文章编号：2095-4980(2023)01-0036-08

卫星通信终端通用测试平台高速接口设计

陈　静[1]，靳铭洋[2]，陈　翔[*3]

(1.军事科学院 战争研究院，北京 100091；2.中国人民解放军78156部队，四川 成都 610000；
3.中山大学 电子与信息工程学院，广东 广州 510006)

摘　要：卫星通信系统向多频段多体制发展的趋势对卫星通信终端测试平台的通用性提出了更高的要求，基于通用处理器的软件无线电(GPP-SDR)技术可用于解决该问题。但目前GPP-SDR处理器和射频前端之间的高速数据传输仍是主要的技术瓶颈。本文针对卫星通信终端通用测试平台需求，提出了基于中断机制的PCI-e高速传输方法和流程设计，并通过移植Xenomai操作系统对高速接口传输的实时性进行优化；设计了一系列的测试实验对所设计接口的实际传输速率和中断丢失概率等指标进行测试。实验结果验证了所提方案具备高速率和低时延的传输特性，能够很好地满足当前卫星通信终端通用测试需求。

关键词：卫星通信测试；软件无线电；高速接口；中断机制

中图分类号：TN927.2　　**文献标志码：**A　　**doi：**10.11805/TKYDA2020442

High-speed interface design for universal testbench of satellite communication terminals

CHEN Jing[1]，JIN Mingyang[2]，CHEN Xiang[*3]

(1.Institute of War Research，Academy of Military Sciences，Beijing 100091；2.Unit 78156 of PLA，Chengdu Sichuan 610000；
3.School of Electronics and Information Technology，Sun Yat-Sen University，Guangzhou Guangdong 510006，China)

Abstract：Testbench for satellite communication terminals is required to be more universal since satellite communication systems are becoming multi-band and multi-system integrated. General-Purpose-Processor based Software Defined Radio(GPP-SDR) is suitable to solve this problem yet the high-speed data transmission between the processor and RF front-end of GPP-SDR is still one of the main bottlenecks. In this paper, to satisfy the requirements of satellite communication terminals testing, a high-speed transmission method based on interrupt mechanism is proposed. The real-time of the transmission interface is also optimized by utilizing the Xenomai Operating System(OS). A series of experiments are designed to test the actual transmission data rate, interrupt loss probability and other indexes, verifying that the proposed approach has good performance in high speed and low latency and can meet the current requirements on universal testing of satellite communication terminals.

Keywords：satellite communication testing；Software Defined Radio；high-speed interface；interrupt machanism

相比于地面移动通信，卫星通信具备覆盖面积广、通信容量大、不易受地理环境约束等优点，因此在通信领域一直处于战略性位置[1-2]。近几十年来，卫星通信系统获得了快速发展，卫星通信终端测试技术也相应地不断改进和完善，经历了从手动测试到自动测试，从引进到自主研发，从单机测试设备到自动化测试系统[3]。但卫星通信系统向多频段、多体制发展的趋势，对测试平台的通用性也提出了更高的要求。

在面对多样化、复杂化及频繁变化的测试需求时，虚拟仪器测试策略因其成本低、效率高、扩展灵活、功能强大，逐渐成为最重要的技术之一。基于软件无线电(SDR)和网络功能虚拟化(Network Function Virtualization，

收稿日期：2020-09-09；**修回日期：**2020-11-18

基金项目：国家自然科学基金资助项目(61501523)

[*]**通信作者：**陈　翔　email:chenxiang@mail.sysu.edu.cn.

NFV)是虚拟仪器测试技术的重要驱动技术。其中，SDR 技术于 20 世纪 90 年代左右提出[4−5]，其主要思路是通过构建可扩展的通用硬件前端，尽可能将通信系统中的数字信号处理算法转移为软件编程的方法实现，进而实现通信平台最大程度的通用性、灵活性和兼容性。这些优势使 SDR 非常适用于构建面向多频段、多协议、多体制的卫星通信终端通用测试平台。基于通用处理器的 SDR(GPP−SDR)又是目前最为通用的一种 SDR 系统实现形式，其灵活性更强，开发周期更短。

在目前的 GPP−SDR 系统中，软硬件处理平台(主要指处理器和射频前端)之间的高速数据传输接口是主要的技术瓶颈之一。传统上，通常采用 USB2.0 接口[6]或千兆以太网(Gigabit Ethernet，GE)接口[7]进行传输，这是因为这两种接口较为常见且发展成熟，但其传输速率分别只能达到 60 MB/s 和 125 MB/s，仅能满足一些对数据吞吐量需求不大的场景。目前通信卫星正在向宽带化发展，未来终端的通信带宽将达到几十或上百兆赫兹。相比而言，由 Intel 于 2003 年提出的 PCI Express(PCI−e)接口[8]具有延迟低、传输速率快的优点，更适合用于 GPP−SDR 中，PCI−e 1.x 版本单通道就能达到 250 MB/s 的速率，最多还可扩展至 16 通道，达到 8 GB/s 的高速传输；且延时只有微秒量级。目前已有不少文献研究基于 PCI/PCI−e 接口的数据传输方案[9−10]，但这些方案没有针对传输的速率和延时进行额外优化，实际使用时很难同时满足卫星终端测试的高速率、低时延两方面的要求。

