DOI:10.16157/j.issn.0258-7998.2008.05.033

硬件实时操作系统的设计与实现*

崔建华,孙红胜,王保进
(解放军信息工程大学,河南 郑州 450002)

摘 要:在阐述了硬件实时操作系统的结构和运行机制的基础上,着重论述了基于FPGA设计实现的硬件实时操作系统,并在Actel公司的APA075上实现了任务调度、中断管理、定时器管理等实时操作系统基本功能。该硬件实时操作系统具有强实时性、高确定性和低系统开销等优点。

关键词:FPGA;硬件实时操作系统;系统开销

The design and realization of hardware RTOS

CUI Jian Hua, SUN Hong Sheng, WANG Bao Jin
(The PLA Information Engineer University, Zhengzhou 45002, China)

Abstract: In this paper, on the basis of expatiating a hardware RTOS on the structure and operating mechanism, we focused on FPGA-based hardware design to achieve RTOS, and achieved task scheduling, interrupt management, timer management, and other basic functions of RTOS on APA075 of Actel company. The hardware RTOS has merits such as hardware-real-time, high determinacy and low system overhead, etc.

Key words: FPGA; hardware RTOS; system overhead

随着嵌入式技术的发展,实时操作系统RTOS(Real Time Operating System)被越来越多地应用在嵌入式系统中,如:航空航天、工业控制、汽车电子和核电站建设等众多领域。对于现有基于软件实现的实时操作系统,单纯依靠改进调度算法已不能使其实时性有更大的提高。如果采用硬件逻辑实现RTOS中的任务调度、中断处理和定时器管理等功能,则可使其实时性和确定性显著提高。因为硬件逻辑独立于处理器运行,不占用处理器的处理时间,所节省的时间用于执行任务程序,从而提高了任务集合的可调度性和实时性。本文基于"外部处理器+FPGA"的硬件平台结构[1],在FPGA上设计和实现了硬件实时操作系统。

1 硬件实时操作系统的结构和运行原理

硬件实时操作系统能实现典型实时操作系统的各种功能模块,包括:任务调度内核、中断管理模块、定时器模块、资源管理模块和内存管理模块等[2]。为了实现稳定可靠的系统,本文采用逐步演进的方法。首先实现由任务调度内核(也称作硬件调度内核)、中断管理、定时器管理和接口软件构成的最小系统,然后逐步增加其他功能模块。

硬件RTOS的结构如图1所示。调度内核是系统的核心,负责任务的调度管理;中断管理模块负责管理外部中断;定时器模块负责任务延时和周期执行。硬件RTOS的数据通过接口总线与外部系统通信[3]。

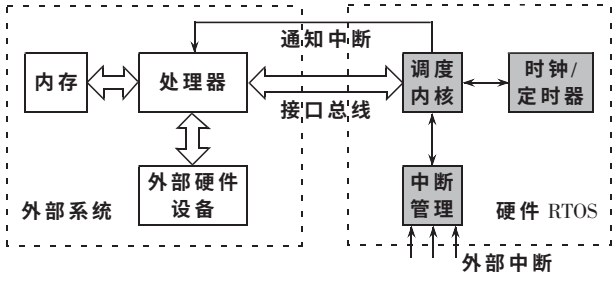

图1 硬件RTOS结构框图

任务调度内核使用FPGA的片内寄存器实现任务控制块TCB(Task Control Block)队列(简称任务队列)。所有未执行任务放在等待队列和就绪队列中。等待队列中的任务在条件满足时将变为就绪任务。本文中,等待任务主要等待定时器时间和中断到达。任务调度内核能在每个调度时机计算出优先级最高的就绪任务。调度内核采用抢占式调度机制,每个调度时机输出优先级最高的

*国家高技术研究发展计划(863计划):2006AA01Z409

就绪任务。

中断模块接管处理器除通知中断以外的所有外部中断，中断服务程序称作“中断处理任务”，与普通任务一样被硬件内核调度。外部中断到来时，触发硬件内核的调度时机，保证中断任务的实时执行。此时，软件实时操作系统的中断嵌套已经转换为处理器中任务的抢占，高优先级的中断任务可以抢占低优先级任务。

