一种双处理器间通信方案的研究与设计

朱伟伟[1],彭建朝[1],代俊锋[2]
(1. 北京工业大学 计算机学院,北京 100124;
2. 北京首科凯奇电气技术有限公司,北京 102299)

摘 要:目前,在 SOC 系统设计中存在着多处理器通信问题。传统的通信方式一般只适用于处理器功能相同的情况。文中提出了一种新的双处理器通信方案,适合一个处理器运行操作系统,而另一个处理器不运行操作系统的情况。通过对系统的需求进行分析,理解双处理器之间的通信内容,使用 Verilog HDL 实现定制的 IP 核。处理器都可对 IP 核进行读写,设置相应的标志位,使双处理器能相互通信。在系统中,存在两个相同的 IP 核,不仅提高了通信的速度,而且提高了通信的效率,使两个处理器都能很好的工作。

关键词:SOPC;μClinux;FPGA;双处理器
中图分类号:TP368.1 **文献标识码**:A **文章编号**:1673-629X(2013)09-0234-04
doi:10.3969/j.issn.1673-629X.2013.09.059

Design and Research of Communication Scheme in Dual-processor System

ZHU Wei-wei[1], PENG Jian-chao[1], DAI Jun-feng[2]
(1. College of Computer, Beijing University of Technology, Beijing 100124, China;
2. Beijing Shoke Catch Electric Technology Co., Ltd, Beijing 102299, China)

Abstract: Currently, there is a problem about multi-processor communication in the SOC system design. The traditional ways of communication generally apply only to the same processor. In this paper, present a new dual-processor communication solution, a processor to run the operating system, while another processor does not run the operating system case. Based on the demand of system analysis, understand the content of the communication between the dual-processor, using Verilog HDL customized IP core. The processor can read and write to the IP core, set up corresponding sign bit; the dual-processor can communicate with each other. In the system, there are two same IP cores, not only improve the speed of communication, but also the efficiency of the communication. The two processors can work well.

Key words: SOPC; μClinux; FPGA; dual-processor

0 引 言

随着应用领域的不断扩大,人们对处理器性能的要求也越来越高。以往单纯的通过提高处理器主频的方式已经满足不了日益增长的需要。

多处理器很好地满足了以上不足,其具有可配置性高、速度快等优点正在越来越多地受到欢迎。

在多处理器系统中,因为存在多个处理器,各个处理器实现的功能可能不一样,这样各个处理器在使用共享资源时,就很可能存在着一定的冲突,从而容易造成死锁等问题[1-2]。

1 概要设计

1.1 系统概要设计

文中所提出的双处理器架构,主要可以分为实时部分和非实时部分两部分。实时部分主要是用来运行对实时性要求高的作业,而非实时部分主要用来运行系统中对实时性要求不是很严格的作业。非实时部分从网络上接收数据,通过数据交换缓冲区传递给实时系统部分,完成非实时系统到实时系统的数据传输,整个过程要保证数据按序到达。实时系统部分接收系统的状态反馈数据后,将状态数据也要通过数据交换缓冲区发送到非实时部分,系统的原理图如图 1 所示[3-4]。

收稿日期:2012-11-16 **修回日期**:2013-02-21 **网络出版时间**:2013-04-22
作者简介:朱伟伟(1987-),男,硕士,研究方向为嵌入式技术;彭建朝,高工,研究方向为计算机体系结构。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130422.1721.020.html