本文面向卫星通信终端通用测试平台需求，提出了一种新的基于 PCI−e 的高速接口设计方案，该方案基于中断触发机制和实时操作系统实现，能够有效提高 GPP−SDR 中高速数据接口的传输实时性和可靠性。

1　卫星通信终端测试场景

图 1 为一个典型的卫星终端测试场景，主要分为卫星通信终端(待测单元)以及通用测试平台 2 部分。在典型的测试场景中，终端测试平台主要是通过模拟卫星移动通信系统信关站侧的基带和射频系统，实现信关站基础数据业务、物理层基带信号处理、物理层过程和射频信号处理功能，用于与卫星终端配合展开互联互通测试，从而对终端的射频一致性、信号处理一致性、协议一致性等指标进行测定。

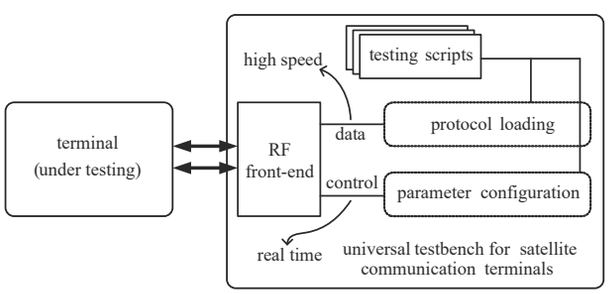


Fig.1 Testing scenario of satellite communication terminals
图 1　卫星通信终端测试场景图

在典型的测试平台设计上，一般根据测试条目通过上位机选择测试脚本，测试脚本自动控制相应的协议栈的加载以及对系统射频参数进行配置。在 GPP−SDR 架构下，这两者最终都需连接至射频前端。软件协议栈和射频前端的连接实际上是一个数据传输通道，因为软件协议栈(包括 L3/L2/L1)层最终输出的是物理层 I/Q 数据，这些数据由射频前端完成 D/A 和变频等操作，然后从射频接口发出。对于数据接口，最主要的需求是速率，速率越高，则能够支持的基带采样率越高，系统能够支持更广泛、更宽带的卫星通信系统测试需求。参数控制和射频前端之间，实际上是一个控制通道，因为参数配置单元将根据测试脚本的指导实时改变射频前端的频率、功率等参数，从而完成测试。对于控制接口，最主要的需求是实时性，实时性越高，系统越能够支持硬件参数的快速切换，从而能够对被测终端的响应时间进行更加准确的测量。综上所述，从卫星通信终端的测试场景出发，可以看到高速率、实时性是接口的主要需求，本文正是围绕这两个需求展开研究和设计。

2　系统架构和编程模型

无线通信系统中，无线帧结构严格定义了物理层固定时间段内的数据格式，描述了时间与传输数据量的对应关系。GPP−SDR 中，从数据量和时间 2 个不同的维度分别出发，可以得到 2 种不同的编程模型：数据驱动编程和中断驱动编程。

数据驱动编程以发送或接收的基带采样数据量大小作为协议处理的驱动事件。如图 2(a)所示，协议处理线程不断发起硬件轮询，监测已经发送或接收的数据量大小，当数据量达到一定门限以后，进入协议处理过程。处理完成后，线程继续回到硬件轮询状态，如此周期往复。在数据驱动编程模型下，帧格式的定时关系完全由通用射频前端负责，协议栈软件模块的结构可简化为只关心输入输出的数据处理单元。

中断驱动编程则是将来自通用射频前端的中断信号作为驱动事件。如图 2(b)所示，协议处理线程通常处于阻塞等待硬件中断信号的休眠状态。中断产生后，操作系统唤醒协议处理线程，调度其执行中断间隔对应数据量的协议处理任务。处理完成后，协议处理线程再次进入阻塞等待的休眠状态。中断驱动编程模型中，中断信号

将来自通用射频前端的高精确度定时信息引入软件。协议栈软件模块和通用射频前端在中断信号的指示下，以流水线的方式同步执行数据传输和处理任务，从而简单完成了软硬件数据同步。显然，该模型要求硬件中断源的频率足够高，以适应现代无线通信系统毫秒级别的帧长度，同时也要求操作系统具有对中断响应的实时性和可靠性。

