第 36 卷 第 4 期 计 算 机 工 程 2010 年 2 月
Vol.36 No.4 Computer Engineering February 2010
·工程应用技术与实现· 文章编号：1000—3428(2010)04—0239—02 文献标识码：A 中图分类号：TP393

基于 Vxworks 的 PCI-RapidIO 桥驱动设计

黄振中，柴小丽，黎 想，骆 意
(华东计算技术研究所，上海 200233)

摘 要：为了使 CPU 能通过 PCI 接口连接到 RapidIO 系统中，利用 PCI-RapidIO 桥的硬件设备，在 Vxworks 操作系统平台上开发该设备的驱动程序。测试结果证明，该驱动程序能在 PCI 端对 RapidIO 总线进行操作，实现 RapidIO 的基本 I/O、消息传递、系统启动和多播功能。

关键词：RapidIO 协议；PCI-RapidIO 桥；Vxworks 操作系统

Driver Design of PCI-RapidIO Bridge Based on Vxworks

HUANG Zhen-zhong, CHAI Xiao-li, LI Xiang, LUO Yi
(East China Institute of Computer Technology, Shanghai 200233)

【Abstract】 In order to make CPU can connect to RapidIO system through PCI interface, this paper uses hardware equipment of PCI-RapidIO bridge, designs the driver of hardware equipment by using Vxworks operating system platform. Test results prove that this driver can operate RapidIO bus at the end of PCI, realize basic I/O, message passing, system bring up and multicast function of RapidIO.

【Key words】 RapidIO protocal; PCI-RapidIO bridge; Vxworks operating system

1 概述

传统的嵌入式系统互连结构是通过分级共享总线实现的，但其较低的带宽和较高的延迟无法满足嵌入式系统对性能越来越高的要求。为了适应嵌入式系统高速通信的需求，因此，使用新的总线来替代原有的共享总线。

RapidIO 是一种新型高性能、低引脚数、基于报文交换的互连体系结构，是为了满足现在、未来高性能嵌入式系统需求而设计的一种开放式互连技术标准，采用高性能 LVDS 技术，能在 4 对差分线上实现 10 Gb/s 的有效传输速率，具有比万兆以太网、PCI express 更高的传输效率，能广泛满足嵌入式系统应用的需求[1]。

目前，仅有少量最新的嵌入式 CPU 内嵌了 RapidIO 总线接口，因此，需要开发一个接口卡，能使以前的 CPU 通过 PCI 接口连接到 RapidIO 系统中。本文研究了一种基于 Vxworks 的 PCI-RapidIO 桥驱动程序。

2 RapidIO 规范

RapidIO 采用 3 层分级体系结构[2]，具体描述如下：

(1)物理层：位于最底层，负责描述器件级接口规范。例如，分组传输机制、流量控制、电特性以及低级错误管理。

(2)传输层：位于中间层，定义 RapidIO 地址空间和在端点器件间传输包所需的路由信息。

(3)逻辑层：位于最高层，定义全部协议和包的格式，为端点器件发起和完成事务提供必要的信息。用户可以根据应用层所需，选择其中的几个协议，并不需要实现所有的协议。

RapidIO 操作基于请求和响应事务[3]。包是系统中端点器件间的基本通信单元。发起器件或主控器件产生一个请求事务，该事务被发送至目标器件。目标器件产生一个响应事务返回至发起器件来完成该操作。RapidIO 事务被封装在包中，包则包含确保将事务可靠传送至目标端点的所有必需的位字段。通常不会将 RapidIO 端点相互直接连在一起，而是通过介于其间的交换结构连接。“交换结构”是指提供系统互连的单个或多个交换器件的集合。

