基于 Petalinux 的 Socket 网络通信系统设计与实现

杨 谢,武传华,路后兵,杨 标
(合肥电子工程学院,安徽 合肥 230037)

摘 要:介绍了一款针对 MicroBlaze 软核处理器特别开发的嵌入式操作系统 Petalinux,成功地实现了其在 ML402 开发板上的移植,并且在该系统上实现了基于 TCP/IP 协议的套接字接口 Socket 的网络通信。

关键词:FPGA;MicroBlaze 软核处理器;Petalinux 移植;Socket 网络通信
中图分类号:TP393 **文献标识码**:A **文章编号**:1674-7720(2012)08-0051-03

Design and implementation of Socket communication system based on Petalinux

Yang Xie, Wu Chuanhua, Lu Houbing, Yang Biao
(Hefei Electronic Engineering Institute, Hefei 230037, China)

Abstract: This paper introduces a kind of special embedded system Petalinux which developed aiming at MicroBlaze, successfully porting it at ML402 development plank, and realized the Socket network communication that based on the TCP/IP protocol.

Key words: FPGA; MicroBlaze processor; Petalinux porting; Socket communication

嵌入式系统是为特定目的而构建的一类计算机设备。该设备具有体积小、功耗低、可靠性稳定、高度自动化、响应速度快等特点,特别适合要求实时和多任务的体系[1]。Petalinux 是由 PetaLogix 公司专门为在 Xilinx FPGA 的 MicroBlaze 软核处理器上运行而开发的嵌入式 Linux。Petalinux 发布的版本中包含定制的 Linux2.4/2.6 内核原码、U-boot 内核编码、相关的开发工具以及开发板参考硬件平台配置,极大地方便了开发人员的使用,缩短了产品的开发周期。

对于如何在嵌入式系统上实现远程网络通信这一问题,本文给出了一种基于 Xilinx 开发板 ML402 的嵌入式网络通信系统的设计与实现方案,成功实现了开发板与 PC 机的实时网络通信。

1 硬件工程设计

1.1 底层硬件平台的选取与设计

本设计方案采用 Xilinx EDK 10.1 在 ML402 开发板搭建一个最简化的硬件平台,结构如图 1 所示。

图中各部件在 FPGA 内部以 IP 核的形式构建并连接,系统以带有 32 bit MicroBlaze 软核的 FPGA 作为控制中心,SysACE 用于存放文件系统和应用程序配置文件;INTC 用来实现中断控制;串口可在调试时输出系统的运行信息;以太网控制器用来实现以太网功能;

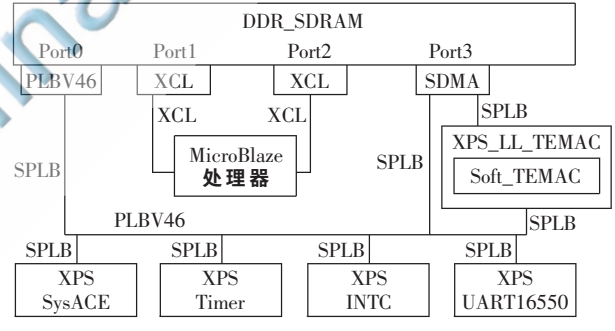

图 1 硬件平台结构图

DDR_SDRAM 通过 XCL 总线与处理器相连,用于对片外存储器进行访问[2]。

1.2 软件平台的配置

在移植 Petalinux 之前,必须配置 BSP(Board Support Package)。所谓 BSP,就是为给定的板子提供特定操作系统支持的代码。介于主板硬件和操作系统之间,属于操作系统的一部分,主要目的是为了支持操作系统,使之能够更好地运行于硬件主板。

Xilinx EDK 已经包括相应的 BSP 产生器,因此,第一步只需要把解压的 Petalinux 文件夹下的 hardware/edk_user_repository/PetaLogix/bsp/petalinux_v1_00_b 文件夹拷贝到 EDK 文件夹下的 \sw\lib\bsp 目录下进行相应的