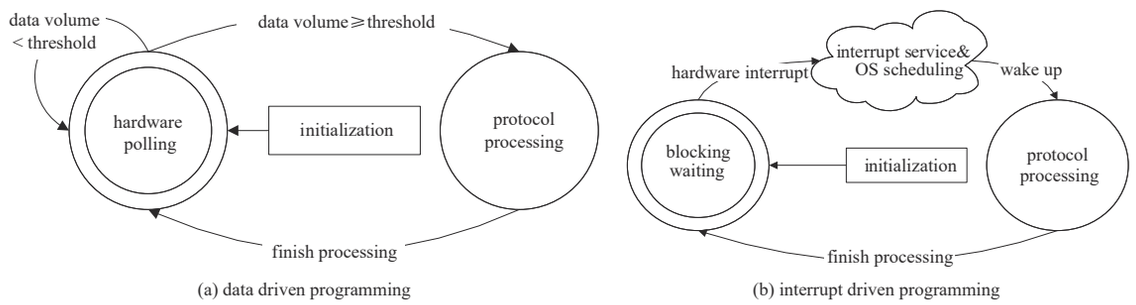


Fig.2 Comparison of two programming models in GPP-SDR
图2 GPP-SDR中两种编程模型对比

以上2种编程模型在工程实现中均是可选的。数据驱动编程本质上是一种CPU主动轮询机制，其优势在于可避免频繁的上下文切换，以及不存在请求遗漏和访问冲突等问题，但硬件轮询过程会浪费CPU宝贵的处理时间，这在速率高、计算任务重的卫星通信终端测试中是尤其不利的。相比之下，中断驱动编程模型的GPP-SDR架构具有比较明显的优势：除了不需要CPU进行轮询外，中断间隔利用硬件侧的时钟完成计数，其定时精确度要远高于软件操作系统提供的；同时，这种基于中断驱动的架构还非常适合于通信系统中基于帧或时隙的处理场景，若将中断间隔设为一个帧/时隙的时间，对于软件侧，收到的每一个中断都有特定的物理意义，天然地能准确定位每个帧/时隙的起止时刻，有利于基带信号处理或是上下行切换等操作。因此，本文在高速接口的设计上采用了中断编程模型。图3为基于中断驱动编程的GPP-SDR系统架构：外置的通用射频前端包括了DMA(Direct Memory Access)控制器、硬件中断控制器、数据缓存、数模/模数转换器、射频收发单元和射频天线，能够完成数据传输、中断产生和射频信号收发的功能。X86 CPU结合实时操作系统构成基础平台，在其上实现物理层的同步、调制解调、编解码等数字信号处理算法，以及MAC、RLC等高层协议处理。射频前端和CPU平台之间的通用数据接口是本文研究的重点，数据接口的吞吐率决定了GPP-SDR系统能够支持的最大无线通信系统带宽。选用吞吐率高、延时小的高速数据接口，保证了硬件平台对多种通信系统的兼容性，使其能够承载频带更宽、传输速率更高的测试需求。基于PCI-e X1.1接口进行高速数据接口设计，能够提供理论单向峰值速率2 Gbps的数据传输功能。

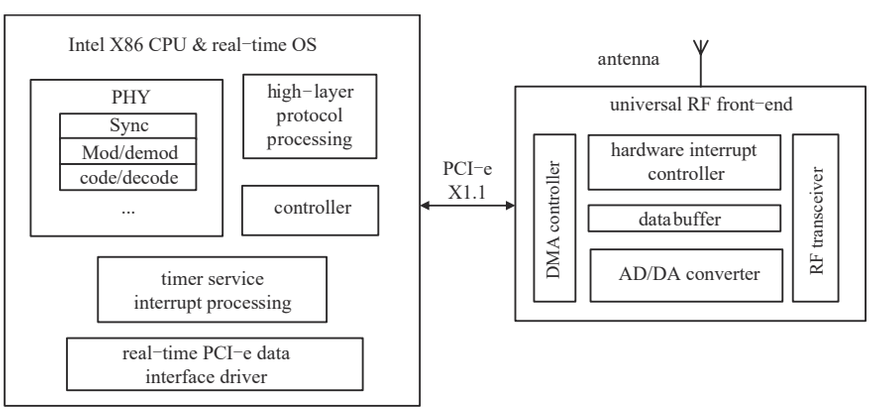


Fig.3 GPP-SDR system architecture designed for universal testing of satellite communication terminals
图3 面向卫星通信终端通用测试需求的GPP-SDR系统架构设计

3 高速接口方案设计

3.1 基于中断机制的驱动设计

对于通用计算机，操作系统需要负责硬件设备的管理和使用。访问硬件设备，需要向操作系统安装硬件对应的驱动程序。驱动程序作为具有硬件访问特权的软件模块运行在操作系统的内核模式下，主要实现对硬件设

备的辨识、注册、读写和卸载[11]。为了方便驱动程序开发，操作系统对硬件设备做了分类，如Linux将硬件设备分为字符设备、块设备等；实时操作系统Xenomai分为CAN设备、串行设备等，并为每种类型提供了标准的驱动编程框架。实现编程框架定义的操作，应用程序便能通过系统调用接口访问硬件设备。对于PCI−e接口，本文分别实现了Linux的字符设备框架和Xenomai的实时串行设备框架，进而研究PCI−e数据传输在2种操作系统下的性能表现。