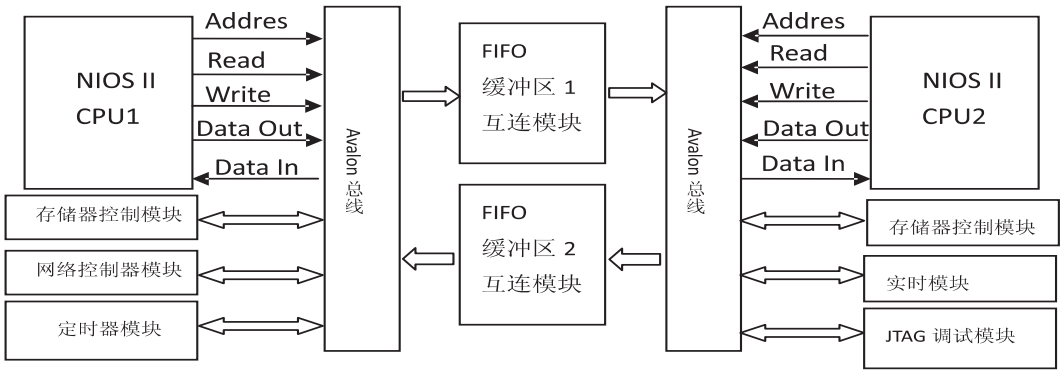

图 1　系统的结构框图

从图 1 可以看出,系统的非实时部分,即处理器 1 的部分,运行在 μClinux 操作系统下,主要的作用是组织数据,主要任务有两部分,一是从网络上接收数据,并将接收到的数据写入 FIFO 缓冲区 1 中,二是从 FIFO 缓冲区 2 中读取数据,再通过网络发送出去。而处理器 2 的部分,主要是系统的实时部分,主要完成的任务为两部分,一是从 FIFO 缓冲区 1 中接收数据,作为实时模块的运行数据,二是接收实时模块的反馈数据,将反馈数据写入 FIFO 缓冲区 2 中。该系统中存在着实时部分和非实时部分之间的数据通信,文中所设计的 FIFO 缓冲区互联结构,能够很好地保证数据传输相互不干扰,确保数据传输的完整性和实时性[5]。

1.2　双处理器之间通信方式

目前,基于 FPGA 的双处理器之间的通信按照通信方式来划分,主要有:共享存储器、邮箱核(mailbox)、串口通讯和 DMA。

共享存储器通信是一种两个处理器之间的阻塞通信方式。在双处理器系统中,可以通过共享片上 RAM 的方式来实现处理器之间的数据交互。为了防止死锁的发生,需要通过设置标志位或者中断信号的方式,实现双处理器的异步接收或发送。同时发送进程和接收进程必须在两个处理器上成对出现,只有当两个处理器上的发送进程和接收进程同时到来时,两个处理器才能共同退出当前通信进程,进入下一条指令。

邮箱核主要提供了两组互斥体,一个用于保证对共享存储器的唯一写访问,另外一个用于保证对共享存储器唯一的读访问。邮箱核结合片上 RAM,可以实现双处理器之间的数据交换。邮箱核加片上 RAM 的组合方式,适合于双处理器之间的单向数据传输,具有实时性好、速度快的优点,但是对于通信的数据结构有一些要求。

串口通讯在双处理器之间的数据交换主要是通过串口通信设备完成的。将系统中的处理器分为主设备和从设备,当主设备要读取从设备数据时,就向从设备发送一个读取的指令,从设备接收到指令后将数据发送到主设备中。主设备向从设备写入数据过程也是如此。这种通信方式,对于软件程序方面依赖性较小,但是传输模式单一,存在一定的延时。

DMA 方式主要是在数据量比较大的情况下,通过对数据整块的处理,能够有效地缩减数据通信占用处理器的时间[6-7]。

1.3　互联模块概要设计

通过对以上双处理器之间通信方式的优缺点分析,结合本系统通信的特点,设计出符合本系统的 FIFO 互联模块。在该系统中,使用的是定制 IP 核的方式来设计该模块,在 IP 核中设计相关的寄存器,设计出相关的读写控制逻辑。该 IP 核的设计主要分为以下几个方面。

第一是对于系统的非实时部分而言,要在操作系统的层次上,读取 FIFO 缓冲区数据和将数据写入 FIFO 缓冲区,保证数据的读取和写入正常。在操作系统中编写该 FIFO 缓冲区设备的驱动,在驱动中判断该设备的读写状态。

第二是对系统的实时部分而言,要在规定的时间内,将数据从该 FIFO 缓冲区中读取,同时将反馈数据在规定的时间内写入到 FIFO 缓冲区中。通过对该 IP 核中定义的控制寄存器的读取,实时地判断该 FIFO 设备的状态,保证及时的读取或写入数据。

