导引头实时系统仿真架构设计

韩丰娟 丛潇雨 郭山红 付姣姣
(南京理工大学电子工程与光电技术学院 江苏 南京 210094)

摘 要 提出一种基于 RTX + Windows + 反射内存网的导引头实时系统仿真架构。通过 RTX 保证系统的实时性,在 RTX 下采用多线程并行方式降低信号处理时间;通过与 Windows 间的共享内存实现系统友好的人机交互;以反射内存卡为传输介质,保证数据传输的实时性。经过测试,该架构可以真实地仿真出导引头系统的整个工作过程,信号处理时间控制在 ms 级,数据传输速度超过 800 Mbit/s,线程切换时间与响应延时小于 13 μs。

关键词 导引头 系统仿真 实时 RTX 反射内存

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.06.015

DESIGN OF REAL-TIME SYSTEM SIMULATION ARCHITECTURE FOR SEEKER

Han Fengjuan Cong Xiaoyu Guo Shanhong Fu Jiaojiao
(*School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, Jiangsu, China*)

Abstract A real-time system simulation architecture for seeker was proposed based on RTX + Windows + reflection memory network. This architecture guaranteed the real-time performance of the system through RTX, and reduced the signal processing time by adopting multi-threaded parallel mode under RTX. The shared memory between RTX and Windows realized the friendly human-machine interaction. The real-time data transmission was ensured by using reflection memory card as transmission medium. After testing, it is found that the architecture can simulate the whole working process of seeker system in real-time, controlling the signal processing time at the level of millisecond. The data transmission speed is over 800 Mbit/s and the thread switching time and response delay were less than 13 μs.

Keywords Seeker System simulation Real-time RTX Reflection memory

0 引 言

在导引头系统研制过程中,仿真验证占有非常重要的地位[1],为了能更接近于真实的作战场景,近年来导引头系统仿真对实时性的要求越来越高。目前实时系统仿真难于实现的原因主要有两方面,一方面是仿真多基于 Windows 操作系统,虽然 Windows 拥有强大的图形化接口与众多第三方硬件的支撑,但其存在着延迟不确定、线程优先级倒置、线程调度机制不确定、IO 设备访问受限、定时器精度差等诸多问题,很难满足实时性要求。另一方面复杂的仿真系统多为多机互联的拓扑结构,目前常用的通信机制如红外、无线、USB 等,很难实现多机间的实时通信。

为了提高 Windows 系统的实时性,满足硬实时系统严格的响应时间要求,本文采用了美国 IntervlaZero 公司的 Windows 操作系统实时性解决方案 RTX[2]。该方案的实时性主要体现在以下三个方面:第一,相比于 Windows 下毫秒级的定时精度,RTX 下可达到 100 ns[3-4];第二,RTX 能独立地进行中断管理,而不受 Windows 干扰[5],可以实现对中断的实时响应。第三,RTX 对线程的控制透明、灵活,拥有更多的线程数和更高的优先级,线程间切换时间小于 10 μs,能够大幅优化导引头回波产生和信号处理的时间,提高系统

收稿日期:2018 - 12 - 25。江苏省自然科学基金项目(BK20180457)。**韩丰娟**,硕士生,主研领域:雷达信号处理,软件系统仿真。**丛潇雨**,博士。**郭山红**,副研究员。**付姣姣**,硕士。

实时性。在满足实时性的同时,RTX 可以通过共享内存的方式与 Windows 实现通信,从而保留了对 C/C ++ 标准模板库与图形化交互界面的支持。为了解决多机间通信同步的问题,本文采用了由反射内存卡 + 光纤 Hub 卡组成的反射内存网通信机制,它是一种实时网络[6],实现了基于总线公共存储策略的硬件级数据同步传输,具有速度快、通信协议简单、传输纠错能力强等优点。

基于以上的分析,本文提出了基于 RTX + Windows + 反射内存网的解决方案,完成了一种多机互联的导引头实时系统仿真架构设计,实现了对实时性和友好交互的兼顾。