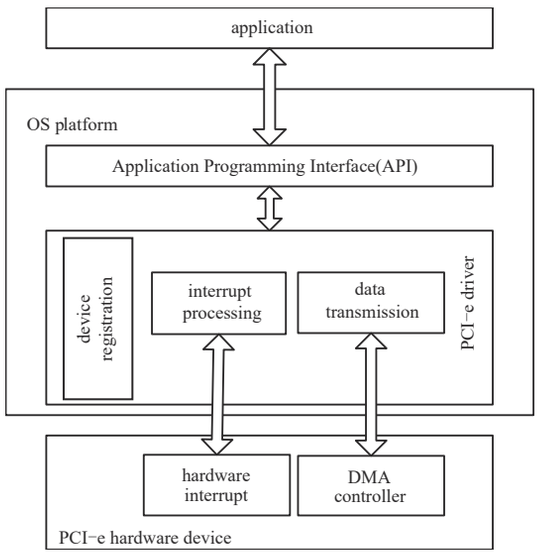


Fig.4 PCI−e driver structure based on interrupt mechanism
图4 基于中断机制的PCI−e驱动程序结构

2种操作系统中PCI−e接口驱动程序的结构基本一致。如图4所示，中断处理部分捕获通用射频前端板产生的中断信号，通知操作系统唤醒当前阻塞等待中断的软件协议栈线程；数据传输部分控制通用射频前端板中的DMA控制器，完成计算机和通用射频前端之间的数据交换；驱动程序实现了Linux字符设备或实时串行设备编程框架的系统调用接口，允许协议栈软件以文件读写的方式接收或发送数据。此外，为了最大程度体现接口的灵活性和通用性，驱动程序还开放了硬件配置的相关接口，允许软件直接修改通用射频前端的硬件配置寄存器，动态改变以下参数设置：a) DMA运行参数；b) 周期中断产生的时间间隔；c) 中断间隔内收发数据块的大小。这些参数的可调整有助于GPP−SDR系统适应不同卫星波形协议的各种不同带宽和采样率配置需求。

据此设计，基于中断机制的PCI−e数据传输的过程可用图5来描述。传输的具体步骤可总结如下：

1) 中断信号产生，驱动程序的中断处理函数通知操作系统唤醒阻塞等待的线程，操作系统调度该线程执行。此过程耗时主要是操作系统的调度延时(scheduling delay)；

2) 线程向硬件写入DMA命令控制字，设置硬件DMA运行参数(本次传输数据块在内存中的起始地址和数据块大小)。此过程耗时对应PCI−e硬件寄存器写入延时(register writing delay)；

3) 通用射频前端的DMA控制器启动，按照步骤2)设置的DMA运行参数，将内存中对应位置的数据通过PCI−e总线传送到硬件缓存中。此过程耗时对应突发传输延时(burst transmission delay)。

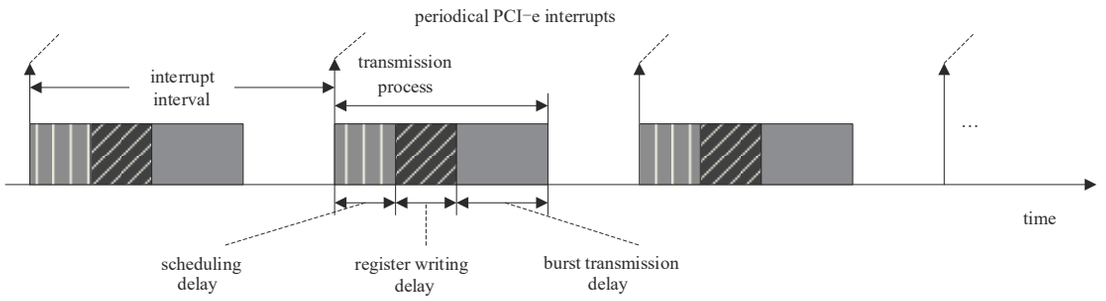


Fig.5 PCI−e data transmission process
图5 PCI−e数据传输过程

3.2 基于Xenomai操作系统的实时性优化

在GPP−SDR系统中，处理器上操作系统对中断的响应速度直接影响高速接口的实时性。传统Linux操作系统采用软实时的调度方式，其进程调度的时间不确定性以及系统定时器过高的颗粒度，很难满足对实时性能要求苛刻的场景。为此，本文在GPP−SDR系统处理器上移植Xenomai实时操作系统[11]，提高处理器对实时进程的调度能力和中断响应速率，进一步优化了高速接口的实时性，降低了传输的时延。

