一种实时操作系统硬件加速设计*

沈国新，张德学，王桂海，焦汉明
（山东科技大学 信息科学与工程学院，山东 青岛 266510）

摘 要：RTOS是嵌入式系统中重要的组成部分，但其本身的运行使整个系统的性能下降。针对RTOS的任务调度和时间延时处理部分进行分析，并加以硬件实现。在运行63个任务时，采用硬件加速模块，任务响应时间为2 180个时钟周期。相比没有硬件支持的系统，任务响应时间可降低85.8%，提高了系统的可预测性。

关键词：RTOS；任务调度；时间延时；任务响应时间；可预测性

中图分类号：TP316.2 **文献标识码**：A

Design of accelerating RTOS in hardware

SHEN Guo Xin, ZHANG De Xue, WANG Gui Hai, JIAO Han Ming
(College of Information Science and Engineering, SDUST, Qingdao 266510, China)

Abstract: RTOS is an important part of embedded system, but it will degrade the performance of the system when running. In this paper, the task schedule and time delay functions in RTOS have been analysised and redesigned by hardware. Under running 63 tasks with hardware accelerator, the task response time is 2 180 clock cycles. Compared to running pure software, the task response time can be reduced by 85.8%, the predictability of the system has been improved.

Key words: RTOS; task schedule; time delay; task response time; predictability

随着科技的进步，嵌入式系统的功能逐渐由简单向复杂发展，开发难度也随之提高。嵌入式操作系统的使用，屏蔽了部分硬件信息，提供给开发者统一的平台，降低了开发难度，提高了代码的重复利用率。在一些特殊的领域（医疗、汽车、航空航天），对嵌入式系统的实时性要求非常高。在这些场合，任务必须在给定的时间内响应并正确完成。而实时操作系统RTOS（Real Time Operation System）本身的运行，必然会引起性能的下降，在任务数量增加时，这种下降更加明显。例如，使用μC/OS-II实时操作系统在PowerPC处理器上运行，在TimeTick（时钟节拍）周期为10 μs、运行64个任务的情况下，TimeTick中断函数占用的CPU时间已达到42%[1]。

目前，RTOS软件层面的研究已经很成熟，可有效提高RTOS性能的方法有以下几种：

（1）提高处理器的运行频率[2]。这对功耗相当敏感的嵌入式系统并不是好方法。同时高频时钟所引起的电磁干扰对电路板布线的要求也更高；

（2）设计专用于RTOS系统服务的硬件。硬件对相同的操作可并行处理。如果设计一种硬件，在任务数量或TimeTick频率增加的情况下，系统也能在固定的时钟周期内完成所有任务域的更新，从而降低RTOS运行所占的CPU时间。

本文设计了实时系统加速RTA（Real-Time Acceleration）模块，对任务调度和系统时间管理进行硬件化，降低了任务中断时间，并对最终的测量数据进行对比，得出结论。

1 RTA的硬件设计

本文的硬件平台使用OR1200[3] CPU，它是一款由Open Cores网站维护的开放源代码CPU，内部结构可见可修改，且没有版权问题。RTA模块作为从设备连接到Wishbone总线[4]上。在RTA模块中，由硬件实现任务管理和时间管理。RTA中的寄存器全部映射到内存空间上，软件通过对寄存器的访问来控制RTA模块的运行。

该专用硬件可分成如下两部分：

* 基金项目：青岛市科技计划资助项目（07-2-3-1-jch）

(1)任务管理和时间管理部分。RTA 模块支持 64 个任务,使用基于优先级的调度策略,每个任务有唯一的优先级。RTA 只在需要任务切换时才中断 CPU。时间延时的最小单位是 TimeTick(时钟节拍),最长时间延时可达 65 535 个 TimeTick;

(2)用于产生 TimeTick 信号的 Timer(计时器)。RTA 必须有独立的 Timer 为其产生 TimeTick 信号。在本文中,利用 OR1200 自带的 Timer 完成此工作。

