一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法

石 炜[1,2]，孟金芳[2]

（1. 通信信息控制和安全技术重点实验室 浙江 嘉兴 314033；
2. 中国电子科技集团公司第三十六研究所 浙江 嘉兴 314033）

摘要：目前很多高实时性、高数据吞吐率、高灵活性的信号处理平台选择以 vxworks 为操作系统，PPC 与 FPGA 之间以高速 RapidIo 为互连的组织架构。很多时候，驱动设计者用对寄存器、内存直接访问的方式进行驱动的设计和开发，这种驱动设计方法管理混乱，不适宜驱动的模块化设计，不利于 FPGA 设备的抽象。针对这些问题，本文基于 VxWorks 的 vxBus 驱动开发模型，提出了一种 PPC 与 FPGA 间以高速 RapidIo 为互连的驱动设计方法，实现了 FPGA 设备的高抽象，大大有利于应用开发者对 FPGA 设备的透明调用，实现了驱动的模块化设计。通过在以 Mpc8641D 为主处理器，V7 FPGA 为预处理芯片的信号处理板上试验证明了该方法的可行性和有效性。

关键词：VxWorks；vxBus；RapidIo；驱动；模块化设计

中图分类号：TN332　　**文献标识码**：A　　**文章编号**：1674-6236(2015)24-0139-03

A driver design which highly-speed connects PPC and FPGA based on vxBus

SHI Wei[1,2], MENG Jin-fang[2]

(1. *Science and Technology on Communication Information Security Control Laboratory*, *Jiaxing* 314033, *China*;
2. *No.*36 *Research Institute of CETC*, *Jiaxing* 314033, *China*)

Abstract: Many signal processing platforms of highly real-time、high data throughput and high flexibility select the organizational structure which uses vxworks as the operating system and high-speed RapidIo as the connection of PPC and FPGA. A lot of times, driver designers develop programs in the way of accessing registers and memory directly. This method of driver design has chaotic management, disagrees with the modular programming of drivers, goes against the abstraction of FPGA devices. To solve these problems, based on the driver developing model of vxBus in vxworks, this paper proposes a driver design which is used in the condition of high-speed RapidIo as the connection of PPC and FPGA, which achieves the aim of high abstractness of FPGA devices. It is conductive for the application developer to call FPGA devices transparently, and realizes the modular programming of drivers. A series of experiments have been done on the signal processing platform of Mpc8641D as the main processor and V7 FPGA as the preprocessing chip, to prove the feasivility and effectiveness of the driver design method.

Key words: VxWorks; vxBus; RapidIo; driver; modular design

随着数字信号处理系统向着高实时性、高数据吞吐率、高灵活性的方向发展，目前很多信号处理平台选择以 PPC 为处理器，vxWorks 为实时操作系统，FPGA 进行信号预处理，两者利用 RapidIo 为通信链路的设计架构。在这种架构下，设计和实现好 PPC 与 FPGA 之间的 RapidIo 通信驱动尤为重要，通常驱动设计者采用直接寄存器或内存访问的方式进行驱动的设计和开发，这种开发方式虽然有开发简单直接，效率高的特点，但是驱动管理混乱，不适宜驱动的模块化设计，没有对 FPGA 设备进行较好的抽象，应用开发者无法透明和灵活调用。本文基于 VxWorks 的 vxBus[1]驱动开发模型对 PPC 与 FPGA 之间以高速 RapidIo[2-3]为互连的驱动设计进行了研究，实现了 FPGA 设备的高抽象，大大有利于应用开发者对 FPGA 设备的透明调用，同时满足信号处理平台 PPC 与 FPGA 之间通信的高实时性、高数据吞吐、高灵活性、高效率。

