文章编号:1001 - 9081(2005)11 - 2715 - 02

通信系统中大量定时器的设计与分析

邹仕祥
(解放军理工大学 通信工程学院,江苏 南京 210007)
(zoushixiang@126.com,zoushx@21cn.com)

摘 要:提出了一种用软件实现大量定时器的方法,由定时器管理模块(TMM)实现。基本原理是:TMM 设置 m 个定时精度,每个定时器精度有 n 个超时时刻,相同定时精度、相同超时时刻的定时器构成一个链表,TMM 设置 m 个循环指针 Pos[1..m],记录每个定时精度上次超时的链表位置。创建定时器时,根据定时时长选择合适的定时精度,计算将插入的链表位置,并插入对应的链表。循环指针 Pos[1..m]在系统时钟的驱动下按照各自的频率移动。当循环指针移动到某个位置,该位置所对应的链表上的所有定时器超时。由于插入定时器节点无需排序,并且批量定时器超时减少了比较次数,所以采用这种实现方法大大减少了系统用于定时器管理方面的开销。

关键词:定时器;定时精度;进程;管理算法
中图分类号: TP316;TP311.52 **文献标识码**:A

Development of a large number of timers in communcaition system

ZOU Shi-xiang
(*College of Telecommunication Engineering, University of Technology of PLA, Nanjing Jiangsu* 210007, *China*)

Abstract: A method to realize a large number of timers quickly by software was proposed, and the software was called Timer Management Module (TMM). TMM set m timing precisions and n time-out values for each precision. The timers with same precision and time-out value were linked together, TMM set m circular pointers, named Pos[1..m], which recorded the sequence number of the links, in which the timers had latest expired. When creating a timer, TMM first chose an appropriate timing precision according to the time-out value, and calculated the position, then insertd the timer node into the corresponding link. The circular pointers Pos[1..m], driven by system clock, moved around with their own frequency. When a circular pointer moved to a certain position, all timers in the corresponding link had already expired. By no sorting and reducing the comparison times, the algorithm cuts down the overhead of management timers greatly.

Key words: timer; timing precision; process; management algorithm

通信系统离不开通信协议,而定时器是保证通信协议正常运行的基本要素之一。通常通信协议用到的定时器定时精度要求不高,但数量可能比较大,特别是一些大型通信系统,如交换机、基站,某一时刻存在的定时器数量可能多达几千个。而硬件提供的定时器个数很少,操作系统提供的定时器要么数量有限、要么太多的定时器会降低系统性能,不能满足要求,必须通过其他方法解决。

本文描述了用软件实现大量定时器的方法。为方便,我们称实现和管理定时器的软件为定时器管理模块(Timer Management Module,TMM)。

1 几种大量定时器设计方法分析

在目前的通信系统中,硬件提供的定时器个数很少,一般只有几个,大量的定时器必须由软件提供。而软件方法只有两种:由操作系统提供或由应用程序提供。

1) 用操作系统提供定时器

操作系统提供的定时器优点是实时性好,定时精度高,但缺点很明显:首先,任何操作系统所能提供的定时器个数是有限的,有些操作系统无法提供多达几千个的定时器,如 Linux 只能提供 256 个定时器。其次,操作系统提供的定时器都是与中断相关联的,定时器超时处理部分或者全部包含在中断处理中。也就是说,大量的定时器会导致大量的中断处理,必然降低系统的性能。

2) 用应用程序提供定时器

应用程序提供的定时器在实时性和定时精度方面都不如操作系统提供的,但能满足大部分通信系统的要求,这是因为通信系统中除用于时钟同步的定时器外(这类定时器数量少,可用操作系统提供的定时器),大量的定时器是用于监视通信系统内部或者通信双方的行为是否正确,定时时长一般都在秒级以上,实时性和定时精度要求不高。我们所接触的一些通信软件代码,如 Trillium 公司的 SS7 协议栈代码、Radvision 公司的 H.323 协议栈代码以及 Telelogic 公司的定时器实现代码,都采用这种方法提供定时器。下面以 Telelogic 公司的定时实现代码为例,分析这种实现方法的优缺点。Telelogic 公司提供一种名叫 SDT 的通信协议开发工具,能将基于 SDL 语言的程序流程图直接转换为基于 C 语言的程序代码。SDT 提供了定时器的实现代码,下面分析其实现机制:

一个定时器用一个五元组表示:< Id, Timeout, Message, Parameter, Pid >。其中,Id 是定时器的内部标识(或称句柄);Timeout 是定时器超时时刻,由定时时长与创建时刻相加所得;Message 是定时器到期时定时器模块向进程回发的消息码,该值与定时器类型有关,是定时器管理模块自动生成的。Parameter 是参数体,它是一个字节流,由使用者在创建定时器时设置;Pid 是进程的进程标识。

收稿日期:2005 - 05 - 26;**修订日期**:2005 - 08 - 11
作者简介:邹仕祥(1972 -),男,安徽桐城人,讲师,硕士,主要研究方向:综合业务数字网.