本文使用的系统是在 μC/OS-II 实时操作系统基础上改进实现的。该 RTOS 由 Micrium 网站维护,已经应用于商业产品[5]。整个软硬件的实现在 FPGA 开发板 DE2-70 上完成,系统时钟频率为 25 MHz。

1.1 任务管理和时间管理

任务管理和时间管理的设计框图如图 1 所示。

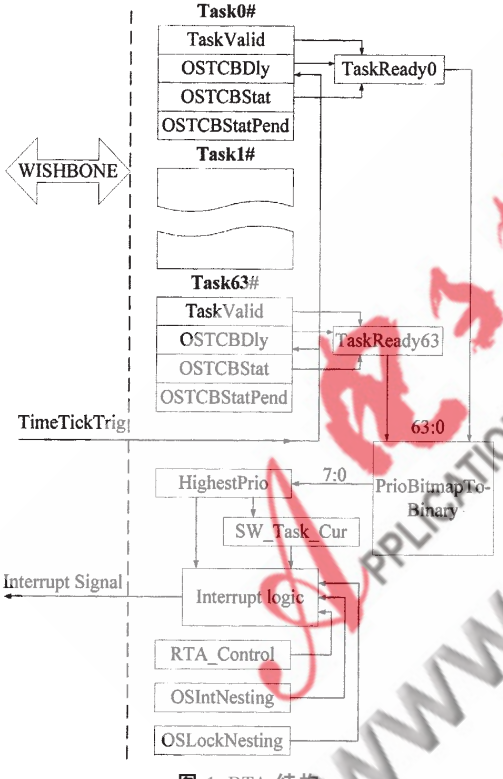

图 1 RTA 结构

每个任务都有 4 个域:TaskValid、OSTCBStat、OSTCBDly 和 OSTCBStatPend。每个任务都有一个任务就绪标志 TaskReady,RTA 通过 PrioBitmapToBinary 模块找到最高的优先级并送给 HighestPrio。在 CPU 响应外部中断或者给调度器上锁时,可以通过 OSIntNesting 和 OSLock-Nesting 寄存器关闭 RTA 的中断。

μC/OS-II 实时系统内核中,任务调度基于 TimeTick 完成,由于程序只能顺序执行,任务的 timedly 域更新也是顺序执行的,从而使得调度函数的执行时间与运行的任务数量有关。在 RTA 模块中,基于 TimeTick 的调度机制并没有改变,只是原型中顺序执行的 timedly 更新,在硬件中可以同时执行。在使用 RTA 模块的系统中,移去了软件中的用于任务调度的数据结构,相应地在硬件中予以实现。

当有更高优先级的任务进入就绪态时,就会产生 RTA 中断。硬件实现上,当进入就绪态的上个时钟周期的最高优先级和本时刻的最高优先级不同时,便产生中断信号。在 μC/OS-II 中,每个 TimeTick 时刻都会发生中断,这就需要更频繁地保存 CPU 寄存器,相比本文提出的方法,浪费了更多的 CPU 时间。

1.2 TimeTick 信号的产生

RTA 的运行需要一个可配置的 Timer 来为其产生 TimeTick 信号。在本文中,通过对 OR1200 进行改造,利用其内部的 Timer 产生中断信号作为 RTA 任务调度的标准时钟节拍,而将 RTA 的中断信号连接到原来 Timer 在 CPU 的接口处。这样,CPU 通过 Wishbone 总线可对 Timer 进行读写,且 RTA 产生的中断不会占用可编程中断控制器 PIC(Programmable Interrupt Controller)。改造后的框图如图 2 所示。

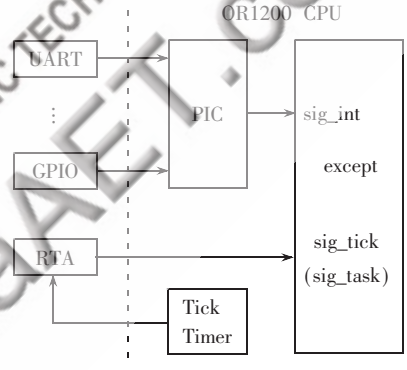

图 2 改造后的 Timer、RTA、OR1200 连接图

1.3 软件实现

因为任务数据结构的改变,源码中所有涉及到任务数据结构的函数都要进行修改。由于任务调度和时间处理由 RTA 模块执行,原先执行 TimeTick 的中断函数要作相应修改,在中断时,只需读取 RTA 中 HighestPrio 寄存器,然后做上下文切换,运行该优先级的任务即可。