定时器管理模块实现任务的延时和周期运行，在设定任务周期或延时时间后，每当设定时间到时便会使对应任务就绪，并触发硬件内核的调度时机，由后者执行一次任务调度。

接口总线硬件实现处理器与硬件 RTOS 之间的数据传输和事件通知。接口软件从功能上分为两类：(1) 系统 API。被应用程序调用，能通过接口总线向硬件 RTOS 发送命令。硬件 RTOS 收到命令后，解析数据，执行相应操作。(2)通知中断服务程序。硬件 RTOS 进行一次任务调度后，如果发现新“选出”的最高优先级就绪任务与当前处理器正在执行的任务不同，则需要进行现场切换。硬件调度内核将利用通知中断告知处理器该就绪任务的 ID 和堆栈地址，触发后者执行中断服务程序，以执行现场切换。

任务存在阻塞(S0)、就绪(S1)和运行(S2)三个状态，定时器存在停止(T0)、运行(T1)两个状态，中断模块存在无效(I0)、有效(I1)两个状态。图 2 描述了最小系统各模块的运行状态。

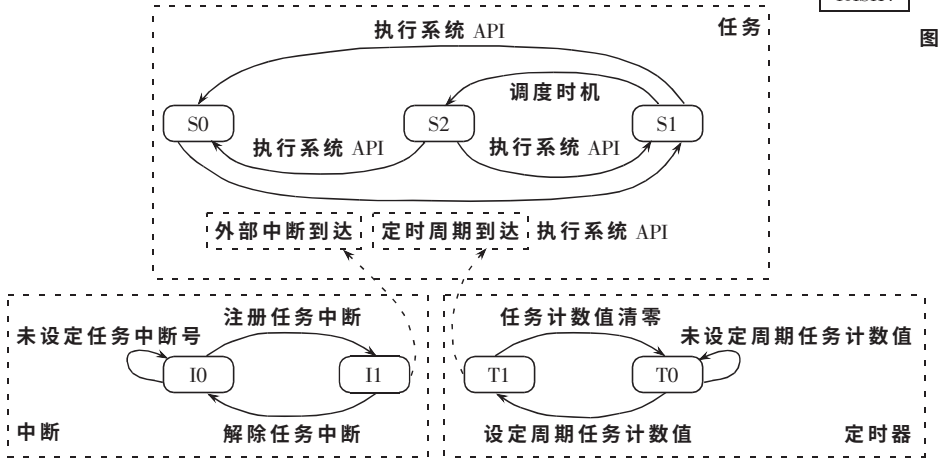

图 2 硬件 RTOS 运行原理

2 系统设计与实现

本文选用 ARM9 系列的处理器 S3C2410 和 Actel 公司的 FPGA 芯片 APA075，用后者实现硬件实时操作系统，调度在处理器上执行的任务。

2.1 任务调度内核的设计实现

实时操作系统的核心是任务调度内核，其主要功能是根据调度算法，在每个调度时机确定下一个将要执行的任务，并适时进行现场切换。为此，调度内核需要维护一系列的任务队列(如就绪任务队列和等待任务队列)，并在每个调度时机，根据优先级重新排列就绪队列，以计算出优先级最高的就绪任务。

合理的任务控制块是实现硬件调度内核的关键，应能根据调度算法分配优先级，并利于优先级比较，而且能够索引最终结果。根据上述需求，设计了如图 3 所示的任务控制块结构。其中 ID 为任务号，State 为任务状态，Prio 为任务优先级，SP_End 为任务堆栈终止地址，Run_Time 为任务起始运行时间，End_Time 为任务结束运行时间。起始和截止时间能根据特定算法分配任务的优先级。Delay_Counter 为任务延时计数器，Timer_Counter 为周期任务分频计数器，Int_Number 为中断号。

Int_Number	Delay_Counter	Timer_Counter	End_Time	Run_Time	SP_End	Prio	State	ID

图 3 硬件 RTOS 的任务控制块结构

硬件调度内核使用多级比较器，能确定当前时刻优先级最高的就绪任务，并将其信息通过接口总线发送到处理器。比较器采用相邻任务两两比较的方式。所以，2^n 个任务需要 n 级比较器，构成 2^n-1 个比较单元。多级比较器的结构示例如图 4。