本文采用操作系统自适应域(Adaptive Domain Environment for Operating System，ADEOS)实现Linux和Xenomai两个操作系统在同一个处理器平台上的共存。Linux和Xenomai可以在ADEOS上作为域各自实现：实时任务(包括与高速接口相关的处理程序等)由Xenomai处理，Linux内核作为最低优先级的进程，在没有实时任务时运行。这样可以在保留利用Linux操作系统稳定成熟、对硬件支持广泛优势的基础上，完成实时性的优化。ADEOS是在现有操作系统下插入一个硬件虚拟层，通过向上层多个操作系统提供一些原语和机制而实现硬件共

享，位于硬件和ADEOS上层的操作系统仍可以自由操作硬件，不会因为ADEOS的存在而受到任何的影响和约束(实际上上层操作系统可以完全不知道有ADEOS的存在)，通过ADEOS可以实现操作系统对系统资源的共享。

对于基于中断的高速接口设计，最重要的是利用ADEOS提供的中断管理机制，中断通过中断管道在ADEOS上层域之间的传播，优先级越高的域在管道内的位置越靠前。Xenomai利用ADEOS提供的中断管道机制及中断屏蔽实现对系统中断的控制。如图6所示，Xenomai在ADEOS中断管道内的优先级高于Linux，当底层有PCI-e周期性中断发生时，ADEOS将中断放入管道内，由于该中断属于实时任务中断，Xenomai域会调用相应的实时中断处理程序并利用中断屏蔽将此中断终止；对于其他非实时任务所需的中断，则将其传入至Linux域内。通过这种方法接管了Linux系统的中断，并利用Xenomai的实时性优势实现对PCI-e高速接口产生的PCI-e中断快速响应。

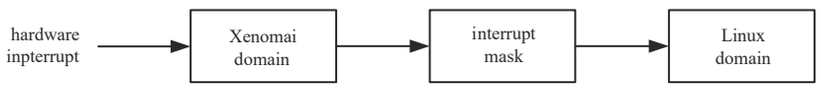


Fig.6 Interrupt control principle of Xenomai
图6 Xenomai中断控制工作原理图

4 实验结果和分析

对设计的PCIe高速数据传输接口的性能进行两方面的测试：一是数据吞吐量，通过测试接口实际能够达到的传输速率，验证所提方案能够支持高速数据传输；另一个是实时性测试，包括对Linux和Xenomai操作系统对中断的响应能力、控制寄存器写入延时和数据突发延时等指标进行对比，验证所提方案能提高接口可靠性和实时性。实验环境的参数配置如表1所示。

表1 PCI-e高速接口测试环境

Table 1 Configuration for testing PCI-e high-speed interface

item	configuration
operating system	Linux kernel 2.6.32.20/Xenomai 2.5.4
CPU	Intel Core2 Quad Q9550 2.83 GHz/ Intel Core i7 860 2.80 GHz
PCI-e interface type	PCI-e 1.1 X1 (peak rate 2Gbps)
sampling rate	30.72 MHz
data width	16 bit I/Q

4.1 数据吞吐量

在数据吞吐量测试环节，将PCI-e的中断间隔配置为500 μs，并以4 480 Byte作为一个数据块，统计中断发生的次数以及每次传输1个数据块需要的时间。在1 h的测试时长内，硬件总共发出了720万次中断，数据传输的时间统计如表2所示。

表2 数据传输时间统计

Table2 Data transmission time statistic

item	value
average value/μs	29.2
maximum value/μs	36.5
minimum value/μs	28.7
standard deviation	24.8

由表2可以得出本文设计的PCI-e高速接口能够达到的数据吞吐量为153.4 MB/s。以几个发展较为成熟的卫星系统为例，我国静止轨道的“天通一号”最高传输速率为384 kbps；美国休斯公司运行的高/中轨道混合组网的SPACEWAY宽带卫星通信系统传输速率为16 kbps~6 Mbps；美国的低轨卫星通信系统铱(Iridium)星二代星L频段传输速率最高达1.5 Mbps，Ka频段的便携式终端速率最高达10 Mbps。总的来说，目前L/S频段的卫星受限于系统带宽约束，用户终端基带采样速率大多在20 MHz以内(假设以数据速率的2倍进行采样)，若每个I/Q采样点采用16 bit的位宽表示，则需要的基带传输速率为80 MB/s。因此，本文设计的高速接口速率显然能够很好地满足目前卫星通信终端测试的需要，并在未来若需支持带宽更宽的系统，本文方案只需做很小的改动即可满足，如采用PCI-e v1.1 x2接口或PCI-e v2.0版本。

4.2 实时性测试

4.2.1 数据突发延时

表3和图7为数据突发延时的测试结果。其中，图7为Linux/Xenomai环境下数据突发延时的累积分布函数曲线图。对于Linux或Xenomai环境，Core2 Q9550型号的CPU测试结果中数据突发延时小于110 μs的概率超过90%；I7 860型号的CPU测试结果中数据突发延时小于90 μs的概率超过90%。可见，200 μs中断间隔内，用于传输数据的时间不超过150 μs，充分说明PCI-e X1.1接口可以完全满足卫星通信终端测试的高速率传输需求。注意到，对于同一CPU型号，Linux和Xenomai环境下的数据突发延时几乎完全一致，这是因为数据突发传输由硬件DMA控制器启动和控制，数据突发传输延时只依赖于测试环境的硬件环境，而与操作系统的关系不大。