2 实验结果

本实验使用的 CPU 为 OR1200,CPU 和所有的外设都通过 Wishbone 总线连接,系统时钟为 25 MHz。在 Altera 的 Cyclone II FPGA 平台上,使用 Quartus 8.1 工具对 RTA 进行布局布线,其共占用 4 197 个逻辑单元 LE(Logic Element)。

任务响应时间是 RTOS 性能的一个重要指标,其定义为:从任务中断产生的时刻起,到恢复任务执行之间的时间。试验中,利用自定义的 Timer 作为测量标尺,在 2 个测试点各读取一次,相减后的数值再乘以此 Timer 的周期,便得到该段测试时间。图 3 是有硬件加速和无硬件加速的任务响应时间的测试结果,单位是系统时钟周期。

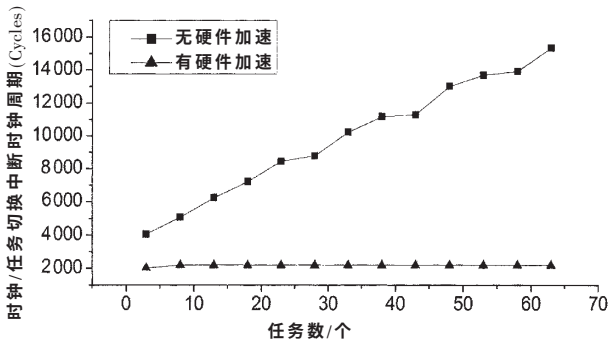

图 3 任务响应时间测试结果

从图中 3 可以看出，在无硬件支持的 RTOS 中，随着任务数的增加，任务响应时间也随之呈线性增加。其原因是，程序顺序执行，在无硬件加速的情况下，RTOS 内核在每个 TimeTick 中断都要对任务的延时域进行顺序更新。随着任务的增加，延时域的处理时间也增长。有硬件加速支持时，任务响应时间缩短，而且与正在运行的任务数量没有关系。这是因为所有任务的延时域都同时更新，在一个时钟周期内即可全部完成。所以使用 RTA 模块后，降低了系统本身占用 CPU 的时间，提高了系统的可预测性。可见，在添加 RTA 模块后 RTOS 的性能得到了提高。

本文将 μC/OS-II 系统中调用频繁的任务调度和时间管理采用硬件实现，达到了降低系统负载、稳定任务响应时间、提高系统可预测性的目的。实验结果表明，使用本硬件，任务中断响应时间可降低 85.8%。

参考文献

[1] KUACHAROEN P, SHALAN M, MOONEY V. A configurable hardware scheduler for real-time systems[C]. In International Conference on Engineering os Reconfigurables Systems and Algorithms, 2003.

[2] NORDSTROM S, LINDH L, JOHANSS L, et al. Application apecific real-time microkernel in hardware.Real Time Conference[C]. 14th IEEE-NPSS Volume, 2005.

[3] LAMPRET D, MLINAR M, WIEGELMANN J, et al. OpenRISC 1000 architecture manual[EB].http://www.opencores.org. 2006.

[4] LABROSSE J J 著. 嵌入式实时操作系统μC/OS-II(第2版)[M]. 邵贝贝，译.北京：北京航空航天大学出版社, 2003:7-12.

[5] 倪继利，陈曦，李挥. CPU 源代码分析与芯片设计及 Linux 移植[M]. 北京：电子工业出版社,2007：42-64.

(收稿日期：2009-10-24)

作者简介：

沈国新，男，1983 年生，硕士，主要研究方向：集成电路设计。

张德学，男，1977 年生，副教授，主要研究方向：集成电路设计。

王桂海，男，1960 年生，副教授，硕士，主要研究方向：模拟电子电路设计。