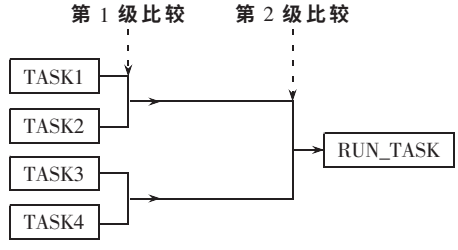

图 4 多级比较器结构示例

一个比较单元的实现代码如下：

```
process(Ready_TASK0_pro ,
                    Ready_TASK1_pro)
    begin
      if (Ready_TASK0_pro<
            Ready_TASK1_pro) then
        cmp_level3_01_pro<=
              Ready_TASK0_pro ;
        cmp_level3_01_id<=
              Ready_TASK0_id ;
      else
        cmp_level3_01_pro<=Ready_TASK1_pro ;
        cmp_level3_01_id<=Ready_TASK1_id ;
    end if ;
end process ;
```

每个比较单元以任务 ID 为索引，以优先级为比较内容，每次比较都将优先级高的任务送入下一级比较，经过 2^n-1 次比较，可以把 2^n 个任务中优先级最高的任务选出。硬件逻辑是物理并行执行的，所以与软件实现

的实时操作系统调度内核相比，硬件调度内核的执行速度快，而且这种优势在多任务虚拟并行和高时钟节拍的情况下将更加明显。

硬件调度内核逻辑结构如图 5 所示，处理器和硬件调度内核之间通过接口总线通信。硬件调度内核在调度时机进行任务调度，调度时机在以下几种情况产生：(1)周期任务周期时间到达，任务状态由等待变为就绪。(2)延时时间到达，等待任务状态由等待变为就绪。(3)外部中断到达，中断任务状态由等待变为就绪。(4)应用程序调用由硬件 RTOS 提供的系统 API 改变了任务状态或优先级。

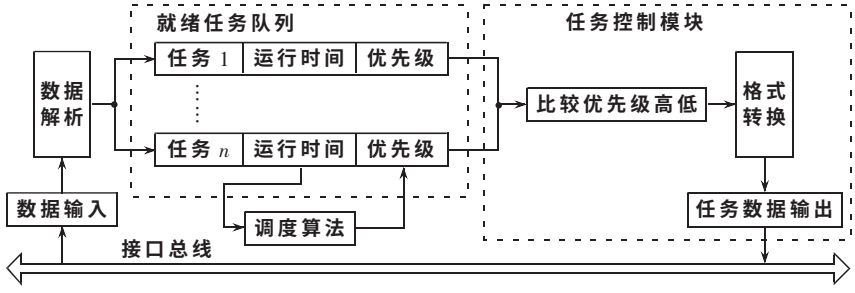

图 5 硬件任务调度内核逻辑结构

以上述第 4 种情况为例，描述调度执行过程如下：处理器在执行到应用程序调用的系统 API 时，将其转换成对应的命令和数据，经编码后传递到调度内核，触发硬件调度内核的一次调度时机。后者解析收到的数据，并执行调度操作，通过比较各任务优先级，计算出优先级最高的就绪任务。若该任务与当前处理器中正在运行的任务不同，则将该任务的 ID 号和堆栈终止地址利用通知中断回送给处理器。

2.2 中断管理模块的设计实现

普通任务控制块的中断号为 0，而中断任务的中断号为大于 0 的整数值，并与中断号对应的外部中断相关。中断管理模块使用 FPGA 的 I/O(输入/输出)管脚，每个管脚对应一个外部中断，一个或者多个外部中断对应一个中断处理任务。应用程序初始化阶段设定任务控制块中断号字段 Int_Number，能注册中断处理任务。在 FPGA 中建立以外部中断为敏感信号的守护进程，当外部中断到达时，该进程将中断任务置于就绪状态，其伪代码如下：

```
process(Task_Int_Number)              //外部中断为敏感信号
  begin
    if (Task_Int_Number'event and Task_Int_Number=
                                  0) then//下降沿触发
        Task_State <=1;               //任务置于就绪态
    else
        Task_State<=0;
    end if;
end process;
```

硬件调度内核以任务状态为敏感信号，当外部中断通过中断管理模块使中断任务就绪时，会触发调度内核执行一次任务调度。中断管理模块的逻辑结构如图 6 所示。为了保证外部事件的实时处理，应用程序可为中断任务设置高于普通任务的优先级。