第三就是要实现对该 FIFO 缓冲区的互斥访问,保证每次读取或者写入的数据都是完整的,确保数据的准确。通过对缓冲区的大小进行设定,当读或者写完成时,设置控制寄存器中相应的标志位,达到互斥访问的目的[8]。

2　详细设计

2.1　IP 核的设计

该 IP 核主要的功能一是对写入的数据进行缓冲,二是对输入的数据进行分隔,保证每次读写的数据都是完整的一帧。该 IP 核在 Quartus II 环境中顶层模块如图 2 所示。

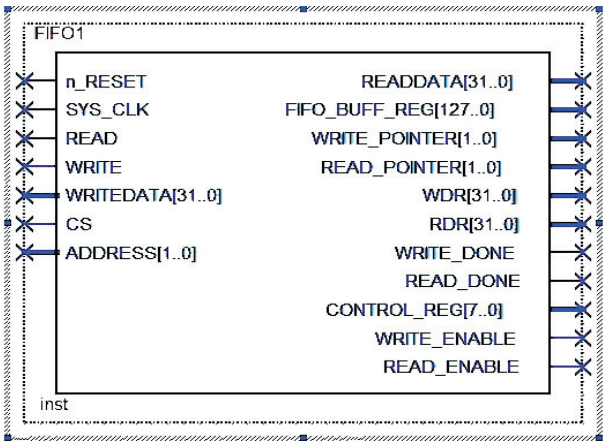

图 2 FIFO IP 核模块的封装

该 FIFO 的缓冲区大小为 128 位,宽度为 32。使用了一个读指针 READ_POINTER 和一个写指针 WRITE_POINTER 来控制 CPU 对该缓冲区的读写,读写指针大小都设置为 2。该 FIFO 的读写数据模型如图 3 所示。当对该缓冲区进行读或者写操作时,相应的指针发生变化。当读或写指针达到一定的条件时,就设置相关寄存器的状态。

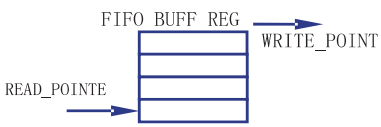

图 3 FIFO 的读写数据模型

当从该缓冲区中读取数据完成时,就将 READ_DONE 置 1,同时将 READ_ENABLE 置 0,将 WRITE_ENABLE 置 1,从而完成读操作,保证读取数据的完整性。同理,当写数据完成了,将 WRITE_DONE 置 1,同时将 READ_ENABLE 置 1,将 WRITE_ENABLE 置 0,完成写操作。这样就可以控制 CPU 对缓冲区的读写,保证数据的唯一性。

可以看出通过对读写标志位的判断,可以达到对该 FIFO 状态的判断,从而控制读写数据的时机。整个状态的判断和控制是通过控制寄存器(CONTROL_REG)来实现的,在控制寄存器中判断 FIFO 可读或者可写状态是由 CONTROL_REG[7:6]来控制的,而向 FIFO 中每次写入或读取数据的大小是由 CONTROL_REG[1:0]来控制的,系统可以随时读取该寄存器来确定 FIFO 的读写状况。该 IP 核的控制寄存器结构如图 4 所示。

7	6	5	4	3	2	1	0
WRITE_ENABLE	READ_ENABLE	0	0	0	0	DATA1	DATA2

图 4 控制寄存器结构图

在 Quartus II 下进行功能仿真如图 5 所示。

从图 5 中可以看出,IP 核复位后,FIFO 的状态是可写的。此时,读取寄存器的状态位 0x80,表明该 FIFO 是可写的。将 0x2 写入到控制寄存器(CONTROL_REG[1:0])中,就确定了 FIFO 中每帧的大小。读取寄存器,为 0x82,表明数据已经写入到控制寄存器中。然后向 FIFO 中写入数据 0x12345678 和 0x87654321 后,可以看到 WRITE_DONE 置 1,READ_ENABLE 被置 1,WRITE_ENABLE 被置 0,同时写指针 WRITE_POINTER 也是正确的,此时读取寄存器,为 0x42,表明写操作完成,数据可读。

在后面对 FIFO 进行读操作可以看出,各个标志位和读写指针都是正确的,可以看到读出的数据为 0x87654321 和 0x12345678。此时,读取寄存器中数据,为 0x82,表明数据可写。