该分级结构如图 1 所示。

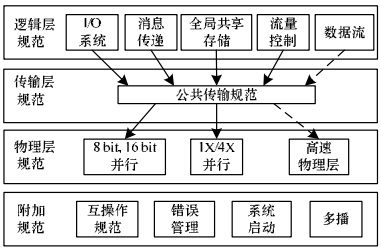

图 1 RapidIO 规范层次结构

3 PCI-RapidIO 桥

PCI-RapidIO 桥接器实现了串行 RapidIO 和 PCI 接口的相互转换以及 RapidIO 的终端设备功能。其总体功能框架见图 2。

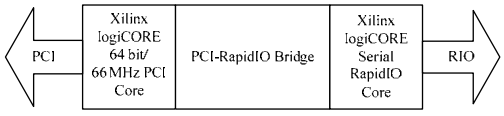

图 2 PCI-RapidIO 桥总体实现方案

其中，PCI 核和 RIO 核均采用 Xilinx LogiCORE IP Core；PCI_RIO_Bridge 完成了这 2 个核之间的接口信号转换，时钟域的跨接等功能。桥接器分别针对 2 个 IP 核提供接口信号与它们相连，在 PCI 一侧桥接器提供 PCI Core 用户应用程序接口信号，而在 RapidIO 侧提供 RapidIO 用户接口程序完成组包与解包的工作。

在 PCI-RapidIO 桥中使用 1 个 DMA(Direct Memory Access)，

作者简介：黄振中(1985－)，男，硕士研究生，主研方向：嵌入式计算；柴小丽，研究员；黎 想，高级工程师；骆 意，工程师
收稿日期：2009-07-08 **E-mail：**hzznihao@sina.com

提供给驱动层一系列寄存器，用于控制桥接器的工作。

4 PCI-RapidIO 桥驱动设计

PCI-RapidIO 桥驱动设计过程见图 3。

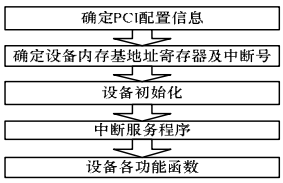

图 3 驱动程序设计过程

4.1 PCI 配置信息的设定及设备初始化

PCI 总线是一种即插即用的总线，在板级支持包的支持下，能够自动地为设备分配内存映射空间和中断号，驱动程序首先要在初始化过程中确定这些在计算机上电后自动分配的值，程序如下：

```
if(pciFindDevice(0x0606,0x8080,unit,&pciBus,&pciDev,&pciFunc)==ERROR)
    {       return 0;       }
pciConfigInLong(pciBus,pciDev,pciFunc,PCI_CFG_ADDRESS_0,&membaseCsr);
pciConfigInByte(pciBus,pciDev,pciFunc,PCI_CFG_DEV_INT_LINE,&irq);
Baseaddr=membaseCsr&0xfffffff0;
intConnect(INUM_TO_IVEC((int)(irq)),(VOIDFUNCPTR)intFunc,0);
intEnable(irq);
```

驱动程序先根据设备标识 Device ID(8080)和供应商标识 Vendor ID(0606)调用 pciFindDevice 函数确定其总线号 busNo、设备号 devNo 和功能号 funcNo，然后根据总线号、设备号和功能号调用 pciConfigInLong 及 pciConfigInByte 函数得到内存映射基地址寄存器 membaseCsr 和中断号 IntNumber，然后根据中断号挂接中断服务程序[4]。

4.2 中断服务程序

根据 PCI-RapidIO 桥的需要，中断服务程序如下：(1)DMA 中断：告诉主机 DMA 已发送完成，在此释放信号量，意味着可以进行下一次 RapidIO 事务发送。(2)RapidIO 通道上下电中断：告诉主机端口 0 或 1 是否连接着另一个 RapidIO 器件。(3)收到 Doorbell 中断：告诉主机是谁发送了该 Doorbell 和该 Doorbell 的内容是什么，提供 DoorbellHook 函数供用户去挂接自己需要的回调函数。(4)收到 Message 中断：告诉主机是谁发送了该 Message、该 Message 共由几个段组成、这是第几个段以及该段长度是什么，然后交给一个任务进行组包工作。提供 MessageHook 函数供用户去挂接自己需要的回调函数。(5)收到 Response 中断(如读数据返回)。