4.2.2 中断丢失概率

在PCI-e数据的传输过程中，中断丢失概率会直接影响可靠性及稳定性，中断发生丢失主要与系统调度延时有关。调度延时由操作系统调度算法决定，根据运行时负载的差异，其数值具有一定的不确定性。不可预测的过长调度延时将导致操作系统无法及时调度线程响应中断信号并做出相应处理动作，对于等待中断信号线程，

相当于某些时刻的中断丢失了。如图 8 所示，调度延时使得原本对应于#1 号中断信号的响应动作被推迟至#3 号中断信号，即线程错过了#1、#2 两次中断的执行机会。对中断驱动编程下的 GPP−SDR 系统，连续的中断丢失将引发系统同步紊乱，严重时甚至造成系统崩溃。从协议栈处理的角度出发，本文主要考察连续丢失中断的数目，而不统计调度延时的具体数值。

表 3　PCI−e 数据突发传输延时对比

Table3　Comparison of PCI−e burst transmission delay

burst transmission delay	Core2 Q9550		I7 860	
	Linux	Xenomai	Linux	Xenomai
average value/μs	106.97	106.89	84.77	85.91
maximum value/μs	133.14	147.16	96.20	86.49
minimum value/μs	106.24	106.01	69.30	84.76

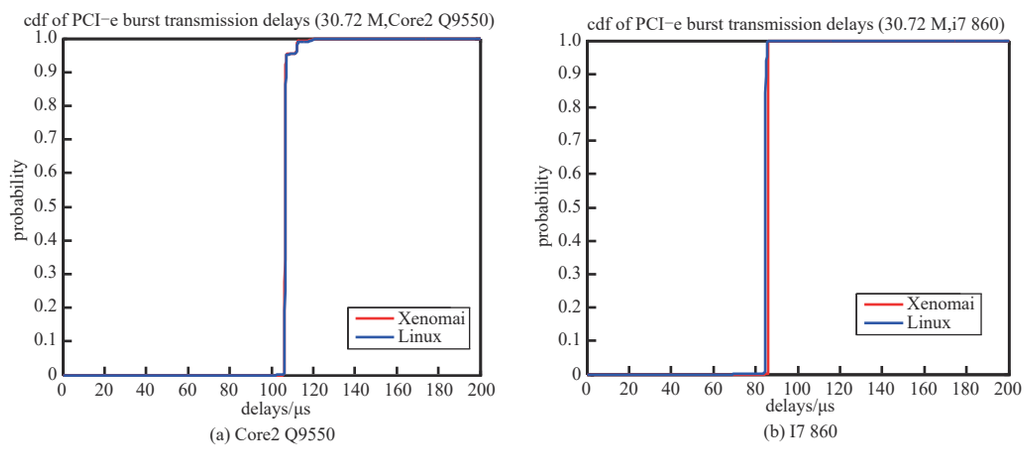


Fig.7 Comparison of PCI−e burst transmission delay

图 7　PCI−e 突发传输延时对比

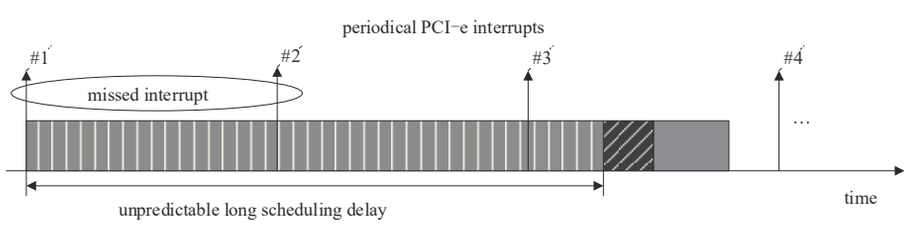


Fig.8 Interrupt loss due to unpredictable long scheduling delay

图 8　不可预测的过长调度延时引发中断丢失问题

表 4　PCI−e 接口中断丢失情况对比

Table4　Comparison of PCI−e interface interrupts loss

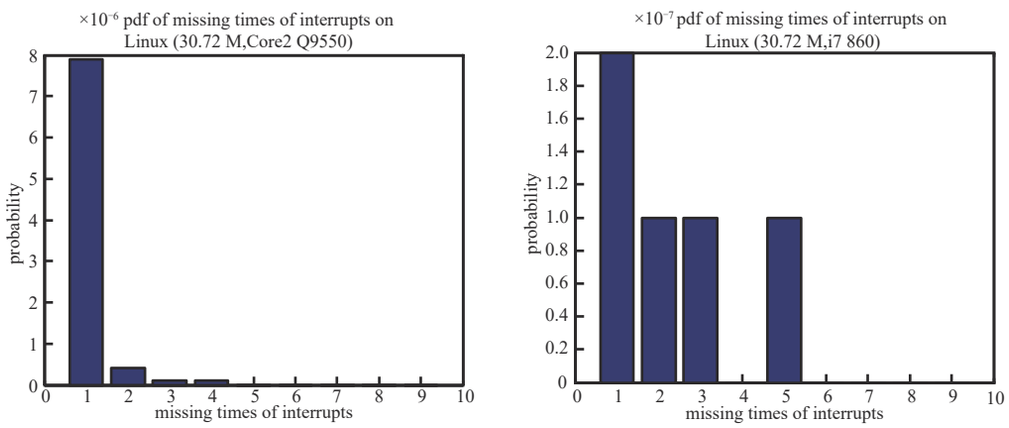
scheduling delay	Core2 Q9550		I7 860	
	Linux	Xenomai	Linux	Xenomai
interrupts loss probability	1.0×10^{-5}	0	5.0×10^{-7}	0
maximum number of consecutive interrupts loss	4	0	5	0
estimated maximum scheduling delay/μs	800	<200	1 000	<200

