嵌入式 RISC 处理器体系结构并行技术的研究*)

周亦敏[1] 魏洪兴[2]

(上海理工大学计算机工程学院 上海 200093)[1] (北京航空航天大学机器人研究所 北京 100083)[2]

摘 要 本文通过对目前国内外主流嵌入式处理器体系结构创新与发展的研究,着重从处理器体系结构中 RISC 规则的突破、数据处理、多线程、多核处理器的构成等多种并行技术的应用,对提高系统运行效率和降低运行功耗,作了较为全面的分析,同时研究了这些并行机制的实现技术。研究表明,嵌入式处理器结构中并行技术的应用,是应对目前嵌入式应用高性能、低功耗挑战的有效方法。

关键词 嵌入式处理器,体系结构,RISC,并行技术

Research on Parallel Technologies of RISC Architecture

ZHOU Yi-Min[1] WEI Hong-Xing[2]

(College of Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093)[1]

(Institute of Robot Research, Beijing University of Aeronautics & Astronautics, Beijing 100083)[2]

Abstract Based on the research on the innovation and development of the current popular embedded processor architecture, this article focuse on the application of several parallel technologies such as the advancement of RISC architecture, data processing, multiple-threads and the structure of Multiprocessor, systematically analyzes how to improve the system effectiveness and decrease the power dissipation for RISC architecture and discussed the implementing methods of the above parallel mechanisms. The result of the research indicates that applicationof parallel technologies is an effective way to deal with the challenge of high performance and low power dissipation for current embedded system.

Keywords Embedded processor, Architecture, RISC, Parallel technology

采用多数量的寄存器组,精简指令条数,单一的 LOAD-STORE 内存读写架构及相对简单的流水线,这是几乎所有精简指令系统处理器(RISC)的核心技术。RISC 的这种高效、精简、低功耗的特点,使绝大部分嵌入式处理器均采用这一结构。嵌入式处理器往往是作为 SoC 中的嵌入式内核设计,与通用处理器相比,除了性能还必须充分考虑代码的高密度,低功耗和芯片的小面积等重要因素[1]。

随着嵌入式系统在运动图像、网络传输、大容量数据处理等复杂应用的拓展,其快速增长的性能需求、越来越高的集成度和时钟频率,导致系统功耗成倍增加。因此,嵌入式处理器体系结构和微体系结构设计的一个重要任务,就是解决怎样同时满足高性能和低功耗这对矛盾,这促使嵌入式处理器体系结构的进一步发展。而多层次的并行技术的创新和应用,则是这几年嵌入式处理器体系结构发展的最重要成果之一。

1 对传统 RISC 规则的改进

嵌入式系统在有限的资源环境下,为使整体运行效率更高,必须对传统的 RISC 规则做出改进。通常 RISC 规则包括:指令的执行周期固定,且为单周期;指令长度固定,指令格式和寻址种类少;只有取数/存数(LOAD-STORE)指令访问存储器,且对操作数据不能预加工等。高性能嵌入式处理器对 RISC 规则的突破,体现在:某些指令的执行周期可变;对现行操作数可进行预处理;指令可根据条件执行;指令长度不唯一,引入高密度的压缩指令子集;补充增强型的 DSP 指令等。RISC 结构的这些改进与增强,大大提高了指令运行的并行性,系统的整体性能得到了很大的提高。

1.1 指令执行周期可变

基于 LOAD-STOR 架构,RISC 处理器在处理数据前要把数据加载到某个通用寄存器中,按传统 RISC 指令单周期的限制,单独地加载或保存一个寄存器的效率并不高。一些嵌入式处理器增加了可以同时加载或保存多个寄存器的指令,即根据需要可对多个寄存器进行数据传输,尽管当操作的寄存器数目较多时,指令的执行多于一个周期,但在整体上效率有了大的提高。如一次能进行 12 个寄存器的进(出)栈指令,需 3 个周期,等效于单周期单寄存器的 12 条进(出)栈指令,代码压缩 12 倍,效率提高 4 倍。这类指令在实时操作系统(RTOS)中多任务的上下文交换中极为有效,保证了任务切换的实时性和切换时间的可确定性,降低了运行功耗,对一个应用复杂、任务调度频繁的系统,其效果尤为明显。

1.2 操作数的预处理

一些嵌入式处理器指令,通过内部桶形移位寄存器(inline barrel shifter),能对现行操作数进行移位、循环等预处理,在操作前对操作数(如:指针偏移量)进行乘 2 、乘 4 等多种预操作,这在内存存取,表格、堆栈等多种数据结构的操作中,体现了更高的效率,并使数据处理指令有了更多的灵活性,提高了代码的效率。