1　vxBus 驱动开发模型

1.1　vxBus 介绍

vxBus 是 vxWorks6.2 版本后推出的用于支持设备驱动的特有的驱动开发模型如图 1，其主要支持：1）支持对应设备的驱动匹配；2）提供驱动程序访问硬件的机制 3）支持驱动的模块化设计；4）支持在 WorkBench 开发环境中调用的组件，实现驱动的可配置。vxBus 在总线控制驱动服务程序的支持下，能在虚拟总线上发现设备并与之匹配，执行初始化工作，完

成驱动和硬件设备之间的正常通信。其中 vxBus 最核心的功能是组件功能,它把每个设备驱动程序和 vxBus 支持的模块都抽象成一个组件，所有的这些组件都可以单独在 Workbench[4]中进行配置。

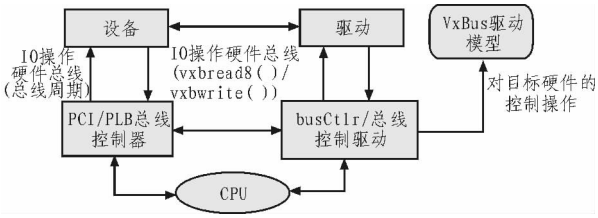

图 1　vxBus 系统关系图

Fig. 1　The vxBus system context diagram

1.2　vxBus 驱动组成

vxBus 下驱动的源程序由以下几个文件组成,详见表 1。

表 1　vxBus 下驱动的源程序组成

Tab. 1　The sources of vxBus Driver

源文件名	说明
Readme.txt	关于驱动的说明文件
makefile	驱动的编译规则
driverName.cdf	驱动的描述文件,包括的该驱动依赖的组件,驱动的位置,父目录以及子目录,以及 Workbench 下的说明信息等
driverName.dr	向 vxBus 进行注册的函数
driverName.dc	向 vxBus 进行注册的函数的声明
driverName.c	驱动的源程序文件包括:①包括设备自身数据结构定义②不同阶段的初始化函数③驱动方法结构声明④vxBus 注册结构⑤vxBus 注册函数

1.3　vxBus 开发步骤

vxBus 的开发步骤如图 2 所示。

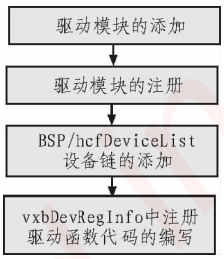

图 2　vxBus 驱动开发的基本步骤

Fig. 2　The step of vxBus driver design

1)驱动模块的添加

vxWorks 采用模块化机制管理各个功能单元，驱动也同样是由一个或多个模块组成，其驱动管理是依靠.cdf 文件完成,其添加就是对 driverName.cdf[1]的创建和编写。

2)驱动模块的注册

vxWorks 驱动的注册通过 driverName.cdf 和 driverName.dc 进行管理,其注册的实现是通过函数 vxbDevRegister 完成。其函数定义为 vxbDevRegister(structvxbDevRegInfo*pDevInfo)。

3)hcfDeviceList 设备链的添加

在 BSP 包中的 hwconf.c 文件中根据设备结构体添加新增设备的信息表，其设备信息结构体为:struct hcfDevice {char*devName;int devUnit;int busType;int busIndex;int count; const struct hcfResource*pResource;};

4)注册驱动代码的的编写

可知 vxWorks 的驱动信息结构体为 struct vxb DevRegInfo,其定义为:

Struct vxbDevRegInfo { Struct vxbDevRegInfo *pNext; UINT32? devID; UINT32? busID; UINT32? vxbVersion; char??? drvName [MAX_DRV_NAME_LEN +1]; struct drvBusFuncs * pDrvBusFuncs; struct vxbDeviceMethod * pMethods; BOOL (*devProbe)(struct vxbDev * pDevInfo0;struct vxbParams * pParamDefaults;};其中:

struct drvBusFuncs{void (*devInstanceInit)(struct vxbDev *);void (*devInstanceInit2 (struct vxbDev *);void (*devInstanceConnect)(struct vxbDev *);};

这一步就是添加函数 devInstanceInit、devInstanceInit2、devInstanceConnect 的代码内容,对于 vxBus 来说它只是给了一个驱动框架模型,使驱动程序更加统一。

