文章编号:1001 - 2486(2005)02 - 0075 - 05

通用计算机实时仿真技术*

李兴玮,张卫华,潘玉林,黄柯棣
(国防科技大学 机电工程与自动化学院,湖南 长沙 410073)

摘 要:通用计算机由于各种相互兼容的标准硬件及丰富的应用软件支持而获得了广泛的应用,但通用计算机往往运行多任务的操作系统,这种通用操作系统一般不能满足半实物仿真的实时性要求。讨论了通用计算机半实物仿真的实时时钟获取方法,重点研究了通用操作系统的帧时间不稳定问题。提出了保证实时仿真帧时间的几种方法,并建立了一个具体的通用仿真计算机系统。试验结果表明,基于通用计算机的实时仿真是可行的。

关键词:半实物仿真;实时仿真;实时;时间计数器;帧时间;线程
中图分类号:TP391.9 文献标识码:A

Real-time Simulation Technology Based on the General Computer

LI Xing-wei, ZHANG Wei-hua, PAN Yu-Lin, HUANG Ke-di
(College of Mechatronics Engineering and Automation, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The general-purpose computer, supported by all kinds of compatible standard hardware and plenty of application softwares, has been most wildly used in the world. However, the operating system running on it is always the multi-tasks OSs, such as windows 9x/2000 and Linux, which cannot meet the need of the real-time simulation in a general way. Firstly, the acquiring of high-resolution real-time clock is discussed. Secondly, the stability of frame time under the general-purpose OS environment is studied at length. Several methods used to carry out the real-time simulation on the general-purpose computer has been presented, and an actual general-purpose simulation system also established. The experiment results show that the real-time simulation based on the general computer is feasible.

Key words: hardware-in-the-loop; real-time simulation; real-time; time counter; frame time; thread

随着计算机技术的迅猛发展,专用仿真计算机系统已逐渐跟不上仿真技术发展与应用的潮流[1]。一方面,由于 VLSI 工艺、精简指令集计算机(RISC)技术、分布/并行处理技术、仿真算法及软件的迅速发展,高档微机及工作站等通用计算机在速度、精度及存储容量等方面已经为利用通用机进行实时仿真打下了基础;另一方面,虽然仿真专用机在半实物仿真中仍然占有重要地位,但因其应用面窄、生产数量少,导致价格居高不下。随着通用计算机性能的迅速提升,建立基于通用计算机的仿真系统逐渐成为可能。但是,由于半实物仿真对仿真计算机的特殊要求,通用计算机往往无法直接应用于对时间要求苛刻的半实物仿真。利用通用计算机进行实时仿真,必须在硬件和软件两个方面对通用计算机进行一定的改造,尤其需要在通用计算机实时仿真系统结构,以及实时操作系统中的实时任务调度、内存管理和实时 I/O 通信等方面做大量实际、细致而富有挑战性的研究工作[2]。

1 实时时钟的获取

为了控制仿真计算机与外接实物的实时同步运转,常常需要进行各种复杂而精确的时间控制。因此,仿真计算机必须有可编程的实时时钟,其分辨率在微秒级,要求可控、可数、可读/写,以便为帧时间测量和外界实物的实时同步控制与通信提供有力的手段。高精度时间计数器不仅是仿真软件实现实时

* 收稿日期:2004 - 10 - 14
作者简介:李兴玮(1969—),男,副教授,在职博士生。

控制及实时数据采集的关键技术,也是实现帧时平衡、进程调度、时间同步和实时通信技术以及仿真控制框架的基础。然而,无论是标准的C运行时库,还是常规的Win32 API,均只能提供最小为1ms的时间控制精度,而实际上由于系统中断8以及Windows消息传递机制的影响和制约,其最小时间控制精度仅为55ms[3](即18.2次/s)。一般的时间计数器实现方法,不仅精度低,而且控制方法单一,难以满足半实物仿真对复杂时间控制算法的需求。为了获得更高的时间控制精度,可采取两种做法:

• 使用外部时钟。该法不仅需要增加高额的硬件费用,而且还需要进行复杂的中断处理程序(DOS环境)或设备驱动程序(Windows环境)编程。

• 改变时钟0的计数初值以获得远快于标准系统时钟频率的计数频率。但是,该法取决于CPU、系统要求的时间以及中断8处理程序。随着时钟0中断率的提高,系统很容易在运行其它事件时用完时间,甚至丢失时钟滴嗒,并且在Windows环境下,仍然需要进入ring0级编程[4]。

