DOI: 10.3785/j.issn.1008-973X.2009.07.001

支持多线程处理器的实时操作系统实现研究

成杏梅, 刘 鹏, 顾雄礼, 江国范, 姚庆栋
(浙江大学 信息与电子工程学系, 浙江 杭州 310027)

摘 要: 考虑到多线程处理器开发对实时操作系统的需求, 结合其硬件特点, 对已有的实时操作系统进行修改, 给出了多线程处理器的操作系统实现. 这种处理在充分利用多线程技术特点的同时, 还可以保留原有实时操作系统的性能优势. 以 MPEG-1 解码系统为例, 给出了其在媒体系统芯片和多线程处理器上基于实时操作系统的任务调度实现. 实验结果表明, 提出的操作系统实现方案充分利用了多线程技术, 能够提升系统的实时性能, 降低实时操作系统的管理开销, 并简化终端程序员的编程工作.

关键词: 多线程; 实时操作系统; 系统芯片; 任务调度
中图分类号: TP316 文献标识码: A 文章编号: 1008-973X(2009)07-1177-05

Study on implementation of real-time operating system that supports multi-threading processor

CHENG Xing-mei, LIU Peng, GU Xiong-li, JIANG Guo-fan, YAO Qing-dong
(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou* 310027, *China*)

Abstract: According to the requirement for the real-time operating system (RTOS) of the development of the multi-threading processor, an implementation plan of RTOS for multi-threading processor that combined the hardware characteristics was presented by modifying the existing RTOS. The proposed method can maintain the performance of the existing RTOS when fully utilizing the multi-threading technology. The MPEG-1 decoder system was taken as an example and the task scheduling based on the RTOS on the Media-SoC was compared with that on the multi-threading processor. Experimental results indicated that the presented implementation plan fully utilized the multi-threading technology, and improved the real-time performance of system, decreased RTOS overhead and simplified programming work.

Key words: multi-threading; real-time operating system (RTOS); system-on-chip (SoC); task scheduling

实时操作系统(RTOS)已成为系统集成芯片(system on chip, SoC)的重要组成部分[1], 它有利于应用程序的开发及系统集成芯片的功能验证[2]. 应用于系统集成芯片的实时操作系统的实时性要求很高, 并且受芯片成本的影响, 实时操作系统的代码体积会有严格的限制[3]. 已经有很多成熟的商业版的实时操作系统, 如 VxWorks、QNX、WinCE 等. 采用这些操作系统需要负担额外的费用; 另一方面, 有些嵌入式实时操作系统(如 WinCE)的代码量很大, 达到 200 kB[4]. 考虑到自主设计芯片的需要, 实验室开发了实时操作系统 Iota[5], 它实现了任务间的通信和同步管理、资源管理(存储器、外设)以及异常处理等工作.

与提升系统的时钟频率、超标量[6] 和超长指令字 VLIW[7] 等技术相比, 多线程技术有利于开发粗粒度的线程间的并行性, 它以增加较少的硬件资源, 带来

收稿日期: 2008-01-15. 浙江大学学报(工学版)网址: www.journals.zju.edu.cn/eng
基金项目: 霍英东教育基金资助项目(94031); 国家自然科学基金资助项目(60873112).
作者简介: 成杏梅(1978—), 女, 河南济源人, 博士生, 从事实时操作系统等方面的研究. E-mail: amity@zju.edu.cn
通讯联系人: 刘鹏, 男, 副教授. E-mail: liupeng@zju.edu.cn

处理器性能的提升,从而成为提升嵌入式系统领域单处理器性能的研究热点.线程的定义不十分明确,可以是一个进程,可以是一个任务[8],本文不做区分.多线程技术发展的一个趋势是与多处理器系统芯片(multi-processors system on chip, MPSoC)相结合.MPSoC 上某个核或多个核采用多线程技术,使芯片兼备 MPSoC 和多线程技术的性能优势.

陈科明[9]所在实验室研发的 RISC3202 处理器核也采用了多线程技术.考虑到不同应用的需求及处理器快速的开发实现,RISC3202 处理器是以两个简单的 RISC3201 核为基础,通过扩展其功能模块,并重新设计 RISC3202 的数据通路和控制通路,而发展起来的一个多模式工作的处理器核,包括双发射超标量模式(简称双发射模式)、双处理器核模式(简称双核模式)以及双线程模式等 3 种工作模式.其中,RISC3201 处理器核已经在 Media-SoC[10]系统上经过流片验证.考虑到多线程处理器开发对操作系统的需求,本文对实时操作系统 Iota 进行改进,讨论了多线程处理器实时操作系统的实现方案.