1　导引头实时系统仿真架构

如图 1 所示,本文导引头实时系统仿真分为六个模块,分别是作战想定人机接口、导引头回波产生、导引头干扰产生、导引头接收机 1、导引头接收机 2、数据融合与场景演示。每个模块所在的计算机构成分布式控制节点,各节点通过反射内存卡在光纤 HUB 上实现互连,组成了一种星型结构的反射内存网络。整个系统在 RTX 下进行回波产生、干扰产生、信号处理、数据融合,在 Windows 下进行可视化展示,RTX 与 Windows 通过共享内存进行通信,各个模块在中断机制的控制下通过反射内存网实现同步数据传输,进而实现实时仿真。

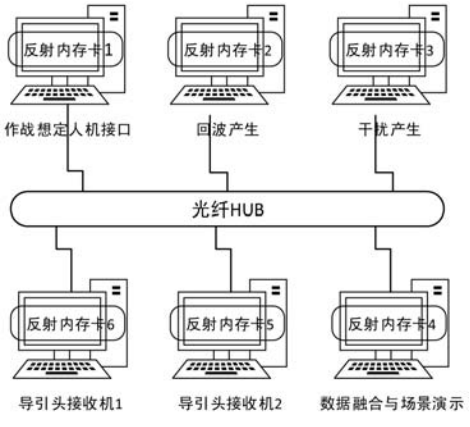

图 1　导引头系统架构图

图 2 给出了系统的数据流程,用户在作战想定人机接口模块的 Windows 软件界面上输入装订信息,RTX 应用程序通过共享内存读到装订信息后,再通过反射内存网将其分别发送给回波产生模块和干扰产生模块。两模块的 RTX 程序根据得到的信息计算出回波与干扰信号,并传送给导引头接收机 1 和导引头接收机 2 进行信号处理。接收机模块将处理得到的目标信息通过反射内存网传送给数据融合与场景演示模块,该模块在 RTX 系统下进行数据融合,然后将融合结果通过共享内存传送给 Windows 进行动画展示,最终完成整个导引头工作过程的实时仿真。

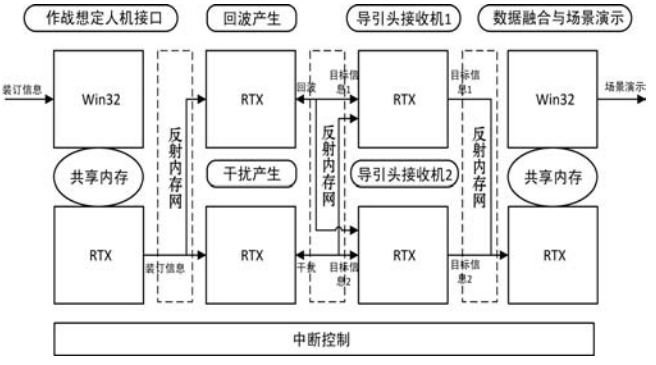

图 2　数据流程图

2　关键技术实现

2.1　RTX 与 Windows 的结合使用

(1) 在 Windows 下进行 RTX 开发。

RTX 是在 Windows 环境下安装和运行的。IntervalZero 公司将 RTX 封装为一个实时子系统 RTSS,以驱动的形式加载在 Windows 上。开发人员可以使用 Visual Studio(简称 vs)工具来加载 RTSS 并进行 RTX 开发,具体步骤如下:

① 安装 WDK(Windows Driver Kit)。

② 依次安装:RTX SDK、RTX runtime、RTX MM。

③ 在 vs 中新建 RTX Application 工程。

④ 在新建工程下调用 RTSS 的 API 函数 RtAPI 进行相关应用程序开发。

(2) RTX 与 Windows 间的通信机制。

为了克服与 Win32 进程间数据交互的困难,RTX 提供了 IPC 通信机制:共享内存、事件体、互斥体、信号量。其中共享内存实现进程间的数据交互[7-8],事件体、互斥体、信号量保证进程间的同步。本文基于该机制使系统具有实时性的同时能够实现友好的人机交互。