实际上,两种方法采用的都是中断处理的控制方法,不利于实现半实物仿真所需的各种复杂的时间控制算法。因此,都具有很大的局限性。

1.1 基于时钟2的高精度时间计数器

每个PC系统至少包含一个8254可编程时钟或等价芯片,该芯片包含三个独立的16位时钟计数器。不同的时钟通道具有不同的用途。时钟0用于产生系统中断,以保持系统时间和执行专门的定时服务;时钟1用于主板DRAM电路刷新,对该时钟编程可能会造成DRAM刷新操作停止,从而导致所有DRAM内存内容丢失,因此强烈建议不要使用时钟1;时钟2用于一般的应用程序而不会对系统产生影响——这正是利用时钟2实现高精度时间计数器的原因所在[5]。

每个时钟计数器的输入频率均为固定值,通过设置不同的计数初值,可以获得不同的时钟脉冲输出频率。在时钟2的计数过程中,可先读取时钟2的计数值,计算出两次读操作之间所经历过的时钟脉冲个数,再除以固定输入频率,即得到两次读操作之间的时间间隔。如果想对时钟计数器进行操作,必须对其工作模式进行相应设置。8254为每个时钟通道提供了六种操作模式,但是由于芯片硬件连接的问题,某些模式对于特定的通道并不适用。时钟2以二递减计数,因此,在一个计数周期内,两次读数之间所经历的时钟脉冲个数是前后两次计数值之差的一半。该法在现有CPU速度下,其精度可达到10~30μs,并且适用于DOS和Windows9x操作系统。

1.2 基于CPU时间戳计数器(TSC)的高精度时间计数器

在Pentium系列及其兼容CPU的内部均有一个64位的二进制时间戳计数器[6],该计数器按照CPU的主频计数,即每来一个CPU时钟信号,TSC计数加1。因此,只需知道TSC计数值和CPU时钟信号周期(即CPU主频),两者相乘就可计算出从系统启动以来的时间。通过读取TSC的计数值,可以实现精度为纳秒级的时间控制方案。例如:某CPU的主频为600MHz,则最小计时精度约为1/600M = 1.667ns。CPU时钟信号周期的准确与否直接决定了TSC计时方法的准确与否。为了获得准确的CPU时钟信号周期,可利用时钟2延迟一段确定的时间,然后通过延迟前后的TSC计数值计算CPU的时钟信号周期。该方法的精度完全取决于CPU速度,鉴于目前几乎所有的CPU速度都在100MHz以上,因此其计时精度至少在10ns以下。它适用于所有Windows操作系统,但不适用于DOS操作系统[7]。

2 帧时间的稳定性

如果能用Windows操作系统进行对时间要求苛刻的半实物仿真,则不仅可节省十分有限的经费,而且将给广大的用户带来极大的使用方便。这就提出了在通用操作系统(非实时操作系统)下保证半实物仿真的仿真帧时间问题。

图1是在Pentium Ⅲ 933计算机上进行数学仿真时,基于Windows操作系统采集到的某武器系统作战飞行模型的仿真帧时间示意图。由图1可知,该武器系统的作战飞行模型解算时间小于0.20ms,当有系统开销时的最大仿真帧时间则达到0.55ms。仿真帧时间在一个较大的范围内波动,原本解算时间小于0.2ms的作战飞行模型却根本不能满足用0.5ms的仿真帧时间进行半实物仿真,这就是Windows操作系统下仿真的帧时间不稳定问题。

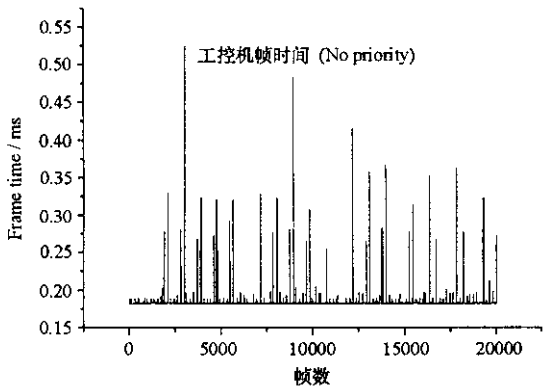

图 1 帧时间(缺省优先级)
Fig.1 Frametime(Normal)