*)基金项目:国家 863 高技术研究发展计划资助(批准号:2005AA420070)、上海市教委 2005 科技发展基金(批准号:05EZ48)。周亦敏 副教授,主要研究方向为嵌入式系统,计算机系统结构,网络应用与智能设备等。魏洪兴 博士后,副教授,主要研究方向为嵌入式系统,智能机器人,机械电子自动化等。

1.3 指令的条件执行

在一些嵌入式处理器的指令设计中，除了常规的转移、条件转移指令外，每一条指令都是可条件执行的，即在指令助记符后增加相应的条件后缀，满足设置的条件，这条指令方可执行，与此相对应，指令执行后对状态存储器的影响也可人为设置。这种指令的执行方式与C、C++、JAVE等高级语言的句法更类似，更宜于编译器生成高效的可执行代码。实验证明，完成同一功能的程序，使用条件执行的指令能节省近25%的代码空间，这对系统存储空间有限、开发成本极为敏感的嵌入式应用而言，是一个非常诱人的指标。

1.4 低密度指令的引进

一些32位的嵌入式处理器，在保留常规指令长度为单一32位的前提下，引入了16位的压缩指令子集，一些功能简单，操作码、操作数表达较易的指令（如：单寄存器操作的指令）无须用32位来表示，仅用16位就能反映所有的操作信息。压缩指令子集的增加，能以小部分寄存器不能访问和与32位指令切换产生的功耗等轻微的性能损失，换来指令密度的大幅下降。一个最大限度合理应用32位标准指令和16位压缩指令的程序，其代码密度可减少35%左右。

1.5 增加型的DSP指令

传统上，不同特征的数据并行应用由不同的体系结构和专用部件执行。一些非哈佛结构的嵌入式处理器，为提高数字信号的处理能力，运算器中增加了DSP的运算指令[2]。如多字节的饱和算术指令，能避免整型运算溢出产生负的结果，保证返回整型的最大值而不会发生回绕。增加的DSP指令可实现灵活快速的16×16乘法和饱和运算，使面向DSP的函数能方便地移植到常规的嵌入式处理器中。DSP指令的引入，使得某些嵌入式的应用，无需哈佛结构DSP处理器的升级换代，也能实现诸如VoIP等的复杂信号处理需求，这使常规嵌入式处理器能以极小的代价在DSP应用领域得以拓展。

2 面向数据处理的并行技术

为了对视频处理器和2D/3D图像处理等提供更多的DSP处理能力，需在数据处理层增加并行能力，许多嵌入式处理器通过单指令流多数据流（SIMD）架构来实现这一目标。SIMD主要用于解决向量和阵列等比较规整的数据结构的计算问题。这种架构只有一个控制单元，每次只能执行一条指令，但这条指令可以同时对多个数据进行操作。SIMD结构的特点是对不同数据实体的同一操作通过几条数据通路并行执行，它增加额外的特殊数据通路，可以把标准的32位数据路径分割成4个8位，或者两个16位的数据路径，提高数据传输的并行性，用相对少的指令完成复杂的计算，同时只需很少的内存访问，特别适合嵌入式应用的场合。可以说，SIMD技术在计算效率和低功耗之间能获得很好的平衡。

比如为使MPEG和H.263视频压缩算法更有效，专门设计了差值绝对值累加和运算（SAD）指令，在运动估计运算中通过计算 $SAD=\sum | R(i,j)\times C(i,j) |$ 来比较二个像素块，判断是否相似，这个过程需多次用SAD运算[3]。复杂DSP处理需要多个类似SAD的低功耗高效率的运算指令。

为了拥有更强的运算能力，很多嵌入式处理器采用开放的总线结构，很容易进行多个协处理器的扩展，指令执行机构提供对协处理器指令的支持，一旦需要，可方便地在原有嵌入式系统中扩展诸如浮点运算、DSP运算器等协处理器，这使嵌入式应用功能的增强、系统的升级快捷而方便，能有效缩短开发周期。

3 线程层面的并行技术

目前较为复杂的嵌入式应用多基于RTOS。在RTOS中，应用的实现都是由线程（任务）完成的。可以把线程看作拥有私有PC指针、寄存器组和堆栈，但有着相同内存空间的进程。多线程的运行，要求在异常处理、线程调度和上下文切换中保持数据的一致性，并有很强的实时要求。很多嵌入式处理器体系结构有用户、特权、异常等多种系统模式，每种模式有私有寄存器组和堆栈，并可进行（处于用户模式时除外）不同模式的切换控制，这就为多线程的运行提供了结构上的支持，为使上下文的切换和异常处理更有效，嵌入式处理器在线程切换和异常处理代码的返回时，能提供状态寄存器、系统模式切换的高效指令。如ARM V6的一条CPS指令，能取代ARM V4T中的四条操作指令，显著提高了效率。