中断服务程序的编写应尽可能精练，使其能尽快返回，不能造成系统阻塞而影响系统性能。

4.3 设备功能函数

RapidIO 设备功能函数有很多，现阶段基本实现了 RapidIO 逻辑层中的基本 I/O、消息传递，以及附加规范中的互操作规范、系统启动和多播等。主要功能函数列举如下：

4.3.1 基本硬件抽象层函数

基本硬件抽象层(Hardware Abstraction Layer, HAL)包括：

(1)配置读函数

```
VSTATUS rioConfigurationRead(VINT8 localport, VINT16 destid, VINT8 hopcount, VINT32 offset ,VINT32 *readdata)
```

读取某个本地或远程 RapidIO 器件的 CAR 或 CSR 寄存器。

(2)配置写函数

```
VSTATUS rioConfigurationWrite(VINT8 localport, VINT16 destid, VINT8 hopcount, VINT32 offset, VINT32 writedata)
```

改写某个本地或远程 RapidIO 器件的 CAR 或 CSR 寄存器。

(3)Nread 函数

```
VSTATUS rioNread(VINT8 localport, VINT16 destid, VINT32 pciaddr, VINT32 rioaddr,VINT32 bytecnt)
```

读取远程端点的内存数据。

(4)Nwrite 函数

```
VSTATUS rioNwrite(VINT8 localport, VINT16 destid, VINT32 pciaddr, VINT32 rioaddr,VINT32 bytecnt)
```

向远程端点的内存写入数据。

(5)rioNwriteDoorbell 函数

```
VSTATUS rioNwriteDoorbell(VINT8 localport, VINT16 destid, VINT32 pciaddr, VINT32 rioaddr,VINT32 bytecnt,VINT16 dbinfo)
```

先写入数据到远程端点的内存，再发送门铃事务，该门铃事务带有 16 bit 的简单消息 dbinfo。

(6)发送门铃事务函数

```
VSTATUS rioSendDoorbell(VINT8 localport, VINT16 destid, VINT16 dbinfo)
```

发送门铃事务给远程端点，该门铃事务带有 16 bit 的简单消息 dbinfo。

(7)发送消息事务函数

```
VSTATUS rioSendMessage(VINT8 localport, VINT16 destid, VINT32 pciaddr, VINT32 standardsize,VINT32 bytecnt)
```

给远程端点发送最大值为 4 096 Byte 的消息。

(8)接收消息函数

```
VSTATUS rioReceiveMessage(void *buffer, int timeout)
```

从消息接收队列中获取一条消息，放入指定的 buffer 中，其中消息的前 8 个字节包括 SourceID 和消息大小信息，Timeout 为等待时间。

4.3.2 设置广播专用 ID 函数

```
VSTATUS rioSetMuticastReg(VINT8 localport, VINT16 destid, VINT8 hopcount, VINT16 multicastID)
```

使用该函数对交换芯片设置好广播专用 ID(multicastID)之后，就可以用 Nwrite 函数来进行广播通信，只须将 destid 填成广播专用 ID 即可[5]。

4.3.3 路由表 HAL 函数

路由表 HAL 函数包括：

(1)在指定 RapidIO 交换机的路由表中添加一个条目

```
VSTATUS rioRouteAddEntry(VINT8 localport, VINT16 destid, VINT8 hopcount, VINT8 tableidx, VINT16 routedestid, VINT8 routeportno)
```

在指定的交换机路由表中添加一个条目，即将 routeportno 参数写入由 routedestid 选择的路由表条目。当 tableidx 为 0xff 时，指向的是一个全局路由表。

(2)在指定 RapidIO 交换机的路由表中读取一个条目

```
VSTATUS rioRouteGetEntry(VINT8 localport, VINT16 destid, VINT8 hopcount, VINT8 tableidx, VINT16 routedestid, VINT8* routeportno)
```

从指定的交换机路由表中读取一个条目，即从路由表读取由 routedestid 参数选择的路由表条目值并将其存入由 *routeportno 指针指示的位置。当 tableidx 为 0xff 时，指向的是一个全局路由表。

4.3.4 系统枚举函数

```
VSTATUS rioSystemEnumerate(VINT16 hostdevid)
```

(下转第 243 页)