配置即可。接下来打开已建立的硬件工程，进行软件平台配置，点击 Software 菜单，启动 Software Platform Setting。系统会弹出软件平台的配置窗口，可以看到共有 3 个可配置项——Software Platform、OS and Libraries 和 Drivers。右方的窗口为可配置选项的参数。首先对 Software Platform 进行配置，点击 Software Platform，在窗口右侧可以看到可配置参数，包括两个子窗口，其一是 processor parameters，其中包括处理器主频信息、交叉编译器等选项。其中，extra compiler flag 指定了在生成 BSP 与库的过程中，交叉编译器所用的编译标志，archiver 和 compiler 分别指定了生成 BSP 与库所用的工具链，在这里只需保持默认即可。在 OS and Library settings 子窗口中打开 OS 的下拉菜单，选择 Petalinux，版本只有 1.00.b，如果第一步没有完成，则在点开 OS 的下拉菜单后，没有 Petalinux 选项。

完成上一步之后，选中 OS and Library 可配置选项，这里主要是针对开发板对 μClinux 的 BSP 进行配置，包括 Flash 与 Memory 以及输入输出调试端口的配置，在这里主要对以下参数进行修改：

Lmb memory：dlmb_crtlr

Main memory：DDR_SDRAM

Stdin：RS232_Uart

Stdout：RS232_Uart

最后点击 OK，退出，基于 Petalinux 的 MicroBlaze 软件平台配置完成。下一步是根据软件平台的配置生成针对 MicroBlaze 处理器的 BSP 与库，使 Petalinux 与开发板的信息交互成为可能。进入 EDK 的 Software 菜单，点击 Generate BSP and Libraries，系统会自动生成板级支持包与库。之后就可以在 microblaze_0/libsrc/petalinux_v1_00_b 文件夹下生成 auto-config.in 文件[3]。

1.3 Petalinux 操作系统的移植

软件平台完成后需要对内核进行配置，嵌入式系统开发一般采用交叉编译的方法，通过 PC 机对内核和应用程序进行编译，具体步骤如下：

(1)将工程所在目录复制到 Petalinux 解压目录下的 ~/hardware/user-platforms 目录下。

(2)进入 Petalinux 解压目录，运行 source ./settings.sh 命令，设置 Petalinux 环境变量。

(3)进入~/software/petalinux-dist 目录，运行 petalinux-new-platform-k 2.6-v Xilinx-p ml402 新建用户平台；其中-v 后缀为 FPGA 的生产厂商，-p 后缀为工程使用的 FPGA 开发板名称，-k 为配置内核的版本。然后运行 make menuconfig 命令，进入 Vendor/Product Selection 选项，选择相应的平台，退出并保存。

(4)进入工程所在文件夹，运行 petalinux-copy-auto-config 命令，将 libgen 生成的 microblaze_0/libsrc/Kconfig.auto 和 autoconfig.in 转换成 linux 格式，并拷贝到当前活跃的 platform 下(例如 software/linux-2.6.x/arch/Microblaze/platform/ml402)。它是根据在 make menuconfig 中选择的 vendor/platform 来拷贝的。

(5)内核的配置与编译

搭建的底层硬件平台的不同决定了系统内核的区别，参考文献[2]中薛慧敏针对不同情况给出了较为详细的配置过程，可作为参考，在此不再赘述。

(6)Xmd 下载启动

Xmd 是 Xilinx EDK 提供的调试工具，可以使用该工具对 EDK 开发的工程进行调试。使用该方法下载 image 文件，启动 Petalinux 后，通过串口超级终端可以看到系统启动过程。

2 网络通信程序的开发

Petalinux 移植成功后，就可以使用 petalinux-new-app 命令在其上建立软件应用工程，进行软件应用的开发。新建的软件应用工程放在~/petalinux/software/user-apps。

2.1 软件应用工程的建立

在 petalinux-dist 文件夹下输入命令：petalinux-new-app petaserver，其中 petaserver 为应用工程名称。创建成功后，在 user-apps 文件夹下新建了以工程名称 petaserver 为名称的文件夹，里面包括.C 的应用程序、Makefile 的编译规则和 readme 的帮助文件。