SDT 用“单链”策略实现定时器,其算法是:令 L 是定时器节点链表,初始状态下 L 为空;创建定时器就是把定时器节点 T 按照到期时间从小到大顺序加到 L 中;释放定时器 T,就是从 L 中删去 T。判断一个定时器是否到期就是考查当前指向的定时器节点的 TimeOut 是否小于(含等于)当前系统时刻:如果大于,定时器没有超时,等以后再考查;否则,就执行这个节点上的定时器,并且将指针后移。

这个算法不消耗多余的空间,但存在明显缺点:1)每次增加一个新的定时器,在排序上要花费一定的时间,特别是对大型通信系统;2)对每个定时器的超时判断要花费一定的时间。

2 对应用程序提供定时器方法的改进

2.1 逻辑结构

与 SDT 的定时器逻辑结构类似,改进的定时器也用一个五元组表示:<Id,Length,Message,Parameter, Pid>。其中,Id 是定时器的内部标识(或称句柄),由定时器管理模块使用,进程创建一个定时器时,管理模块返回这个内部标识,进程关闭一个定时器时,要把这个标识传给管理模块。Length 是定时时长,单位是秒或毫秒,这取决于调用的创建定时器函数。Message 是定时器到期时定时器模块向进程回发的消息码。Parameter 是回发消息的参数体。Pid 是进程的进程标识。

改进的定时器结构与 SDT 的定时器结构区别在于:前者的定时器结构中记录的是时长,后者记录的是超时时刻。

2.2 链表结构

改进的方法采用“双链”策略实现定时器。其实现机制是:

设置 M 个物理定时时长 PT_1, PT_2, …, PT_M,它们之间相差 16 倍,即 PT_1 是 1ms,PT_2 是 16ms,PT_3 是 256ms,以此类推,每一个 PT 称为一个定时精度。

令 L 是定时器节点链表。把 L 作划分:$\{L_1, L_2, \cdots, L_M\}$,每个 L_K 都是与 PT_k 相关的定时器集合,称为同一定时精度集;继续对 PT_k 作划分:$\{L_{K,1}, L_{K,2}, \cdots, L_{K,n}\}$,每个 $L_{k,i}$ 中所有定时器都具有相同的到期时刻。

设置 M 个循环指针 $Pos[1..M]$;它们的初值都是 1。

创建一个定时器的算法如下:

```
Begin
  构造一个定时器节点 T;
  填写 T 的要素,如回发消息和参数等;
  令 T 的定时时长为 LEN;
  查找 T 的最佳计量单位 k,即 LEN > PT(k-1) * n,同时 LEN <= PTk * n;
  j: =(LEN/ PTk + Pos[k]);
  if (j > n) j: = j - n;
  j 就是 T 应处的划分子集号;
  把 T 加入 Lk, j;
  返回 T 的句柄;
End
```

遍历 L 的算法就是:

```
Begin
  For k: =1 to M Do
  if PTk 已到期 Do
  Begin
    Pos[k]: = Pos[k] +1;
    if (Pos[k] > n) Pos[k]: =1;
    现在,Lk, pos[k] 上所有定时器已到期;
    For ALL Ti IN Lk, pos[k] Do
    Begin
      向 Ti 的占用者回发消息;
    End
  End
End
```

在实现时我们选择 M = 8,n = 256,这样,最小定时时长是 1ms,最长定时时长是 $256 * 16^7$ ms,约为 19 089 小时,这样的跨度足够了。

3 与具体实现相关的一些问题

物理定时器的操作 有多种方法可以实现,由于我们采用的硬件平台是 x86 系列的,所以本方案用 RDTSC 指令实现物理定时器,它只与 CPU 频率有关,与操作系统无关;PowerPC 系列的 CPU 有类似的指令。

遍历定时器的时机 时钟中断或循环查询,在实现时我们选择循环查询。

定时器句柄的管理 与内存管理类似,一个定时器节点就是一个结构,多个节点就形成一个结构数组,该数组的下标就是定时器的句柄。句柄的申请、释放与内存管理中句柄管理一样。定时器模块所需要的存储空间是准静态分配的,即在系统初始化时一次性向内存申请分配,以后工作时不再追加,也不释放。