1 实时操作系统的实现

1.1 任务的调度管理

任务管理是实时操作系统最重要的内容之一.为了满足系统集成芯片对实时性的要求,Iota 采用了基于优先级的抢占式调度算法.在 Iota 中,任务有以下状态:新建、就绪、执行、阻塞和中断,如图 1 所示.Iota 提供了系统函数用于实现任务的状态转移和任务间的切换执行,从而实现多任务的管理.Iota 内任务的伪代码如下所示:

```
Task{
        while(1){
        Task code;
        Iota function;
        Task code;
    }
}
```

1.2 存储管理

虚拟存储管理会造成延时访问的不确定性,从而在实时性要求比较高的场合很少使用.Iota 结合硬件特点,设计了一个简单的虚拟存储管理机制,将应用程序分割在不能互相访问的域上,它避免了访问时延的不确定性,同时又增强了系统的健壮性和应用程序的可移植性.

运行 Iota 的处理器实现了基本的内存管理单元(MMU),其中应用程序空间为 0 ~ 0x7fffffff,需要经过 TLB(translation lookaside buffer)表映射.硬件会根据相关寄存器的值和虚拟地址查询 TLB,进行虚实地址翻译.由于嵌入式系统应用的可测性,Iota 将每个任务的空间,包括代码段、数据段和堆栈段的虚实地址空间的对应关系写进一个数据结构,并在任务执行前写进 TLB 表项,任务在执行时将任务号赋值给相应的寄存器从而进行正确的虚实地址翻译.Iota 内每个任务号都是唯一的,从而避免了错误的地址翻译造成系统的不稳定.

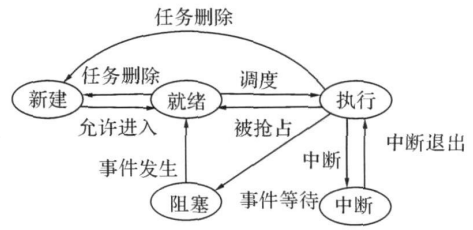

图 1 **Iota** 任务状态转换图

Fig. 1 Diagram of Iota task state transition

2 实时操作系统对多线程技术的支持

2.1 多线程处理器软件架构

多线程处理器通常为每个线程维护独立的 PC(program counter)和寄存器,同时存在一种机制触发线程切换(如某个线程发生 cache miss),线程切换的代价要尽可能小.多线程技术不仅适用于传统的高性能应用(如网络路由器),也非常适合实时性要求很高的消费类应用任务.多线程技术利用新增的硬件资源实现了线程之间的硬件切换,提升了系统性能.以数字电视为例,通常会有一个控制线程和一个处理函数(如音频解码程序).处理器在绝大部分时间内执行音频解码程序(记为主线程 A);同时为了响应用户的控制,处理器要能够及时切换到控制线程(记为辅线程 B);实时性要求很强的线程 B 被处理完成之后,返回主线程 A.

另一方面,在软件实现上,多线程技术由于自身的实现特点,能够在已有软件架构的基础上,经过一些修改,实现自身的软件开发系统.为了叙述方便,在接下来的讨论中,以 RISC3202 为目标,讨论支持多线程处理器操作系统的实现.

2.2 RISC3202 多线程的实现

RISC3202 具备多线程处理器的特点,但是与一般的多线程技术实现不同的是,RISC3202 是以两个简单的 RISC3201 核为基础形成的多模式工作的处理器,它能够根据应用的场合配置成双发射模式或

者双线程模式, 构造灵活. 在某些应用场合中, 任务是一个不可拆分的单线程任务或者不存在可并行执行的多个任务, 此时采用高性能的多发射架构比较合适. 在某些场合中, 由于存在 cache miss 或者其他事件引起处理器长时间停顿造成性能损失, 此时采用多线程技术尤为合适.

多线程技术非常适合实时应用, 它通常给实时任务一个特定的处理通道, 当然不同的多线程处理器实现不同, 在 MIPS34K 中它是通过其特有的 QoS 机制[11] 实现的. 在 RISC3202 中, 当实时任务需要得到处理时, 处理器从双发射模式切换到双线程模式, 使实时任务得到及时处理. RISC3202 处理器同通常的多线程技术一样, 实现了线程之间的硬件切换, 降低了线程切换的开销, 也增加了系统的实时处理能力.

2.3　多线程处理器操作系统的实现方案

根据 RISC3202 处理器的多线程技术实现特点, 考虑对原有的实时操作系统进行修改, 使其支持 RISC3202 处理器, 缩短 RISC3202 的开发时间, 进一步提升系统实时处理的能力.