2.2 FIFO 的驱动的设计

在实时部分和非实时部分之间通过 FIFO 传递数据,非实时部分运行的是 μClinux 操作系统,需要对 FIFO 编写驱动。在 μClinux 下开发 FIFO 设备的驱动和常规 Linux 下开发驱动流程一样。

在 Linux 系统中,设备的驱动程序被组织为一组完全不同任务的函数的集合,通过这些函数使得设备操作犹如文件一般。在应用程序看来,硬件设备只是

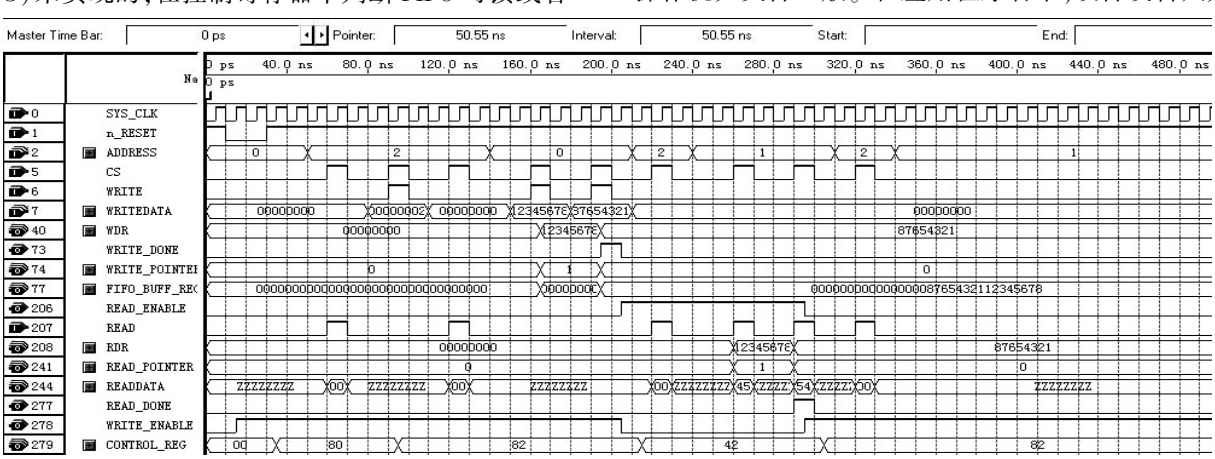

图 5 FIFO IP 核的功能仿真

一个设备文件,应用程序可以像操作普通文件一样对硬件进行操作。例如 open( )、close( )、read( )、write( )等等。Linux 主要将设备分为两类:字符设备和块设备。字符设备是指设备发送和接收数据以字符的形式进行;而块设备则以整个数据缓冲区的形式进行。文中 FIFO 设备属于字符设备,其设备驱动主要定义以下几个函数:

```
int fifo_open(struct inode *node,struct file *filp);
int fifo_release(struct inode *node,struct file *filp);
ssize_t write_data(struct file *filp,const char _user *buf,size_t count,loff_t *ppos);
ssize_t read_data(struct file *filp,char _user *buf,size_t count,loff_t *ppos);
static void fifo_setup_cdev(struct fifo_dev *dev,int index);
static int _init fifo_init(void);
static void _exit fifo_exit(void);
```

当该 FIFO 设备启动的时候,先初始化,调用 fifo_init( ),在 fifo_init( )函数中将调用 fifo_setup_cdev( )函数。初始化函数向系统注册文件操作方法 fifo_fops。fifo_fops 是一个结构体,它包含文件操作方法的函数指针。定义如下:

```
static const struct file_operations fifo_fops =
{
.owner=THIS_MODULES,
.write=write_data,
.read=read_data,
.open=fifo_open,
.release=fifo_release,
};
```

初始化完成后,对设备的读写主要是通过调用 write_data( )和 read_data( )来实现的[9-10]。

μClinux 针对无 MMU 的处理器做了移植,与 Linux 使用虚拟内存不同,μClinux 采用真实地址方式,这样操作硬件设备非常方便。SOPC 系统把存储设备,IO 映射为统一编制的存储设备[11]。