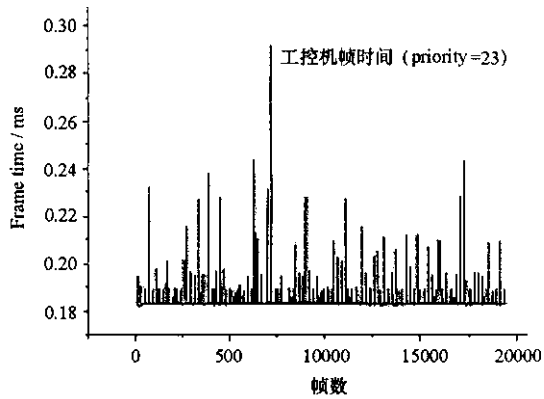

图 2 帧时间(优先级为 23)
Fig.2 Frametime(priority = 23)

考察图 1 可知,该武器系统的仿真模型解算时间之所以存在一定的波动,主要是由于计算机操作系统存在一定的系统开销,或者说,Windows 操作系统的实时性能不佳。Windows 操作系统的任务调度策略是抢先式的[8],操作系统何时调度何任务对用户的应用程序来说是透明的,因而,是不可预测的。Windows 操作系统在调度这些对应用程序透明的任务时,是把仿真任务当成普通的应用程序来看待的,它不知道仿真任务具有实时性的要求(即使知道,有些系统调度也是必需的),按照抢先式的任务调度策略,可能剥夺仿真任务的服务权利,转去为别的操作服务,服务完毕再回来为仿真任务服务,这就导致了仿真任务的帧时间不稳定。

2.1 采用实时操作系统

半实物仿真要求仿真计算具有实时性。为了达到实时性的要求,在计算量已知并选定了仿真算法后,首当其冲的办法就是提高 CPU 的计算性能。但是,实时仿真并不等同于高速计算,在 CPU 速度固定的情况下,操作系统的实时特性对半实物仿真的实时性起着决定性的作用。于是,通常的解决办法就是采用实时操作系统。

现在较为流行的实时操作系统的特色各异。Ctask、pDOS 等由于继承了 DOS 的单任务和不可重入特性,因而只能满足简单的实时应用;Digital UNIX 由于继承了 UNIX 分时操作系统的特点,因而只能适应有限的实时需求;VxWorks 虽然可满足广泛的实时系统的需求,但价格昂贵。而且,这些实时操作系统都存在共同的问题,即市场占有率低、应用开发环境及支持软件缺乏。

2.2 采用 RTLinux 操作系统

Linux 操作系统是基于 PC 机的免费的类 UNIX 开放式操作系统,并采用了源代码公开的发布策略。由于世界各地的 Linux 用户和开发者的不断努力,Linux 已成长为高稳定性的、性能优异的操作系统。新墨西哥理工学院开发的基于标准 Linux 的具有硬实时特性的 RTLinux 是一个理想的实时操作系统。实际上,RTLinux 已经成功地应用于从航天飞机空间数据采集、科学仪器测控到电影特级图像处理等广泛的实时环境下。但是,目前 RTLinux 存在的问题有[9]:

- 标准的 Linux 设备驱动程序中经常关中断。这对某些操作是必需的,但在 RTLinux 中无法真正禁止中断的发生,因此有可能造成误操作;此外,驱动程序可以被实时任务抢断,因此为了在 RTLinux 下充分发挥设备的实时特性,相应的驱动程序必须以实时任务的形式重写。
- RTLinux 下的实时任务和普通任务进程之间通信只能通过 RT-fifo,这种机制的缺点在于数据通信速度较慢,且会产生缓冲溢出等问题。
- 应针对具体的实时应用环境探讨最优的实时调度策略,并开发相应的测试和分析工具。
- 应分析 RTLinux 的软实时特性,研究如何实现 QOS(Quality of service)保证,以进一步广泛支持不

同的软实时应用。

2.3 改善 Windows 操作系统的实时性

能否对 Windows 这个通用(非实时)操作系统加以调整,在一定程度上加以抑制甚至完全消除对应用程序透明的操作,改善它的实时性能,并使其成为能够满足半实物仿真实时性要求的操作系统呢?回答是肯定的。