2　MPC8641D 与 FPGA 的 RapidIo 通讯设计

2.1　平台介绍

本文中设计的平台对象如图 3 所示。Mpc8641d[5]与 V7[6]系列 FPGA 以 RapidIo 相连接，其驱动设计即为假设 FPGA 为驱动设备,通过驱动完成对设备的打开、关闭、配置、读、写操作。

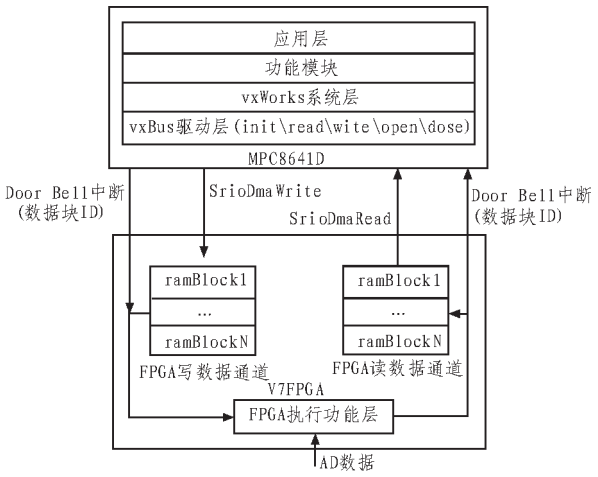

图 3　信号处理平台简图

Fig. 3　The diagram signal processing platform

2.2　RapidIo 通讯设计

RapidIo 支持 Nread\Nwrit[7]的读写、门铃、消息操作,为满足 PPC 与 FPGA 之间灵活快速响应的通讯设计,采用门铃响应与 Nread\Nwrite 读写相配合的实现方式，设计的控制和数据流图如图 2 所示。

驱动的主要操作包括 5 个部分，包括初始化 (init)、读(read)、写(wite)、打开设备(open)、关闭设备(close)。其中:

初始化包括根据配置指令进行不同的配置操作,块数据大小、数据块数、起始地址等。

读过程包括两部分,首先等待 FPGA 设备发起的门铃中断,再根据门铃信息提供的数据块号发起 Nread 操作读取定块数据。

写过程包括两部分,首先根据数据块号发起 Nwrite 操作完成定块数据的写操作,再向 FPGA 发起门铃操作告知 FPGA 有新的数据块到达。

打开设备包括申请资源、初始化等操作。

关闭设备包括释放资源、复位等操作。

3 驱动开发

按照本文第二节中介绍的 vxBus 驱动开发步骤和第三节设计的 MPC8641D 与 FPGA 的 RapidIo 通讯方案进行驱动开发。

3.1 驱动开发实例

1)驱动模块的添加

设计的驱动模块命名为 “FpgaDeviceDriver”,将其作为 BSP 包中硬件驱动的子模块,其依赖于 vxBus、PLB 两个子模块,其注册函数为 "FpgaDriverRegister",在 BSP 调用 hardWareInterFaceBusInit 函数后再调用,则编写 Fpga Device Driver.cdf 文件如下:

```
Component   FPGA_DEV_DRIVER {
  NAME    FPGA VxBus driver
  SYNOPSIS    FPGA VxBus driver provided by JEC
  _CHILDREN    FOLDER_DRIVERS
  REQUIRES    INCLUDE_VXBUS \
      INCLUDE_PLB_BUS
  INIT_RTN    FpgaDriverRegister();
  INIT_AFTER    INCLUDE_PLB_BUS
  _INIT_ORDER hardWareInterFaceBusInit
   _CHILDREN    FOLDER_DRIVERS }
```

2)驱动模块的注册

编辑 FpgaDeviceDriver.dc 内容为 IMPORT void FpgaDriverRegister (void);

编辑 FpgaDeviceDriver .dr 内容为:

```
#ifdef FPGA_DEV_DRIVER
void FpgaDriverRegister (void);
#endif
```