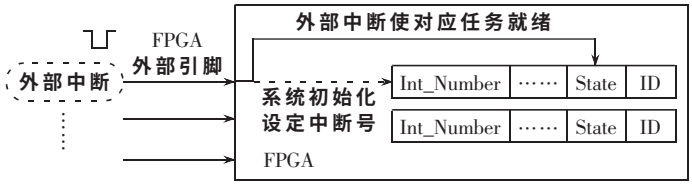

图 6 中断管理模块逻辑结构

2.3 定时器管理模块的设计实现

定时器管理模块以 FPGA 外部晶振为基准进行分频，创建多个定时器以设定任务周期。与定时器关联的任务称作周期任务。定时器管理模块在每个定时器的周期到达后，使对应的周期任务就绪，触发调度内核进行任务调度。

设定任务控制块中的 Timer_Number 注册周期任务。如图 7 所示，定时器管理模块实现：(1)任务周期运行。在应用程序设置并使能周期计数器 Timer_Counter 后，每当系统时钟到达，计数器减 1。当计数器为 0 时，触发对应的守护进程，使任务进入就绪态并重置周期计数器。任务状态的改变触发 FPGA 中的调度进程，执行一次任务调度。(2)任务延时。在应用程序设置并使能延时计数器 Delay_Counter 后，模块使任务变为等待状态。每次系统时钟到达，计数器减 1。当计数器为 0 时，触发对应守护进程，使任务进入就绪态，并执行一次任务调度。

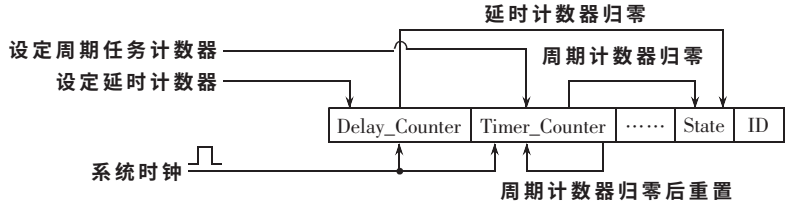

图 7 定时器管理模块实现机制

2.4 接口设计

接口总线的硬件由 32 位数据总线、5 位地址总线、1 个片选引线和 1 个通知中断引线构成。接口软件包括系统 API 和通知中断处理程序。

2.4.1 系统 API 的设计

系统 API 实现处理器中应用程序与硬件 RTOS 的交互。首先为任务调度内核、中断管理和定时器管理等各硬件模块分配处理器访问地址；然后，定义各模块使用的 API，确定其功能和转换后向硬件 RTOS 发出的数据格式及内容。

任务管理 API 包括：创建任务(Task_Creat)、阻塞任务(Task_Block)、改变任务优先级(Task_Change_Prio)和

改变任务状态(Task_Change_State)。

定时器管理 API 包括:设定分频值(Set_Timer_Fre)、使能周期任务(Set_Timer_Task)和延时(Delay)。

中断管理 API 包括:设定中断任务(Set_Int_Task)和等待中断(Wait_IRQ)。

下面以创建任务 API 为例,描述其伪代码和功能注释:

```
Task_Creat(ID,PRO,STATE) //创建任务,ID 为任务索引,
                         //PRO 为优先级,STATE 为起始状态
{
   Senddata=0001 :: ID : PRO : STATE ;      //拼装数据,
                                            //变成发送数据格式
   &TASKADD=Senddata ;     //将数据发送给硬件RTOS
}
```

2.4.2 通知中断服务程序的设计

实时操作系统中,每个任务都有堆栈空间,用于现场切换时保存各寄存器值和栈指针。其中,栈指针指向的堆栈起始地址和保存现场后的终止地址最为重要。保存当前运行任务的现场时,从其堆栈的起始地址开始执行入栈操作;而恢复将运行任务的现场时,从其堆栈的终止地址开始执行出栈操作。