为了保证半实物仿真的实时性,可以采用提高仿真任务优先级的办法。Windows 操作系统下的任务优先级共有 32 级。0 级为系统级,专为清零页面保留;1 至 15 级为可变级;16 至 31 级为实时级。在 Windows 操作系统下,可利用 W32 API 的调度函数 SetPriorityClass 和 SetThreadPriority 将仿真任务的优先级提高。函数 SetPriorityClass 负责设置进程的优先等级,函数 SetThreadPriority 则负责设置线程的优先等级。图 2 是将仿真任务的优先级提高到 23 时,在 Pentium Ⅲ 933 计算机上采集到的该武器系统作战飞行模型仿真帧时间示意图(各坐标含义同前)。由图 2 可知,尽管该武器系统的模型解算时间依然有一些波动,但波动范围已降至 0.18ms ~ 0.30ms。由此说明,提高仿真任务优先级,确实可以在一定程度上保证半实物仿真的实时性。

3 内存管理性能的改善

为了保证半实物仿真的实时性,还可以将仿真任务锁入内存,禁止系统调页和进行进程的交换。如果仿真过程中需要存储大量的数据,特别是要与其它进程共享大量的数据时,采用内存映射文件是很好的选择。内存映射文件可以用来保留一个地址空间的区域,并将物理存储器提交给该区域。这里,物理存储器来自一个已经位于磁盘上的文件,而不是系统的页文件。一旦该文件被映射,就可以访问它,就像整个文件已经加载到内存一样。具体使用时,可首先通过 CreateFile()函数打开一个文件内核对象,这个对象标识了磁盘上将要用作内存映射文件的文件。然后,通过 CreateFileMapping()函数创建一个文件映射内核对象以告诉系统文件的尺寸以及访问文件的方式。再用 MapViewOfFile()函数将文件映射对象的全部或部分映射到进程的地址空间。图 3 是采用内存映射文件进行数据存储时的帧时间结果。

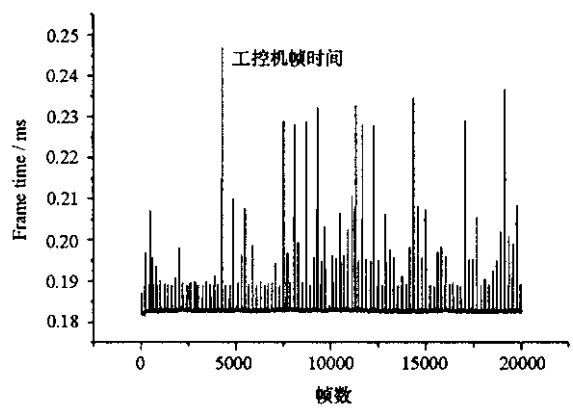

图 3 帧时间(内存映射文件)

Fig.3 Framtime (memory mapping file)

4 实时接口驱动技术

实时仿真应用是由真实世界事件驱动的,计算机的运转必须与真实事件同步,这种同步运转除计算机必须有足够的速度保证外,还必须有强大的 I/O 能力的保证。实时性要求客观事件数据必须被同时采集、分析和显示,并且用确定的形式来响应。这种响应带有同时性要求,至多允许有微妙级的时间偏差,要做到这一点必须配备强大的同步、并行 I/O,具有同时采集、同时转换、并行传输的能力。

在有与高速 I/O 传输相匹配的高频宽带的 I/O 总线支持下,可采用并行转换的 A/D 和 D/A 板卡实现与模拟设备的连接,采用数字并行 I/O 控制器和数字串行 I/O 控制器与数字部件进行连接,其它特殊 I/O 接口控制部件可通过基于多线程的串口、并口直接通讯。此外,也可由 Ring3 级(用户级)代码进入 Ring0 级(最高级)代码[10],以获得系统资源的最高访问权限。

5 仿真数据的实时显示

在很多实际的仿真应用中,需要实时显示一定的数据。事实上,可以依靠系统的多线程、抢先多任务机制,将实时显示功能置于不同于仿真主线程的单独线程中实现,然后通过优先级调度、线程同步等机制保证仿真主线程不被耗时的显示线程所耽误。此时,仿真主线程的执行和休眠可以采用定时器或

中断调度等方式控制,而显示线程的调度则主要有两种策略:一是将显示线程的优先级设置成低于仿真主线程的优先级,显示线程将始终处于等待状态,只有在仿真主线程休眠后才能获得 CPU 时间片执行,这实际上是通过 Windows 的优先级调度机制来完成线程的同步;二是使用同步对象,在仿真主线程每次休眠之前激活同步对象,使其变为'发信号'状态,通过同步对象唤醒显示线程执行。