3)hcfDeviceList 设备链的添加

在 BSP 包中的 hwconf.c 文件中的 hcfDeviceList[]数组中添加:

```
#if defined (FPGA_DEV_DRIVER)
{“FpgaDeviceDriver”,0,VXB_BUSID_PLB,0,
FpgaDevNum,FpgaDevResources}
#endif
```

其中 FpgaDevResources 和 FpgaDevNum 分别定义为:

```
const struct hcfResource FpgaDevResources [] =
   {
        {    "irq",          HCF_RES_INT,    {(void*)
(INUM_TO_IVEC(8))} },
        { "irqLevel",    HCF_RES_INT, {(void *)1} },
        { " BlockSize ",  HCF_RES_INT, {(void *)16*1024 },
        { " Blocks",     HCF_RES_INT, {(void *)2} }
   };
#define FpgaDevNumNELEMENTS(FpgaDevResources)
```

4)注册驱动代码的的编写

这一步就是添加函数 FpgadevInstanceInit、Fpgadev Instance Init2、FpgadevInstanceConnect,FpgaDevInt,FpgaDevTx,FpgaDevRcv 的代码内容。

3.2 驱动开发实验

完成驱动开发后,FPGA 设备驱动很好的融入到 work Bench 的开发系统中,在建立 Image 工程时开发者能方便的进行动态配置如图 4 所示,并且设备驱动很好的融入到 vxWorks 文件系统中,其设备可以供应用程序员如操作普通文件一样 open/close/write/read,大大有利于应用程序开发者的透明调用不用了解设备中的细节参数。同时通过数据发送和接收测试其实时性虽有所损失但是其传输速率和响应时间仍然能满足实时性要求,延迟响应时间由直接中断响应的 15 μs 下降至 20 μs,但传输速率几乎没影响同样达到 800 MB/s。

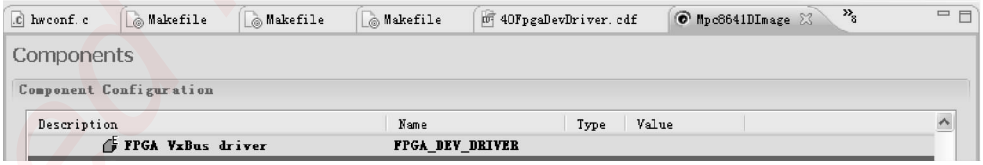

图 4 Workbench Image 工程配置图

Fig. 4 The diagram of Workbench Image config

4 结 论

在研究了 VxWorks 的 vxBus 驱动模型的基础上,结合自身平台的特点设计了 MPC8641D 与 FPGA 之间 RapidIo 通讯的驱动方案,经过开发和实验证明,基于 vxBus 的驱动开发相比直接对寄存器、内存直接访问方式的驱动的设计和开发更灵活,更具有可配置性,硬件设备更加抽象不失高实时性的同时更有利于应用程序开发者的透明调用。

(下转第 145 页)

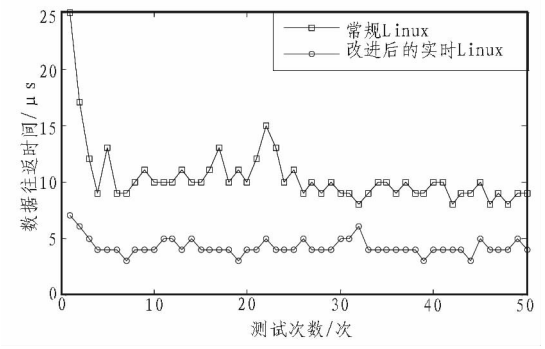

图 5　本地回环测试

Fig. 5　Local loopback test

略,与采用常规 Linux 内核及常规通信策略进行对比测试,以 5 min 为最大时间间隔随机向地面控制中心发送 4096Byte 数据,所传送的数据大小足以表示常规/故障信息。测试结果如图 6 所示。图 6 曲线中的断点代表本次发送失败。