RISC3202 处理器可以根据应用需要从双发射模式切换到双线程模式, 如图 2 所示. 图 2(a)给出了双发射工作模式下的软硬件状态, 图 2(b)给出了双线程工作模式下的软硬件状态. 在双发射工作模式下, 一个实时操作系统及其管理下的进程运行在一个处理器核上, 记为主线程 A. 在双线程工作模式下, 主线程 A 和实时性更强的线程 RT(real time thread)同时运行, 记为两个硬件线程 HT(hardware thread). 通过对已有的实时操作系统进行修改, 使其支持 RISC3202 处理器, 用户不用关心硬件的实现细节, 这样可以简化程序员的编程工作, 并保留原有操作系统的性能优势.

在实时操作系统中, 需要做的修改如下:

(1) RISC3202 处理器有 3 个寄存器文件, 为方便起见, 将它们命为 RF1、RF2、RF3. 其中, 主线程使用寄存器文件 RF1, 线程 RT 使用寄存器文件 RF2, 寄存器文件 RF3 作为备用. 当系统发生异常时, 某个进程被中断执行, 程序跳转到中断程序入口. 此时操作系统首先将主线程寄存器文件 RF1 备份到寄存器文件 RF3, 而不是直接将被中断程序的上下文保存到内存中.

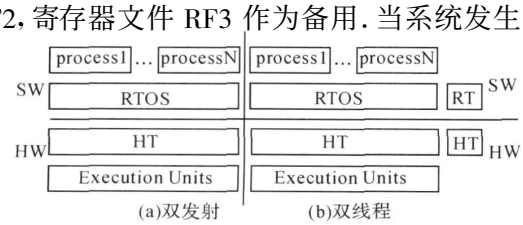

图 2　双发射和双线程模式下软硬件状态

Fig. 2　Software and hardware state under dual-issue and dual-thread mode

(2) RISC3202 处理器提供了一对指令, 即 INC_thread rs 和 DEL_thread. 指令 INC_thread rs 使 RISC3202 从单硬线程运行状态进入双硬线程运行状态, 寄存器 rs 保存的新线程起始地址开始运行. 执行指令 DEL_thread, 使 RISC3202 从双硬线程运行状态切换到单线程运行状态. 某种中断或异常的发生要求线程 RT 开始运行, 实时操作系统在中断服务程序调用 INC_thread rs 指令恢复线程 RT 的运行, 中断服务程序非常简单.

(3) 操作系统在退出中断处理程序(即执行 IntExit()函数)时, 会调用调度器检测是否有更高优先级的任务就绪(中断发生也许会使某个软线程就绪). 如果有就将寄存器 RF3 中的文件保存到内存中并从内存恢复高优先级任务的上下文执行; 如果没有, 则将被中断程序从寄存器文件 RF3 读到寄存器文件 RF1 恢复执行. 将寄存器文件 RF1 的内容备份到寄存器文件 RF3 或者从寄存器文件 RF3 恢复被中断进程的上下文到寄存器文件 RF1, 可以通过一对硬件指令来实现.

以操作系统中断处理程序 IntExit()函数的执行为例, RISC3202 处理器中其实现过程如下(其中粗体部分为变化部分):

```
IntExit{
    对就绪队列进行判断是否有更高优先级的任务执行;
    if(有更高优先级的任务就绪)
    {
    将备份寄存器的内容及 EPC 寄存器的内容等写入当前任务的保存现场的内存中;
    执行任务切换函数 taskswitch();
    }
    else
    {
    将备份寄存器的内容写进 RF1;
    将程序中断的地址即 EPC 寄存器的内容写入寄存器 rs;
    程序跳转到 rs 指向的地址;
    }
}
```

实时线程的示意代码如下所示:

```
RT_func()
{
    real time thread code;
```

```
    DEL_thread;
}
```

当实时线程执行完毕后,执行指令 DEL_thread,使 RISC3202 从双线程运行状态切换到双发射运行状态.操作系统和实时线程协同工作,完成系统工作模式的切换,充分发挥硬件资源的能力.

3 实 验

本文以 MPEG-1 解码系统为例,进行了其在 Media-SoC 系统上和 RISC3202 处理器上任务管理调度的实现和比较. MPEG-1 解码系统可以分解为系统解复用任务、音频解码任务和视频解码任务 3 个基本任务.系统解复用任务是实时性任务,必须对标准复合码流进行及时处理,但不必连续处理.系统解复用任务正确解码要求不产生复合码流的堆积.