以数据融合与场景演示模块为例,在 RTX 下创建共享内存,然后将接收到的目标位置、速度等信息写入;在 Windows 下打开共享内存后便可将信息读出进行动画展示,其中对共享内存缓冲区访问的同步通过互斥体实现,任何一方对共享内存进行操作时,都需要对互斥体加锁,互斥体被释放后共享内存才可被另一方使用。该过程中用到的 RtAPI 函数如表 1 所示。

表 1 RtAPI 中 IPC 通信实现函数

函数名	功能
RtCreateSharedMemory	创建共享内存
RtOpenSharedMemory	打开共享内存
RtCreateMutex	创建互斥体
RtOpenMutex	打开互斥体
RtWaitForSingleObject	等待互斥体被释放

2.2 RTX 下反射内存卡驱动设计

本文导引头实时系统仿真中多机间的数据传输是通过反射内存网实现的,因此对 RTX 下反射内存卡驱动的开发十分关键。首先需要在 RTX 系统的 Control Pannel 工具中将设备从 Windows 移到 RTX 下。根据系统需要,本文的反射内存卡 RTX 驱动主要实现以下三个功能:开关设备、中断控制以及读写数据。驱动程序的流程如图 3 所示。下面对三个功能的实现分别进行阐述:

(1) *开关设备* 打开反射内存卡的步骤如下:首先将驱动程序附加到硬件设备上;然后将总线地址转换为相应的系统逻辑地址,再映射到用户的虚拟地址空间;其次在 I/O 总线上设置设备的总线配置数据;最后使用得到的地址空间句柄对设备中断状态,DMA 等进行初始化配置。

关闭反射内存卡只需关掉对设备进行操作的句柄,RTX 会自己回收内存资源。

(2) *中断控制* 中断控制包括中断使能、设置中断类型、发中断、响应中断与等中断五个部分。① 中断使能,首先创建事件对象,然后检查设备是否支持 MSI 或 MSI-X,挂载中断,最后将中断线程/例程关联到 line-based/message-based 的硬件中断。② 设置中断类型,通过配置反射内存卡本地配置寄存器的中断状态位来实现。③ 发中断,即将中断节点、附加信息、控制位信息(NIC、NTN、NTD)写入到本地配置寄存器,其中写 NIC 将启动网络中断。④ 响应中断,启动网络中断后,中断响应函数将判断中断类型,然后设置相应的事件对象。⑤ 等中断,等待中断触发,获取事件对象句柄,然后取出中断附加信息(ISD)、中断节点(SID),最后重新设置中断类型。

(3) *读写数据* 本文使用 DMA 和 memcpy 两种方式分别实现大数据块和小数据块的读写。

写数据时首先判断数据长度,若其小于 DMA 传输的最小值,则直接使用 memcpy 将用户数据复制到反射内存卡的内存中,否则,使用 DMA 进行传输。读数据时依旧首先判断数据长度,若其小于 DMA 传输的最小值,则直接使用 memcpy 将反射内存卡中的数据复制到用户申请的内存中,否则,使用 DMA 进行传输。其中 DMA 传输数据的步骤如下:判断传输方向;得到传输数据物理地址;利用中断的方式控制 DMA 对物理地址上的数据进行分块传输。

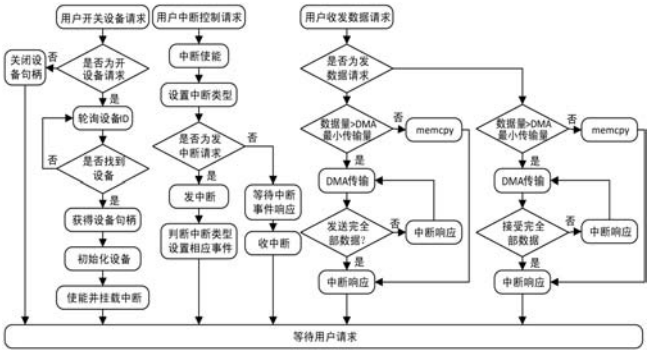

图 3 驱动程序设计流程图

2.3 RTX 下实现多线程

相比于 Windows 下多线程使用的诸多局限性,RTX 留给用户的操作空间大,可支持多达 997 个独立进程,每个进程的线程数不设限,由用户自由设置。RTX 的线程具有 128 个优先级且均高于 Windows 的所有线程和中断[9],线程间切换时间小于 10 μs,因此使用 RTX 的多线程可以极大提高程序的实时性。本文对多线程的应用体现在两个方面,一是实现数据的传输与处理并发进行,本文的数据传输是通过 DMA 控制加反射内存网的方式实现,不需要占用主机 CPU 资源,因此在数据传输过程中闲置的 CPU 可以用来进行数据处理,从而大大提升系统的工作效率。二是通过 RTX 多线程将计算机的多核充分利用起来实现信号处理速度的提升,提高系统的实时性。RTX 系统下线程创建及管理方法如下:

在 RTX 的 Control Pannel 工具中为系统分配内核,其中 Windows 至少分配一个核,其余供 RTX 使用。RTX 程序中主线程会占用第一个核,剩余的内核可由开发人员按照需要自由分配给子线程,为了系统最优,一般建议一个核分配一个线程。此外,为线程分配的内核必须属于其父进程。RTX 下创建使用线程的 C ++ 代码如下:

```
SetProcessAffinityMask(GetCurrentProcess(), 0x3E);
                                        //为当前进程分配内核
ULONG RTFCNDCL childThread1(void * nContext)
{
    //TODO:在此放入代码
}
childThread1 = RtCreateThread(NULL, 0, thread01, NULL,
CREATE_SUSPENDED, NULL);                //创建子线程
RtSetThreadPriority(childThread1, 120);
                                        //为子线程设置优先级
SetThreadAffinityMask(childThread1, 0x04);
                                        //为子线程分配内核
```

```
ResumeThread(childThread1);          //启动子线程
SuspendThread(childThread1);         //挂起子线程
```

3　系统测试

本文首先对所实现的导引头实时仿真系统的准确性、实时性、与真实作战场景的相似性在高性能硬件下分别进行了验证。计算机 CPU6 核、主频 3.7 GHz、32 GB 内存,反射内存卡采用 PCI-5565PIORC,PCIE 接口,容量 128 MB 字节。

(1) 准确性验证　如图 4 所示,在作战想定人机接口模块输入装订信息,在数据融合与场景演示模块将数据导引头系统仿真结果通过 Unity 进行动画展示,可以得到目标的飞行状态,结果准确。

图 4　装订信息输入界面

(2) 实时性验证　本文从以下四个方面分别验证实时性:

① 本文使用 Latrey Time 工具对仿真系统在采用 RTX 前后的延迟时间进行了测量,结果如图 5 所示,可以看到在 Windows 下系统延迟较大,且非常不稳定,最高延迟达 32 ms,而 RTX 下系统延迟时间均小于 10 μs,且相对稳定,符合实时系统要求。

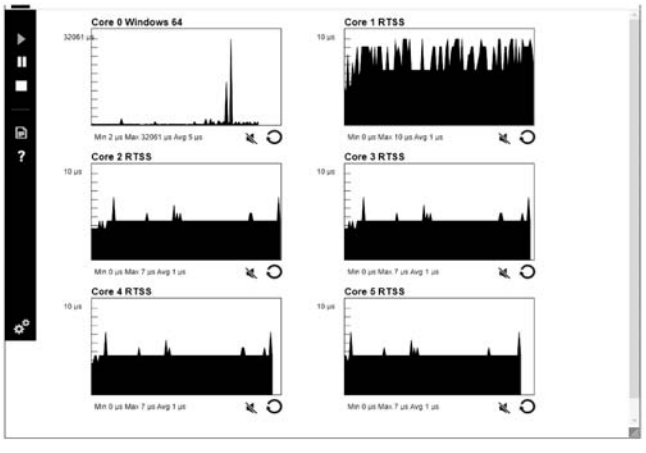

图 5　时延示意图

② 在 RTX 下使用 RtGetClockTime( ) 函数获取系统时间来计算中断响应时间,在 Windows 下使用 QueryPerformanceFrequency( ) 函数和 QueryPerformanceConter( ) 函数获取系统计时器频率和数值,计算得到 Windows 下的中断响应时间,进行 10 次测试后统计结果如图 6 所示。可以看到 Windows 下系统中断响应时间较大,且极不稳定,RTX 下系统中断响应时间稳定在 12 μs 左右,符合实时系统要求。