在中断丢失概率测试中，PCI−e 的中断间隔设置为 200 μs，一共测试了 2×10^7 次中断，结果如表 4 所示。可以看到，在 Linux 环境下，测试的 2 种 CPU 型号分别存在约 1.0×10^{-5} 和 5.0×10^{-7} 的中断丢失概率，并且连续丢失中断的最大数目达到了 4 次和 5 次，即 Linux 调度延时达到了 1 ms，图 9 进一步给出了 Linux 环境下连续中断丢失数目分布的柱状图。反观在 Xenomai 实时操作系统中，测试过程中完全没有出现中断丢失问题，可以保证其调度时延严格小于中断间隔 200 μs，说明经过移植实时操作系统进行实时性优化之后，PCI−e 高速接口的传输延时能够进一步下降，响应中断的稳定性也得到保证，传输可靠性得到提高。

4.2.3　控制寄存器写入延时

PCI−e 传输过程中除了数据流的传输之外，还需对控制寄存器进行写入操作，写入的延时大小也会对传输过程的稳定性和速率产生影响，因此本文对所提方案下 PCI−e 高速接口的控制寄存器写入延时进行了统计和对比，如表 5 所示。

图 10 为在 Linux 及 Xenomai 两种操作系统中的 PCI-e 寄存器写入延时的概率累积曲线图。测试结果表明，Linux/Xenomai 下 PCI-e 寄存器写入延时统计值均比较稳定，绝大部分保持在 10~20 μs 范围内。上述结果中还可以看到，Linux 寄存器写入延时最差可到数百 μs，而 Xenomai 最差也不会超过 100 μs，这也说明 Xenomai 的执行环境切换速度比 Linux 更快，传输接口的实时性保证更强，证明本文设计的基于 PCI-e 的高速数据接口的硬件访问速度快，延时小，这对卫星通信终端测试过程中的快速硬件控制至关重要。



(a) Core2 Q9550 (b) I7 860

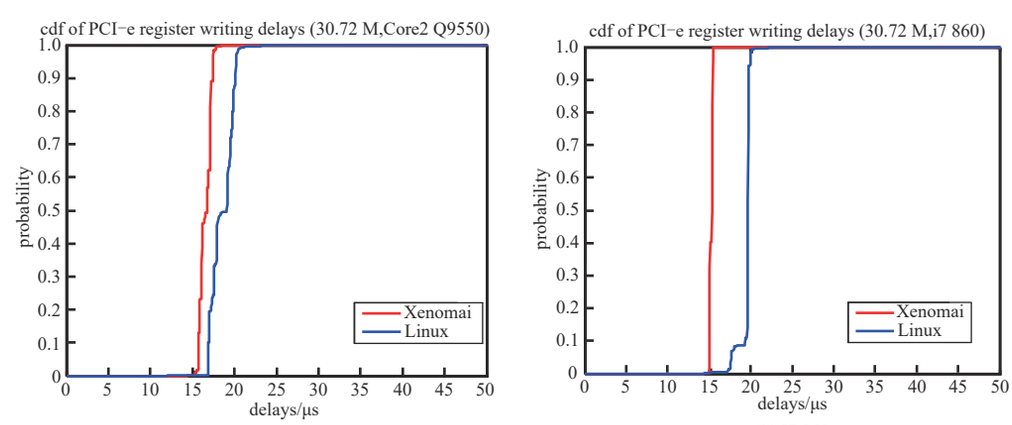
Fig.9 Distribution of the number of consecutive interrupts loss under Linux

图 9 Linux 环境下连续中断丢失数目分布图

表 5 PCI-e 控制寄存器写入延时对比

Table5 Comparison of PCI-e register writing delay

burst transmission delay	Core2 Q9550		I7 860	
	Linux	Xenomai	Linux	Xenomai
average value/μs	18.61	16.60	19.53	15.27
maximum value/μs	354.18	50.51	191.13	19.10
minimum value/μs	12.01	14.48	14.46	14.43



