EOS,只需要接受主核发送的消息并进行任务切换.缺点是主核需要对辅核的状态有清晰地控制,并且需要时时读取辅核状态,这会引起额外的消耗.

在软件执行过程中,由于任务的多样性,单一一种调度方式无法满足所有任务调度的要求.在任务执行时,我们将两种调度方式进行整合,涉及外围设备任务采用中断触发自主调度方式进行调度,不涉及外围设备的任务使用主核调度方式进行调度.

4.4 软件支持

4.4.1 软件接口

在系统执行时,主核和辅核协同执行,辅核作为控制和处理核心存在.在系统配置时,将辅核包装成主核的外围设备,以配置外设的方式对辅核进行初始化配置,配置方式采用驱动的形式实现.通过对相应寄存器的配置,对辅核的时钟频率、内存空间范围和中断方式等进行必要的配置.将并行软件中配置部分以驱动形式实现,主要驱动接口如表 1 所示.

表 1 驱动接口

驱动接口	功能描述
initSPE(void)	配置辅核参数:时钟频率、内存地址、内存大小、中断屏蔽状态和执行状态等
enableSPE(void)	开启 Clock-gate,使能时钟信号和电源
removeSPE(void)	卸载辅核,即关闭时钟信号和电源信号
loaddata(int *daddr, int dlen)	加载数据,参数为待加载数据的首地址和长度
.....	
restartSPE(void)	重启辅核模块

4.4.2 并行程序加载

主核和辅核都是完整独立的通用计算核心,可以独立执行任务.将辅核的并行程序作为一个独立的软件组件,加载到系统中,既实现了并行程序的加载,又无需对主核上的可执行文件进行大的改动.主核源文件和辅核源文件被各自的编译器分别编译,并被整合为统一的可执行文件.当该可执行文件加载时,主核程序和辅核程序被分别加载到各自的内存空间上去.当辅核并行固件加载结束并完成对辅核的最终配置时,并行执行环境就绪,辅核可以根据主核发送的消息和数据完成指定的任务,实现并行处理,并行程序加载过程如图 4 所示.

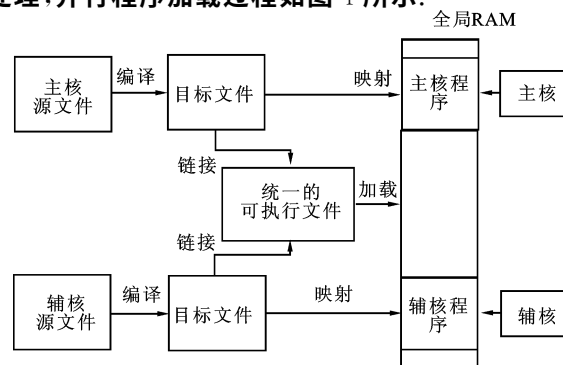


图 4 并行程序加载

4.5 竞争

共享的出现,往往伴随着竞争.有共享的机制,就需要有相应的处理竞争的解决方法.在并行软件执行过程中,我们需要处理的主要问题是数据竞争问题.

双核心之间大数据量的通信通过共享内存的方式实现,当某核心尚未完成对共享内存中数据的操作时,另一核心对该部分数据进行操作会引发数据竞争,导致计算数据出错.这就需要有相应的解决方法处理此种情况.在并行软件中,我们使用条件变量的方式处理数据竞争.系统运行时使用条件变量来标记当前数据状态,当主核正在操作某部分数据时,该部分数据相应的条件变量标记为当前不可用,使辅核无法对当前部分数据进行操作.只有当主核操作结束后,释放该部分资源,辅核才可以操作该部分数据.辅核处理数据时,同样使用此种方式进行操作,通过此种方式,有效防止数据竞争的出现.采用条件变量的方式具有通信消耗低、运行方式简单可靠的优点.

5 实验结果

针对 Virgo 的结构设计,我们将快速排序(QuickSort)这一经典算法进行并行化处理,改写成为并行快速排序算法对并行化软件进行性能评估.

快速排序是常用的经典算法之一,是基于分治策略的递归排序算法,可以改写成并行执行的形式.当具有多个处理核心时,将待排序的子序列分配至多个处理核心上并行进行即可.

图5中显示使用并行快速排序下程序的性能测试.根据图中数据,我们可以得到约1.35的加速比.出于稳定性考虑,辅核的运行频率被配置为主核运行频率的一半,考虑到通信消耗等因素,1.35的并行加速比表明,并行软件有效运行且提升了该系统的整体性能.

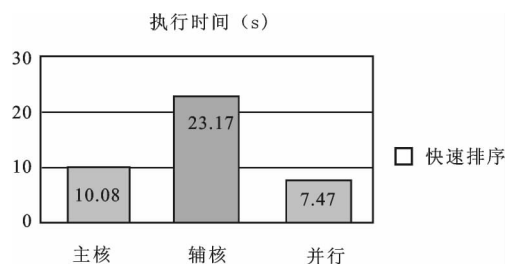


图5 执行时间比较

6 结束语

通过对 Virgo 结构和并行软件设计的研究和理

解,提出了基于嵌入式非对称多核处理器的并行软件结构.通过组件化的设计将并行软件和 Virgo 多核硬件平台紧密结合.通过并行快速排序算法评估并行软件结构,获得约1.35的并行加速比.未来的工作重点集中在软件结构的优化和提升并行加速比方面,进一步挖掘双核心的计算能力.

参考文献:

- [1] Agarwal V, Hrishikesh M, Keckler S, et al. Clock rate versus IPC: the end of the road for conventional microarchitectures[C]//Proceedings of the 27th International Symposium on Computer Architecture. Vancouver, Canada: IEEE, 2000: 248 - 259.
- [2] Hongtao Zhong, Steven A Lieberman, Scott A Mahlke. Extending multicore architectures to exploit hybrid parallelism in single-thread applications[C]//proceedings of the IEEE 13th International Symposium on High Performance Computer Architecture. Scottsdale, AZ: IEEE, 2007: 25 - 36.
- [3] David Geer. Chips maker turn to multicore processors[J]. Computer, 2005, 38(5): 11 - 13.
- [4] 谢向辉, 胡苏太, 李宏亮. 多核处理器及其对系统结构设计的影响[J]. 计算机科学与探索, 2008, 2(6): 642 - 647.
- [5] Yuanhao Xu, Lin Du, Zhimin Zhang. A light-weight scheduler for single-ISA asymmetric multi-core processor using online profiling[C]//proceedings of the 12th ASIC International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. Sydney: IEEE, 2011: 183 - 188.
- [6] 冯叶. 非对称多核体系下的阿姆达尔定律性能模型研究[D]. 上海: 上海交通大学, 2012.
- [7] 陈国良. 并行计算—结构·算法·编程[M]. 2版. 北京: 高等教育出版社, 2003.
- [8] Drosinos N Koziris. Performance comparison of pure MPI vs hybrid MPI-OpenMP parallelization models on SMP clusters[C]//proceedings of the 18th International Parallel and Distributed Processing Symposium. [S. L.], IEEE, 2004: 20 - 30.
- [9] Hiroyuki Tomiyama, Shinya Honda, Hiroaki Takada. Real-Time Operating Systems for Multicore Embedded Systems [C]//International SoC Design Conference. Busan, South Korea: IEEE, 2008: 62 - 67.

作者简介:

李志远 男, (1988—), 硕士研究生. 研究方向为嵌入式系统.