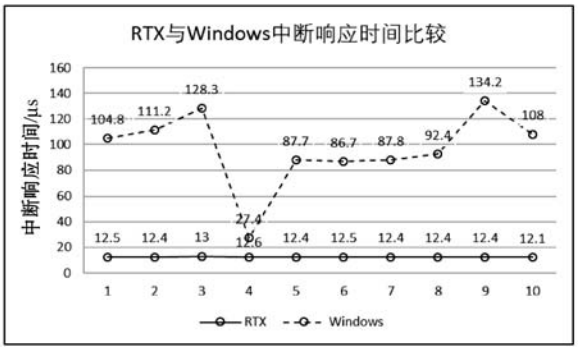

图 6　RTX 与 Windows 中断响应时间比较

③ 使用 CPUUsage 工具和 Monitor Utility 工具获得 CPU 使用率和线程切换时间,在单线程和多线程下的 CPU 利用率如图 7 所示。可以看到,在多线程下 CPU 核的利用率大大提高。一次信号处理时间由 2 s 减少为 0.5 s,提高了系统的实时性。通过查看 Monitor 的输出日志我们获得线程切换时间,最大为为 7.5 μs,符合实时系统要求。

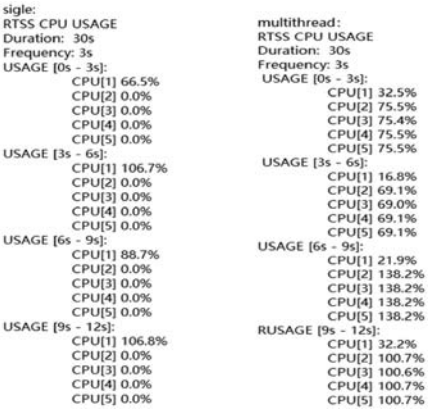

图 7　CPU 使用率打印信息

④ 使用 RtGetClockTIme( ) 函数获得传输时间,根据传输数据大小计算得到反射内存网数据传输率约为 880 Mbit/s,满足实时传输要求。

(3) 与真实场景的相似性验证　分别对本文实时仿真系统与传统 MATLAB 软件仿真系统的响应延迟、信号处理时间与数据传输率进行测试,取二十次测试的平均值,同真实作战场景下的性能指标进行对比,得到表 2。从表中可以看到,传统软件仿真系统响应延迟为 10.4 ms,信号处理时间为 2.7 s,而本文所设计的

实时仿真系统响应延迟仅为 8.7 μs,信号处理时间为 540 ms,更加接近真实作战场景下系统响应延迟和信号处理时间。另一方面,本文设计的实时仿真系统采用模块化的设计,各模块之间的数据传输速为 880 Mbit/s,相比于传统 MATLAB 单机软件仿真,更加接近于真实作战场景。

表 2 仿真方案性能对比

指标	传统 MATLAB 方案	本文方案	真实作战场景
系统延迟/μs	10 400	8.7	<0.1
信号处理时间/ms	2 700	540	<20
数据传输率/(Mbit·s^{-1})	/	880	5 000

4 结 语

本文使用 RTX 及反射内存网技术设计了一种导引头实时系统仿真架构,准确性高,实时性好,与传统的 MATLAB 软件仿真方案相比,更接近于真实作战环境。测试结果表明,系统仿真结果准确,计算机资源利用率较高,整体时延小于 10 μs,中断响应约为 12 μs,跨主机传输速度约为 880 Mbit/s,满足实时系统要求。

参 考 文 献

[1] 李宏伟,马秋. 基于 VMIC 和 VXI 分布式导弹自动测试平台[J]. 航空计算技术,2016,46(3):116 - 118,123.

[2] Song B,Jang C, Kim S, et al. A robot software middleware based on the OPRoS and the RTX[C]//International Conference on Ubiquitous Robots & Ambient Intelligence. IEEE, 2015.

[3] 陈达,路小凡,李明齐. 基于 RTX 增强 Windows 实时性的虚拟无线电实现方案研究[J]. 科学技术与工程,2015,15(17):172 - 178,183.