链表的类型 本方案拟采用单向链表。虽然双向链表的删除操作要比单向链表耗费低,但它的基础开销大,所以当链表不是太长时,双向链操作不一定比单向链耗费低。

检查出现“太老”的定时器 在一个精度集中一个定时器的最大定时时长是 $n *$ 计时粒度,我们把定时时长不大于 $(n-1) *$ 计时粒度的定时器定义为正常定时器,把定时时长等于 $n *$ 计时粒度的定时器定义为太老的定时器。只要在创建定时器时限制定时时长,系统不会出现太老的定时器。出现了这些定时器,说明有些进程没有及时释放定时器,这是一种逻辑错误,需要让调试人员知道。这样,遍历定时器的算法就修正为:

```
Begin
  For k: =1 to M Do
  if PTk 已到期 Do
  Begin
    现在,Lk, pos[k] 上所有定时器都是太老的定时器,向操作员报告;
    Pos[k]: = Pos[k] +1;
    if (Pos[k] > n) Pos[k]: =1;
    现在,Lk, pos[k] 上所有定时器已到期;
    For ALL Ti IN Lk, pos[k] Do
    Begin
      向 Ti 的占用者回发消息;
    End
  End
  L1: 返回各种定期计时标志;
End
```

语句 L1 中的定期计时标志实际上是一个 32 比特的整数,每一位对应着一种定期计时器,它为系统的各个模块提供周期性的定时(前面所描述的定时器提供一次性定时)。

4 对改进方法的分析

从上述定时器的实现算法可以看出:1)向链表中插入新增的定时器无需排序,插入点位置由定时时长直接计算得到;2)同一节点上的所有定时器具有相同的到期时刻。在判断出该节点到期后,其上所有定时器都到期,无需对该节点上的每个定时器都进行到期判断。所以,改进的定时器实现方法大大缩短了系统用于定时器管理方面的开销。

(下转第 2719 页)

送 FIFO 写数据,则由空闲状态进入写状态,同时产生激励写信号到发送 FIFO。若只写单个数据下一周期就返回到空闲状态,否则若是突发写则连续处于写状态,写完所有数据后再返回空闲状态。

读状态　接口逻辑从 PCI 核的信号译码出主机准备从接收 FIFO 读数据,则由空闲状态进入读状态,同时产生激励读信号到接收 FIFO。若只读单个数据下一周期就返回到空闲状态,否则若是突发读则连续处于读状态,读完所有数据后再返回空闲状态。

状态机(b)用来控制收发器的接收模块往接收 FIFO 的"写"时序,它也由三个状态组成:空闲状态、写状态和结束状态。状态及转换关系如下:

空闲状态　当接收数据已准备好且接收 FIFO 不满时进入写状态,否则继续处于空闲状态。

写状态　写接收 FIFO 状态。写信号只持续一个周期便自动进入结束状态。

结束状态　成功写入数据后的状态,下一周期便返回到 S_IDLE 状态。该状态的自身循环是屏蔽冗余的接收数据准备好状态,因为接收模块不会有突发模式,每次只操作一个数据。

状态机(c)用来控制收发器的接收模块从接收 FIFO 的"读"时序,它只由两个状态组成:空闲状态和读状态。状态及转换关系如下:

空闲状态　当发送 FIFO 不空且发送模块就绪时下就进入写状态,否则继续处于空闲状态。

读状态　读发送 FIFO 数据时的状态,读信号只持续一个周期便返回空闲状态。

2.4　串行收发器

高速串行通信要求收发器必须具有高速工作频率,Xilinx 公司推出的高端产品如 Virtex II Pro 系列产品就内嵌 RocketIO 多路吉比特串行收发器 MGT(Multi-Gigabit Transceiver),其速率高达 3.125Gb/s。本文采用 Spartan II 芯片自行设计收发器,其速率虽低但原理一样,同样验证了接口的可行性。

至此,高速通信中的 PCI 总线接口设计已完成,它在接口逻辑和状态机控制下,实现了 PCI 核与用户逻辑间紧密连接,收发器自动监测并接收串行数据写入接收 FIFO,并且自动从发送 FIFO 中取出数据串行发送出去。

3　硬件编程与验证

Spartan-II 200 PCI 开发板为基于 Xilinx 的 Spartan-II FPGA 系列芯片开发设计 PCI 总线接口提供一个良好的仿真环境。板中带 200 000 门的 Spartan-II 芯片(XC2S200-6FG456C)具有 456 个管脚。极高的门密度和大量的用户 I/O,允许在低成本的 FPGA 里实现完整的系统解决方案。Spartan-II 是 2.5vSRAM 工艺 FPGA,带有 14 个 RAM 块,2 352 Slices,每个RAM块容量是4kbit,利用RAM可以用来构造接收 FIFO 和发送 FIFO。另外,基于 Web 配置工具配置 PCI 核工作在 5V 工作电压的环境,以 32bit 方式进行数据传输。