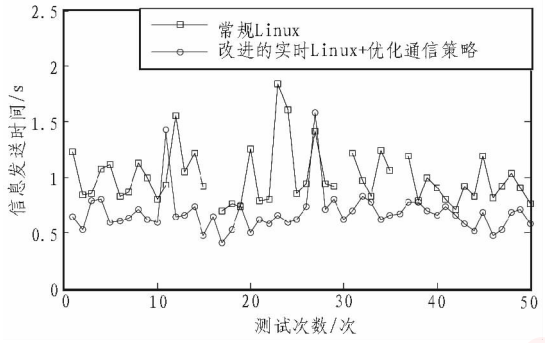

图 6　信息发送测试

Fig. 6　Information transmission test

5　结　论

本文对 Linux 操作系统进行了实时性,优化了城轨列车与地面控制中心的通信策略,并进行实验验证,得出如下结论:

1)采用 4G 移动通信网络完全能够满足高速通信的需求。

2)改进后的实时 Linux 操作系统在实时性能方面有了很大的提升,能够实时响应采集车辆数据采集设备的信息发送请求。

3)采用改进后的实时 Linux 操作系统与优化的通信策略,明显的缩短了信息发送时间,同时有效地保证了通信的可靠性。完全能够满足城轨列车与地面控制中心实时、高速、可靠的通信需求。

参考文献:

[1] 刘永超. 基于Web的高速列车故障诊断专家系统关键技术研究[D]. 大连:大连交通大学,2012.

[2] 邵位. 列车故障诊断专家系统智能技术研究[D]. 大连:大连交通大学,2014.

[3] 韩守谦, 裴海龙,王清阳. 基于Xenomai的实时嵌入式Linux操作系统的构建[J]. 计算机工程与设计,2011,32(1):98-102.

[4] BI Chun-yue,LIU Yun-peng,WANG Ren-fang. Research of key technologies for embedded Linux based on ARM [C]. International Conference on Computer Application and System Modeling,2011:1231-1240.

[5] 刘胜,王丽芳. 基于多核PC的Linux系统实时性改造[J]. 微电子学与计算机,2013,30(8):120-123.

[6] 严丽萍,宋凯,邓胡滨. 基于嵌入式应用的Linux内核实时性改进研究[J]. 计算机工程与设计,2011,32(1):121-124.

[7] 王涛. 基于交换式以太网的列车通信网络实时性研究[D]. 北京交通大学,2014.

[8] 李文,张建泽. 基于S3C2440的嵌入式Linux系统移植[J]. 化工自动化及仪表,2010,37(9):88-92.

[9] Michael Kerrisk. The Linux Programming Interface [M]. Beijing:Post & Telecom Press,2014.

(上接第 141 页)

参考文献:

[1] Wind River Systerms Inc.VxWorks 6.8 Device Driver Developer´s Guide [EB/OL]. (2009). [2014 -6 -16].http://www.windriver.com.

[2] 尹亚明,李琼,郭御风,等. 新型高性能RapidIo互连技术研究[J]. 计算机工程与科学,2004,26(12):85-107.

[3] 吴海燕. 基于RapidIo总线的信号处理平台设计[M]. 电子科技大学,2009.

[4] Wind River Systerms Inc.Wind River Workbench, 3.2 User´s Guide[EB/OL]. (2009). [2014-6-16]. http://www.windriver.com.

[5] Freescale. MPC8641D Integrated Host Processor Family Reference Manual [EB/OL].(2008).[2014-6-16].http://www.freescale.com.

[6] Xilinx. 7 Series FPGAs Overview[EB/OL]. (2011).[2014-6-16] http://www.xilinx.com.

[7] RapidIo Trade Assocition.RapidIo Interconnect Specification, Rev.1.3[EB/OL].(2005). [2014-6-16] .http://www.RapidIo.com.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计

12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计

9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备