3.1 Media-SoC 上任务调度的实现

Media-SoC[10] 是一个双核系统,它包括一个主控 RISC 核及一个 DSP 核.在进行 MPEG-1 解码系统任务管理时,实时操作系统 Iota、系统解复用任务以及音频解码任务运行在 Media-SoC 的 RISC 核上,视频解码任务运行于 Media-SoC 的 DSP 核上.系统解复用任务正确运行输出压缩的音视频码流给音频解码任务和视频解码任务,正因为如此,系统解复用任务在 3 个任务中优先级最高.

Media-SoC 系统中系统解复用任务缓冲区大小为 1 kByte,系统解复用任务内部采用数据驱动原则,当缓冲区内未解码的数据大于门限值要求时,开始解码.为了使得缓冲区内的数据不溢出,采用定时中断周期性地进行系统解复用任务执行.系统解复用任务伪代码实现如下所示:

```
task_system( )
{
  initialization code;
  while(1){
    if(buffer data size> threshold)
    { system decoder action;}
     TimeDelay (1);
  }
}
```

操作系统依据 TimeDelay( )和 timetick( )完成了图 1 中任务状态的转换,实现了 RISC 核上系统解复用任务和音频解码任务的切换执行,并输出视频压缩码流给 DSP 核上的视频解码任务,完成 MPEG-1 解码系统.在 162 MHz 时钟频率下,1 kByte 大小的系统解复用任务缓冲区设计以及 1 ms 时钟节拍的设置,驻住在片外存储器的实时操作系统 Iota 带来的系统管理开销是 4.92 MIPS.

3.2 RISC3202 处理器上任务调度的实现

当 MPEG-1 解码系统在 RISC3202 处理器实现时,考虑到系统解复用任务的强实时性要求,在 RISC3202 处理器上将其实现为一个实时线程 RT.这里采用硬件观测系统解复用任务缓冲区内剩余的数据,当其大于一定的门限值时产生中断进行系统解复用任务的执行. RISC3202 处理器在大部分的时间里进行实时操作系统及其上的音频解码任务的执行.当中断发生时,中断服务程序调用指令 INC_thread rs 恢复系统解复用任务的执行.当操作系统退出中断服务时,检测到音频解码任务仍是当前优先级最高的任务,音频解码任务通过硬件切换恢复其执行现场,和系统解复用任务作为两个线程并行执行.系统解复用任务完成此次解码过程,执行指令 DEL_thread,硬件切换回双发射执行模式.

在 RISC3202 处理器上,当系统解复用任务执行完毕,切换到音频任务执行时,通过调用指令 DEL_thread,硬件切换回双发射执行模式,硬件切换开销为 6 条指令;另一方面,当中断发生时,系统解复用任务需要得到及时处理.此时,操作系统首先将主线程寄存器文件备份,并通过简单的异常分析跳转到相应的中断服务程序,调用指令 INC_thread rs 恢复系统解复用任务的运行,主线程接着退出异常处理程序,通过调度器检测到音频任务仍是优先级最高的任务,则音频任务现场从备份寄存器 RF3 恢复到主线程寄存器文件,音频解码任务恢复执行,整个过程大约需要 22 个指令周期.

将上述两次切换作为一个周期,则每个周期的系统切换开销为 28 个指令周期.考虑到系统解复用任务的输入码流速度为 1.5 Mbps,当设置进行系统解复用任务的门限值为 800 Byte 时,在 RISC3202 处理器上,系统完成 MPEG-1 解码系统,在 1 s 内需要进行约 235 个周期的切换过程,则系统开销为 0.006 5 MIPS.

3.3 实验结果

当 MPEG-1 解码系统在 Media-SoC 和 RISC3202 上运行时,各个任务及系统管理开销如表 1 所示.系统解复用任务始终是单线程执行,因而其运行开销在 Media-SoC 和 RISC3202 上都是一样的.RISC3202 双发射执行模式可以获得的加速比依赖于编译器及程序中指令的并行性[12].音频解码任务针对 RISC3202 结构进行了部分的软件优化,其在 RISC3202 上的执行开销为 34 MIPS.而操作系

统的管理开销在 Media-SoC 和 RISC3202 上相比较, 差别很大, 分别为 4.92 和 0.007 MIPS. 另一方面, 由于多线程处理器实现了线程的硬件备份, 系统对中断的响应速度也有很大的提升, 从原来 70 条左右的指令开销到现在的 6 条指令开销. 可以看出, 操作系统和实时线程的共同作用实现了系统从双发射到双线程模式的自由切换, 屏蔽了硬件的实现细节, 发挥了硬件细粒度(双发射)和粗粒度(双线程)指令并行执行的能力, 降低了操作系统的任务管理开销, 增强了系统的实时处理能力.