(a) Core2 Q9550 (b) I7 860

Fig.10 Comparison of PCI-e register writing delay

图 10 PCI-e 寄存器写入延时对比

5 结论

本文从卫星通信终端通用测试需求出发，分析了基于 GPP-SDR 平台实现的优势，并指出目前该方案主要存在的技术瓶颈，提出了基于 PCI-e 接口的高速接口设计方案。重点探讨了 GPP-SDR 中基于数据驱动编程和中断驱动编程的区别，提出了给予中断机制的 PCI-e 高速传输方法和流程设计；并将实时操作系统 Xenomai 移植至 GPP-SDR 平台上，进一步提出对高速接口传输的实时性优化方案；最后对基于所提方案设计的 PCI-e 高速接口的实际传输速率、突发数据延时、中断丢失概率和寄存器写入时延等指标进行测定，验证了所设计的接口满足高速率、低时延传输的要求，能够很好地满足当前卫星通信终端通用测试的需求。此外，本文设计的接口方案还具有很好的通用性和可扩展性，对于未来 Ku/Ka 频段更高速率终端的测试，在基本不改动接口方案的前提下，

可以通过使用v2.0/v3.0等高版本的PCI-e接口或者采用多路并行通道(PCI-e最高可支持16通道)的方式进行扩展从而完成升级扩展；同时，可以在GPP-SDR的架构下运行OpenAirInterface或srsLTE等开源软件协议栈，并利用本接口的高速和实时特性，即可将测试平台迁移到4G乃至5G蜂窝通信系统测试中。

参考文献：

[1] 张更新,张杭. 卫星移动通信系统[M]. 北京:人民邮电出版社, 2001. (ZHANG Gengxin,ZHANG Hang. Satellite mobile communication system[M]. Beijing:Posts & Telecom Press, 2001.)

[2] 张乃通,汪洋. 对发展卫星通信的思考[J]. 数字通信世界, 2005(10):17-21. (ZHANG Naitong,WANG Yang. Thoughts on the development of satellite communications[J]. Digital Communication World, 2005(10):17-21.)

[3] MITOLA J. The software radio architecture[J]. IEEE Communications Magazine, 1995,33(5):26-38.

[4] MITOLA J. Software radios: survey, critical evaluation and future directions[J]. IEEE Aerospace and Electronic Systems Magazine, 1993,8(4):25-36.

[5] ZHOU J, QI X, SU X, et al. Investigation on USB 2.0 in software-defined radio[C]// The 7th International Conference on Communications and Networking in China. Kunming,China:IEEE, 2012:833-837.

[6] 龙桂明,周春晖,赵明. 基于千兆以太网的软件无线电前端设计[J]. 微计算机信息, 2007(32):1-2,10. (LONG Guiming,ZHOU Chunhui, ZHAO Ming. Design of high speed transmitting for SDR front-end based on Gigabit Ethernet[J]. Microcomputer Information, 2007(32):1-2,10.)

[7] PCI-SIG. PCI Express Base 2.0 specification[S]. 2007.

[8] 吴国华,杨自恒,郭俊磊,等. 基于FPGA的高速PCI-e的数据传输设计与实现[J]. 无线电通信技术, 2019,45(1):96-99. (WU Guohua,YANG Ziheng,GUO Junlei,et al. Design and implementation of data transmission based on FPGA for high speed PCI-e[J]. Radio Communications Technology, 2019,45(1):96-99.)

[9] 李小龙,孟李林,邵瑞瑞,等. 基于FPGA的PCI Express应用平台设计[J]. 电子科技, 2014,27(12):108-111. (LI Xiaolong, MENG Lilin, SHAO Ruirui, et al. Design of PCI Express application platform based on FPGA[J]. Electronic Science and Technology, 2014,27(12):108-111.)

[10] 马鸣锦,朱剑冰,何红旗,等. PCI,PCI-X和PCI Express的原理及体系结构[M]. 北京:清华大学出版社, 2007. (MA Mingjin, ZHU Jianbing,HE Hongqi,et al. The principle and architecture of PCI,PCI-X and PCI Express[M]. Beijing:Tsinghua University Press, 2007.)

[11] Xenomai Team. Xenomai wiki homepage[EB/OL]. (2020-2-6). https://gitlab.denx.de/Xenomai/xenomai/-/wikis/home.

作者简介：

陈　静(1981-)，女，硕士，助理研究员，主要研究方向为软件测试、指挥控制.email:jingchen1210@sina.com.

靳铭洋(1981-)，男，学士，工程师，主要研究方向为通信网管、卫星通信、通信装备管理.

陈　翔(1980-)，男，博士，副教授，主要研究方向为无线与移动通信、卫星通信、物联网.