6 应用举例

图 4 是基于普通微机建立的通用实时仿真计算机系统 KDRTS。KDRTS 的硬件系统采用单机结构,仿真机为一台高配置 Pentium Ⅳ机器,该机不仅担负着编译仿真源程序、启动仿真运行、运行过程中提供交互控制等任务,还负责把中间代码编译成可执行目标码、执行仿真目标程序、接收 I/O 通道送来的实物数据、进行半实物实时仿真。

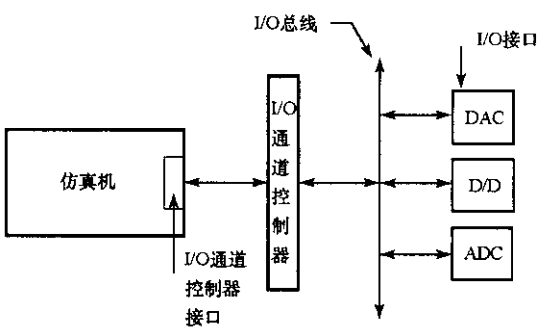

图 4 KDRTS 体系结构
Fig.4 The architecture of KDRTS

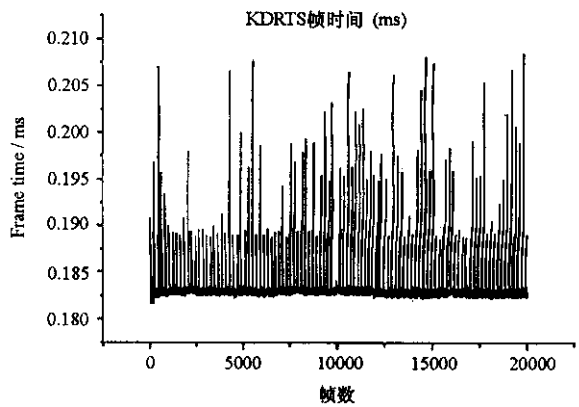

图 5 帧时间 (KDRTS)
Fig.5 Framtime (KDRTS)

通过采用前面提到的一些措施,基于普通微机建立了通用实时仿真计算机系统 KDRTS。图 5 是在 KDRTS 上实测的某制导武器系统半实物仿真试验的帧时间。由图可知,基于 KDRTS 用 0.5ms 的仿真步长对该武器系统进行半实物仿真是完全可行的。

7 结 论

通过成功解决通用计算机半实物仿真的一些关键技术问题,通用操作系统的实时性能得到了明显的改善。通用计算机的操作系统开销(包括任务调度、内存整理、消息处理等不可预测性的任务)被压缩到一定程度,从而使仿真帧时间被压制在狭小范围内波动,能够保证半实物仿真帧时间的实时性要求。把这个非实时的通用操作系统近似地看成实时操作系统,可以在通用计算机下进行半实物仿真。

参 考 文 献:

[1] 李兴玮,张卫华,黄柯棣. PCRTSim:Windows 平台下的实时仿真计算机系统[J].计算机工程与设计,2004(5).
[2] 李兴玮,黄柯棣. 通用操作系统下半实物仿真的帧时间研究[A]. 仿真计算机与软件专业委员会 2003 年学术会议论文集[C]. 厦门,2003.
[3] Gilluwe F V. PC 技术内幕[M]. 精英科技,译. 北京:中国电力出版社,2001.
[4] Solomon D A,等. Windows 2000 内部揭密[M]. 詹剑锋,等,译. 北京:机械工业出版社,2001.
[5] Yuan F. Windows 图形编程[M]. 英宇工作室,译. 北京:机械工业出版社,2002.
[6] Petrone D J,Stackhouse M D. PC-Based Control Goes Real-time Control[J]. Engineering,April 1998.
[7] Microsoft Corporation. Real-time Systems with Microsoft Windows CE 2.1[R]. Microsoft Technology Brief,2000.
[8] Microsoft Corporation. Real-time Systems and Microsoft Windows NT[R]. Microsoft Tech Brief,1995.
[9] 邢国良,等. 基于 Linux 的实时操作系统的分析与研究[J]. 小型微型计算机系统,2001(8).
[10] 张维铭. 使用 VtoolsD 开发 Windows95 虚拟设备驱动程序[J]. 中国计算机用户,1997(12).