表 1　Media-SoC 和 RISC3202 上任务的开销比较

Tab.1　Overhead comparison of tasks on Media-SoC and RISC3202　　MIPS

处理器类型	系统解复用任务	音频解码任务	系统管理开销
Media-SoC	9.4	48	4 920
RISC3202	9.4	34	0.007

4　结　语

多线程处理器支持线程间的并行执行, 线程切换开销小. 实时操作系统对多线程处理器的支持, 使得软线程和硬线程并行存在, 既保证了系统的实时性, 又使得用户只需着眼于操作系统下软线程的编写, 简化了程序员的编程工作. 实时操作系统对多线程处理器的充分支持, 能够提升系统的实时性能, 降低实时操作系统的管理开销.

RISC3202 处理器是一个高性能多模式工作的处理器. 陈科明[9] 实验室正在研发的高性能多处理器系统芯片 MediaSOC64X, 其主控芯片采用 RISC3202. 将来考虑由 RISC3202 实现多路系统解复用任务及音频解码任务, 而 MediaSOC64X 中的每个 DSP 处理器核完成一路视频解码任务. 成杏梅[13] 提出了适用于用于多处理器系统芯片的面向对象的调度, 结合本文中的研究成果, 可实现 MediaSOC64X 上操作系统的运行.

参考文献(References):

[1] JERRAYA A A, FRANJAO, LEVY M, et al. Roundtable: envisioning the future for multiprocessor SoC[J]. **IEEE Design & Test of Computers**, 2007, 24(2): 174-183.

[2] GAO Feng, LIU Peng, YAO Qing-dong. A methodology for platform based high-level system-on-chip verification[J]. **Chinese Journal of Electronics**, 2003, 12(1): 61-64.

[3] PAULIN P G, PILKINGTON C, LANGEVIN M, et al. Parallel programming models for a multiprocessor SoC platform applied to networking and multimedia[J]. **IEEE Transactions on Very Large Scale Intergration (VLSI) Systems**, 2006, 14(7): 667-680.

[4] ZUBERI K M, SHIN K G. EMERALDS: a small memory real-time micro-kernel[J]. **IEEE Transactions on Software Engineering**, 2001, 27(10): 909-928.

[5] 高丰, 刘鹏, 姚庆栋, 等. 一种基于 HDTV 信源集成解码芯片的 RTOS 的设计与实现[J]. 电路与系统学报, 2002, 7(3): 45-49.
GAO Feng, LIU Peng, YAO Qing-dong, et al. Design and implementation of RTOS for a HDTV integrated source decoding chip[J]. **Journal of Circuits and Systems**, 2002, 7(3): 45-49.

[6] BANERJEE S, SHEIKH H R, JOHN L K, et al. VLIW DSP vs. superscalar implementation of a baseline h.263 video encoder[C] //**Proceedings of Thirty-fourth Asilomar Conference on Signals, Systems and Computers**. Pacific Grove, CA, USA: IEEE, 2000: 1665-1669.

[7] TALLA D, JOHN L K, LAPINSKII V, et al. Evaluating signal processing and multimedia applications on SIMD, VLIW and superscalar architectures[C] //**Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors**. Austin, TX, USA: IEEE, 2000: 163-172.

[8] 斯威特曼. MIPS 体系结构透视[M]. 2 版. 北京: 机械工业出版社, 2007.

[9] 陈科明. 媒体多处理器系统芯片的设计研究[D]. 杭州: 浙江大学, 2007.
CHEN Ke-ming. Study on the design of multi-processor system-on-chip for media processing[D]. Hangzhou, Zhejiang University, 2007.

[10] LIU Peng, WANG Wei-dong, XIAO Zhi-bin, et al. MediaSOC: a system-on-chip architecture for multimedia application[C] //**Proceedings of IEEE International Workshop on VLSI Design and Video Technology (IWVDVT2005)**. Suzhou, China: IEEE, 2005: 203-206.

[11] MIPS Technologies Inc. The MIPS32 34K™ core family: power next-generation embedded SoC[EB/OL]. [2006-09-02]. http: // www.mips.com

[12] YAO Ying-biao, WANG Bin, ZHANG Jian-wu, et al. RISC3202: a software configuration dual-issue/dual-core microprocessor[C] //**IFIP International Conference on Network and Parallel Computing Workshops**. Dalian, China: IFIP, 2007: 963-968.

[13] 成杏梅. 基于媒体芯片的实时操作系统实现研究[D]. 杭州: 浙江大学, 2008.
CHENG Xing-mei. Study on RTOS based on system on chip for media application[D]. Hangzhou, Zhejiang University, 2008.