早期的嵌入式处理器用类SWAP指令实现旗语（semaphore）的，它挂起外部总线直到指令执行完成，这在多处理器线程层的并行操作是无法接受的，一个处理器会把整个总线挂起，直到该指令执行结束，这期间其它处理器的执行均被停止[4]。新的嵌入式处理器结构中利用内存中的监控器（exclusive monitor），增加了排他的数据读写操作，即它们的执行不会影响其它处理器的运行。

先进的嵌入式处理器往往采用6级以上的流水线，针对加载/存储和乘/累加有独立的并行流水线，通过并行的加载/存储单元，可以不用等待慢速内存而继续执行，这直接提高了处理器的性能。

通过存储器管理单元（MMU）性能的增强，用物理地址标记的缓存，取代以往的虚地址标记缓存，可大大改善RTOS的上下文切换性能。在虚地址标记缓存中，包含旧的虚地址到实地址的转换关系，每次上下文的切换都要清理缓存，采用实地址减少了外部内存的访问次数，降低了运行功率，整体性能约能提升20%。

4 多内核处理器的并行技术

在遭遇高频率带来的高功耗，硬件侦测指令并行的复杂性和多个独立处理器编程的不可移植等“瓶颈”困难后，研发者开始在嵌入式系统中采用SMP（symmetric multiprocessor，对称式多处理器）系统，SMP系统最大的特点是：系统中有多个处理器，所有处理器处于同等地位；所有处理器共享主存储器，没有自己私用的主存储器。该系统有很好的可靠性，是目前应用最为广泛的多处理器系统。嵌入式应用采用SMP系统，要解决的主要难题，一是多处理器内核之间的通信，二是共享存储器的数据一致性，这需要每个处理器内核必须提供开放的、可挂接的SMP结构的支持，同时要充分利用多处理器内核硬件平台的优点，构造一个对称的、缓存一致的软件平台[5]。

4.1 低功耗旋转锁的设计

多处理器访问共享数据时线程之间的同步开销，是嵌入式SMP系统要解决的问题之一。在很多SMP同步机制的解决方案中，利用内存的一个变量作为软件旋转锁，一个内核访问共享资源前，必须通过一个原子操作得到锁，访问结束后再释放该锁。在SMP OS中，当一个处理器占有锁的时候，另一

（下转第277页）

versity Of Virginia. Dept. of Computer Science, May 1993
11 Kuo T W, Mok A K. Load adjustment in adptive real time systems. Proceedings of IEEE Real Time Systems Symposium, Dec. 1991

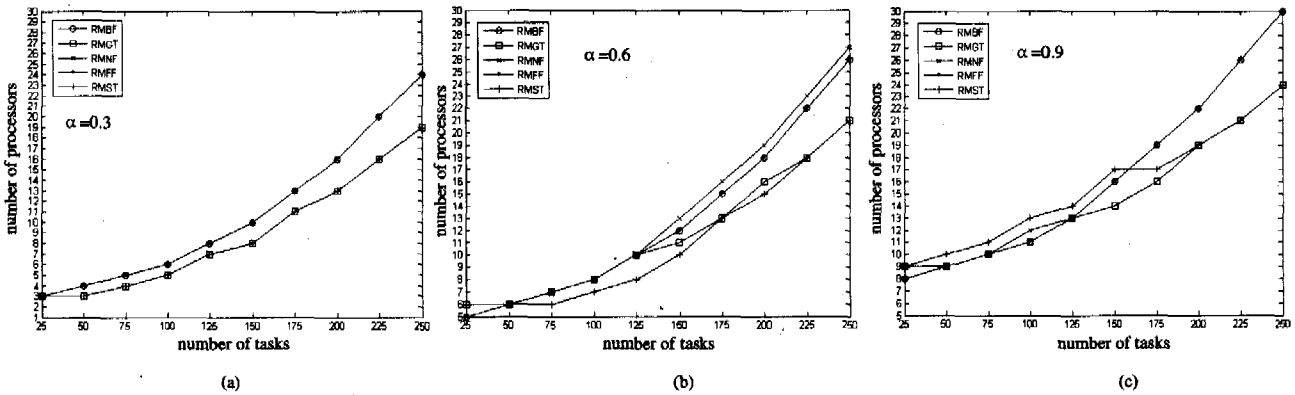

图 3 任务数与处理器数关系曲线

表 2 仿真结果

α	ρ_{RMBF}	ρ_{RMGT}	ρ_{RMNF}	ρ_{RMFF}	ρ_{RMST}
α=0.3	[1.41,1.66]	[1.13,1.47]	[1.41,1.66]	[1.41,1.66]	[1.13,1.47]
α=0.6	[1.40,1.54]	[1.16,1.70]	[1.42,1.54]	[1.40,1.54]	[1.15,1.54]
α=0.9	[1.20,1.41]	[1.13,1.35]	[1.28,1.41]	[1.20,1.41]	[1.13,1.48]