3　自定义 IP 核在双处理器中的应用

3.1　双处理器平台的搭建

根据设计的需要,构建一个完整的 SOPC 系统后,才能真正实现双处理器的设计。在 SOPC Builder 中加入 CPU1 和 CPU2,一个 Timer,定制的 FIFO 组件以及 sdram、flash、JTAG 等外围设备。其中非实时部分的 CPU1 选择 NIOS II/f,复位向量设置为 flash,异常向量设置为 sdram。另外连接 CPU1 的 Timer 要选择全功能型(Full-featured),并将该 Timer 的优先级设置为 0。而实时部分的 CPU2 也要选择 NIOS II/f,复位向量设置为 flash,异常向量设置为 sdram。

3.2　自定义 IP 核功能验证

将定制的 IP 组件添加到系统中,在 CPU1 上运行 μClinux 操作系统,CPU2 的程序是在 NIOS II IDE 环境下运行。在 CPU1 上运行的程序首先使用函数 open( )打开该 FIFO 设备,然后通过读函数 read( )和写函数 write( )读写数据,从而实现 CPU1 读取和发送数据。而在 NIOS II IDE 环境下,使用查询方式,来读取该定制 FIFO 数据,使用函数 IORD( )控制 CONTROL_REG 的状态,如果为可读的,则执行读操作,如果为不可读,则什么都不执行。写数据也是如此。经过验证,数据都能按时按序到达,该 IP 核很好地满足了设计的需要。

4　结束语

文中提出的设计,是在传统的双处理器通信技术的基础上,对传统双处理器通信技术流程进行改进。针对不同的业务逻辑,提出不同的解决方案。所设计的 IP 核很好地满足了实时系统和非实时系统之间的通信需要。整个系统的设计体现了自顶向下的模块化设计思想,同时还充分利用了 SOPC 技术的优势,具有很好的可扩展性和易裁剪性,具有较好的应用前景。

参考文献:

[1] 李兰英,李霄燕. 基于 Nios II 的 SOPC 多处理器系统设计方法[J]. 单片机与嵌入式系统应用,2007(3):18-21.

[2] 王卫源,戴紫彬,钱育蓉. Nios II 多处理器系统方案设计[J]. 微计算机信息,2007,23(7-2):96-97.

[3] Wang Qingbo. Multi-core Architecture on FPGA for Large Dictionary String Matching[C]//Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. [s. l.]:[s. n.],2009.

[4] 宋秀兰,吴晓波. 多处理器通信机制设计[J]. 浙江工业大学学报,2010,38(4):426-429.

[5] 李　柱,孟令军,彭晴晴,等. 基于 SOPC 双核的高速数据采集系统[J]. 化工自动化及仪表,2010,38(7):856-858.

[6] Benkrid K,Benkrid A. Software and Hardware Architectures for FPGA-based Image Processing[M]. [s. l.]:VDM Publishing,2010.

[7] Weltzin C,Bell I. Keep up with multiple cores[J]. Electronics Weekly,2009(2385):253-257.

[8] 蒋巍泉,王　前,吴淑泉,等. 基于 NiosII 的 μClinux 研究与应用[J]. 科学技术与工程,2006,6(8):1069-1072.

[9] 宋宝华. Linux 设备驱动开发详解[M]. 第 2 版. 北京:人民邮电出版社,2010.

[10] 刘金祥,王京仁. μClinux 在实时监控系统中的应用研究[J]. 计算机技术与发展,2009,19(3):220-222.

[11] Amin A F M,Aris I,Abdullah R S A R,et al. Embedded System Implementation on FPGA System With μCLinux OS[J]. Materials Science and Engineering,2011,17(1):156-159.

一种双处理器间通信方案的研究与设计

作者： 朱伟伟， 彭建朝， 代俊锋， ZHU Wei-wei， PENG Jian-chao， DAI Jun-feng

作者单位： 朱伟伟,彭建朝,ZHU Wei-wei,PENG Jian-chao(北京工业大学 计算机学院,北京,100124)
， 代俊锋,DAI Jun-feng(北京首科凯奇电气技术有限公司,北京,102299)

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2013(9)