任务堆栈在处理器使用的内存中分配。应用程序初始化时,将任务堆栈终止地址利用接口总线写入硬件调度内核维护的任务控制块中。每次调度结束后,若需要现场切换,硬件调度内核会将下一运行任务的 ID 和堆栈终止地址利用通知中断发送给处理器。通知中断处理程序首先将当前任务的寄存器保存在该任务以 SP_START 为起始地址的堆栈中,然后从接口总线读取下一运行任务的 ID 和堆栈终止地址,恢复该任务的现场并开始运行。图 8 显示了通知中断服务程序的执行过程。

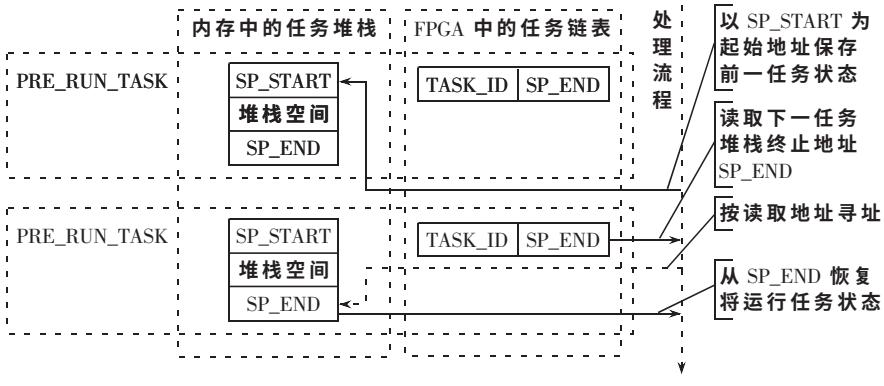

图 8 通知中断服务程序执行过程

3 实验与结果分析

对本文实现的硬件 RTOS,在时钟节拍为 10μs 时,分别测试在 4、8、16、32 和 64 个任务下、由定时器中断触发调度时机引起现场切换,所占处理器时间(系统开销)的情况。其结果与软件实时操作系统 μC/OS 进行比较,如表 1 所示。

表 1 不同任务数目下 μC/OS 与硬件 RTOS 系统开销的比较

任务数	4	8	16	32	64
μC/OS	3%	6%	11%	21%	42%
硬件 RTOS	2%	2.5%	2.7%	3%	3.2%

测试结果表明,硬件 RTOS 中,由定时器中断引起的系统开销不随任务个数的增加而显著变化,但与 μC/OS 相比,差值越来越大。

表 2 列出了任务 32 时,在 10μs、20μs、40μs、80μs 和 100μs 的时钟节拍下,硬件 RTOS 和 μC/OS 由定时器中断引起现场切换所占处理器时间的情况。

表 2 不同时钟节拍下 μC/OS 与硬件 RTOS 系统开销的比较

Time Tick/μs	10	20	40	80	100
μC/OS	21%	15%	10%	3%	2%
硬件 RTOS	3%	2.5%	2.2%	2%	2%

测试结果表明,随着时钟节拍降低,硬件 RTOS 的系统开销变化不明显,而与 μC/OS 相比,差值越来越大。时钟节拍很小时,硬件 RTOS 比 μC/OS 的系统开销低许多。

目前,硬件 RTOS 实现了任务调度内核等基本模块。今后的工作中,将进一步扩展其功能,增加以下几个部分:(1)任务间通信和同步。(2)内存管理。(3)支持更多的调度算法,实现调度算法可配置。(4)针对可在 FPGA 中实现的可配置处理器,用硬件实现上下文切换逻辑。

实时操作系统是嵌入式应用的核心,本文采用 FPGA 实现硬件实时操作系统,包括任务调度内核、中断管理和定时器管理等基本功能。硬件 RTOS 能降低处理器系统开销,提高其利用率,从而提高实时系统任务集合的可调度性,具有一定的研究和使用价值。

参考文献

[1] ADOMAT J, FURUNAS J, LINDH L, et al. Real-Time Kernel in Hardware RTU: A step towards deterministic and high performance real-time systems. In Proceedings of Eighth Euromicro Workshop on Real-Time Systems, 1996.

[2] LABROSSE J J. 嵌入式实时操作系统 μC/OS-II. 北京:北京航空航天大学出版社, 2003.

[3] MOONEY V, BLOUGH D M. A Hardware-Software real-time operating system framework for SOCs. IEEE Design and Test of Computers, 2002.

(收稿日期:2007-11-08)