(上接第 263 页)

个处理器必须等待，等待期间通过短循环的不断测试，最终获得锁。对这种方法的改进是使用后退循环，不再不停地访问锁状态，以减少总线竞争，但同样有系统能量的消耗。嵌入式 SMP 系统的内核中设计一对低功耗的锁等待睡眠指令，一旦等待锁，该线程立即进入睡眠，锁释放时立刻唤醒等待该锁的线程。这种旋转锁的设立，减少了总线竞争并明显地降低了系统的消耗。

4.2 Cache 的一致性

一般 SMP OS 能提供 Cache 一致性的映像区，把数据放在 Cache 中能维持数据的稳定。现有维持一致性的方法通常是通过增加信号，扩展系统总线来控制和检测其他处理器的 Cache。在嵌入式系统中，系统总线的时钟通常慢于 CPU，上述方法除了造成 CPU 与 Cache 之间的瓶颈外，还大大增加了总线的流量和功耗。多内核嵌入式处理器系统设计了一个位于多处理器之间的智能控制单元 SCU(Snoop Control Unit)，它主要用于多处理器的数据一致性控制，由它提供快速的数据路径，使数据在各个 CPU 的 Cache 之间高速移动。SCU 保留了每个缓存行(cache line)物理地址标签的副本，并智能地监督处理器对缓存行的操作。如果一个处理器修改了一个缓存行，另一个处理器对该缓存行先读后写，SCU 就假定这个地址今后也会进行同样的操作，如果同样的操作再次发生，SCU 会自动把缓存行的状态置为无效，而不是消耗能量先把它置为共享状态，这就可以在不引发外部内存操作的情况下，让处理器把缓存行直接传输到其他的处理器中，从而降低对有着共同缓存行的处理器进行 Cache 操控的机率。

4.3 多处理器的通信

SMP 多处理器系统用旋转锁来同步处理器之间的通信和对共享资源的访问，避免了通过访问存储器实现通信。异步运行的系统之间必须经常进行同步，使用的一种方法是通过中断系统来激活另一个处理器。这里的中断是一个特殊的中断系统，它通过 I/O 外设而不是 CPU 来触发中断。假设一个多线程应用程序，在某一个处理器上运行的线程会改变处理器的状态，并且这种改变对该应用程序运行在其他处理器上的线程不是硬件一致的。为了维护一致性，操作系统必须把这些对内存映射的修改同步到其他处理器中，首先把新的内存映射配置到自身的处理器中，随后用低竞争的私有外设总线发中断控制信号，中断控制器立刻触发其他的处理器，使它们根据中断 ID 信息，进行各自的内存映射更新。利用这种中断机制，除了可以对应用程序进行动态负载均衡外，也可以对中断处理进行动态负载均衡，即目前负载最轻的处理器最适合处理中断，并把中断发往该处理器。

结束语 嵌入式处理器往往针对应用固有的并行性特征设计微体系结构，这是大幅度提高性能、降低功耗和设计复杂性的有效方法。这几年，像 ARM、MIPS、POWER PC 等主流嵌入式处理器都为此作了很大的努力，并以很快的速度推出新的处理器体系架构，为嵌入式系统满足更强的功能、更复杂的运算、更低的功耗，创造了应用的基础条件。作者在近期的一些嵌入式项目开发中，切身体会到嵌入式处理器结构的发展，特别是并行技术的创新和应用，对更高需求的满足、研发周期的缩短和开发成本的降低所带来的深刻影响[6]，并以此让更多从事嵌入式开发的同仁分享嵌入式处理器发展的成果。

参 考 文 献

1 Seal D. ARM Architecture Reference Manual [M]. 2nd ed. Pearson Education Limited, 2001
2 Sloss A, et al. ARM System Developer's Guide [M]. Morgan Kauffman. 2004
3 ARM Data Engines for Complex Multimedia Solutions . http://www.arm.com/.
4 Liu J W S. 姬孟洛，李军，王馨，等译. 实时系统[M]. 北京：高等教育出版社，2003
5 Andrew N. Sloss. 沈建华译. ARM 嵌入式系统开发-软件设计与优化[M]. 北京：北京航空航天大学出版社，2005
6 魏洪兴，周亦敏. 嵌入式系统设计与实例开发实验教材 I [M]. 北京：清华大学出版社，2005