对复杂数字系统而言,传统设计方法已不适用。本系统采用硬件描述语言并按照 FPGA/CPLD 一般设计流程[5] 进行设计,使用 FPGA 设计工具 ISE6.2 进行项目管理。首先是系统的设计输入及编写测试基准(Testbench),利用 ModelSim SE 6.0 仿真软件进行仿真;然后使用 Synplify 7.6 综合工具进行综合,综合后再进行综合后仿真;接着实现(具体再分为转换、映射、布局布线三个过程),并进行时序仿真;最后生成 mcs 文件及 bit 位文件。利用专用下载工具通过电缆把 mcs 文件写到 Spartan-II PCI 开发板上的 PROM 中。这样当板卡插入 PCI 槽上电后,PROM 中的数据将自动导入到 FPGA 中,系统可以开始工作。另外,专门为此 PCI 卡编写了在 Windows 下的驱动程序以及用户界面的测试程序。

这里给出 PCI 接口逻辑的工作时序,如图 4 所示。以读写单个数据为例进行说明。当主机发出写数据命令时,接口逻辑译码出写操作并协同图 3 中的状态机(a)产生向发送 FIFO 写数据的激励信号 FSM_Write,写入后发送 FIFO 的写指针增 1,状态标志 TxFifoEmpty 由高变低,同时记数器及时记录当前 FIFO 中的数据个数。同理,当主机发出读数据命令时,接口逻辑译码出读操作并协同图 3 状态机(a)产生从接收 FIFO 读数据的激励信号 FSM_Read,读出后接收 FIFO 的读指针增 1,记数器减 1,及时记录当前 FIFO 中数据个数。

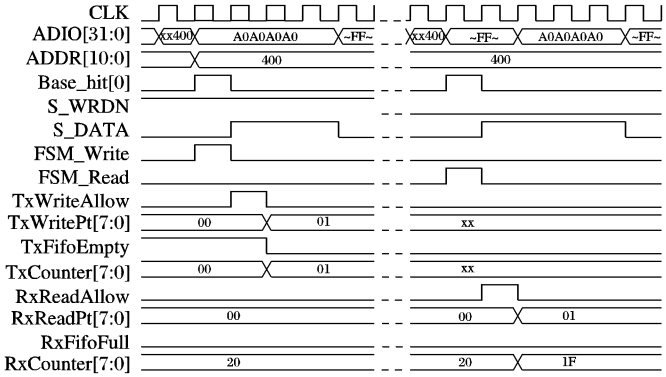

图 4　PCI 接口的工作时序图

参考文献:

[1] SHANLEY T, ADERSON D. PCI 系统结构[M]. 第 4 版. 刘晖,等译. 北京:电子工业出版社,2000.

[2] 雷艳静,苗克坚,康继昌. 高速通信网卡中 PCI 接口的研究与实现[J]. 计算机应用,2005,25(1):173 - 175.

[3] 苗克坚,陆尧,车立昌. 微机 PCI 总线接口的研究与设计[J]. 航空计算技术,2000,30(2):49 - 51.

[4] Xilinx Corp. LogiCORETM PCI Design Guide Version 3.0[EB/OL]. www.xilinx.com,2003 - 10 - 27.

[5] 王诚,薛小刚,钟信潮. FPGA/CPLD 设计工具——Xilinx ISE5.x 使用详解[M]. 北京:人民邮电出版社,2003.

(上接第 2716 页)

同时,我们应看到,上述实现定时器的算法在定时时长处理上有一定的误差。最大误差出现在:应用程序需要定时 271ms,实现时实际定时时长为 256ms,误差约为 5.9%。如果定时时长平均分布,那么平均误差约为 2.5%。由于通信系统中定时器的定时时长都有一定的取值范围,也就是说,值是可以调整的,所以这样的误差不会影响系统的正常运行。

5　结语

通过上述方法,不仅解决了硬件和操作系统提供的定时器不足的问题,而且较"单链"实现方法,大大提高了遍历定时器链表的速度;定时器异常处理,为软件调试提供了一种调试手段。我们在研制移动交换机 MSC 和基站 BSS 时,都采用了这种方法,效果良好。

参考文献:

[1] 汤子瀛,哲凤屏,汤小丹. 计算机操作系统[M]. 西安:西安电子科技大学出版社,1996.

[2] 严蔚敏,吴伟明. 数据结构:C 语言版[M]. 北京:清华大学出版社,1997.

[3] 谭浩强. C 语言设计[M]. 第 2 版. 北京:清华大学出版社,1999.