[4] 庞琳. 基于 Sora 平台的低功耗 WiFi 系统设计与实现[D]. 河南:郑州大学,2014.

[5] 闻达, 李明齐. 基于 RTX 的虚拟无线电高速数据接口实时性研究[J]. 科学技术与工程, 2017(27):232 - 236.

[6] 纪红. 基于反射内存网络的实时网络关键技术的研究[D]. 哈尔滨:哈尔滨工程大学,2013.

[7] 郑艳. 利用共享内存,实现进程间高效率数据共享[J]. 城市建设理论研究,2012(2).

[8] 李琴. 共享内存与有名管道在实时系统中的应用[J]. 大众科技,2008(4):71 - 72.

[9] 陈达, 陆小凡, 李明齐. 基于 RTX 增强 Windows 实时性的虚拟无线电实现方案研究[J]. 科学技术与工程,2015,15(17):172 - 178.

(上接第 65 页)

的有效交互。

3) 采用组件化设计思想和标准的数据交互格式,围绕运行监控需求,通过对 TCP/IP 协议在网络通信层和命令管理层的封装,并利用一对多的数据通信方式,确保了主从端数据可靠、有效的交互。

4) 采用自主搜索、二次自主搜索和主动反馈等三种节点嗅探机制,确保了仿真节点的可靠管理。

5) 科学合理规划运行监控工具在运维管理系统中的功能定位,隔离工具与具体应用关联,进一步扩展了工具在模拟训练、作战实验、装备试验等各类分布式仿真系统中的应用。

参 考 文 献

[1] 项磊, 杨新, 余晓刚, 等. 一种基于 HLA 的卫星分布式仿真系统研究及其应用[J]. 计算机应用与软件, 2015, 32(2):61 - 65.

[2] 吴集, 金士尧, 沈雪石. 面向方面的多智能体分布仿真平台设计优化[J]. 系统仿真学报, 2010, 22(11): 2579 - 2586.

[3] 邓毅俊, 曹建. 面向多 Agent 的分布式仿真平台[J]. 计算机仿真, 2012, 29(6): 363 - 368.

[4] 林亮亮. 基于 HLA 的仿真系统设计与实现[J]. 计算机与现代化, 2011, 188(4): 17 - 20.

[5] 刘力力, 张为华. 模拟器采样工具设计与实现[J]. 计算机应用与软件, 2012, 29(1): 1 - 3,73.

[6] 张柯, 邱晓刚, 彭春光, 等. 分布仿真实验管理系统的设计与实现[J]. 系统仿真学报,2008,20(24):6627 - 6630.

[7] 郭奇志, 陈光, 任卓君, 等. 基于 Android 智能手机的实验管理系统[J]. 计算机与现代化, 2015(10): 73 - 76.

[8] 林之丹. 计算机网络实验管理系统的设计与实现[J]. 赤峰学院学报(自然科学版), 2015, 31(9): 28 - 30.

[9] 王晓鹏. 大型分布式系统服务对象部署与运行监控技术的研究与实践[D]. 长沙:国防科学技术大学, 2006.

[10] 郑伟. 大规模数据中心监控系统的设计与实现[D]. 北京:中国科学院大学, 2013.

[11] 龚关, 廖湘科. 网络与业务运行监控系统的设计与实现[J]. 计算机工程, 2007, 33(10): 252 - 254.

[12] 陈彬, 张柯, 刘晓铖, 等. 分布仿真实验管理系统网络中间件的研究与实现[J]. 系统仿真学报, 2008, 20(24): 6639 - 6642.

[13] 赵斌, 郝红旗. 网络中间件在分布式仿真系统中的应用[J]. 计算机仿真, 2009, 26(12): 100 - 102.

[14] 史永胜, 李秀静. 基于单元要素的虚拟仪表平台设计方法[J], 计算机工程与设计, 2015, 36(8): 2287 - 2302.

[15] 沈萍萍, 陈珂, 张燕, 等. 分布仿真系统二次调度负载均衡策略研究[J]. 计算机仿真, 2011, 28(9): 223 - 225.