2.2 Socket 程序的创建

进入新建的应用工程文件夹，输入命令：gedit petaserver.c，打开文本编辑器，对.C 的应用程序进行编辑。本应用工程主要是作为网络服务器，接收客户端传送的命令，消息经过处理后再回馈给客户端，主要流程如图 2 所示。

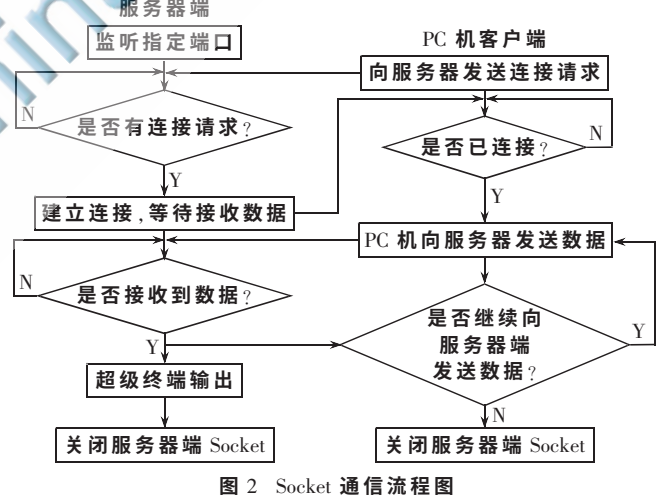

图 2 Socket 通信流程图

现行的网络协议中 TCP/IP 协议是最通用的一个，因此，本程序也使用该协议实现网络的互联[4]。

Socket 接口是 TCP/IP 网络的 API，Socket 接口定义了许多函数或例程，程序员可以用它们来开发 TCP/IP 网络上的应用程序。网络的 Socket 数据传输是一种特殊的 I/O，Socket 也是一种文件描述符。Socket 具有一个类似于打开文件的函数调用 Socket()，该函数返回一个整型的 Socket 描述符，随后的连接建立、数据传输等

操作都是通过该 Socket 实现的。常用的 Socket 类型有两种：流式 Socket (SOCK_STREAM) 和数据报式 Socket (SOCK_DGRAM)。流式是一种面向连接的 Socket，针对于面向连接的、无差错的、发送先后顺序一致的、包长度不限和非重复的 TCP 服务应用；数据报式 Socket 是一种无连接的 Socket，对应于无连接的 UDP 服务应用，主要以独立的数据报进行网络传输，数据报的最大长度为 32 KB，传输不保证顺序性、可靠性和无重复性，它通常用于单个报文传输或可靠性不重要的场合。根据以上特点，本应用选择流式 Socket[1]。

(1)服务器端 petaserver.c 主要代码如下：

```
{
    if((sockfd=socket(AF_INET,SOCK_STREAM,0))==-1)
        //创建套接字 Socket 函数可以调用 Socket 函数，
            该函数返回一个类似于文件描述符的句柄
    {
        fprintf(stderr,"socket error! \n");
        exit(1);
    }
    if(bind(sockfd,(struct sockaddr *)&my_addr,sizeof(struct
        sockaddr))==-1)  //Bind 函数将 socket 与本机上
            的一个端口相关联，在该端口监听服务请求
    {
        fprintf(stderr,"bind error! \n");
        exit(1);
    }
    if(listen(sockfd,10)==-1)          // Listen 函数将一个
                        套接字转换为被动倾听套接字
    {
        fprintf(stderr,"listen error! \n");
        exit(1);
    }
    if((connfd=accept(sockfd,(struct sockaddr *)&their_addr,
                                        &sin_size))==-1)
                        //Accept 函数从倾听套接字的
                完成连接队列中接收客户端连接请求
    {
        fprintf(stderr,"accept error! \n");
        exit(1);
    }
    printf("server:got connection from %s\n",inet_ntoa
                                        (their_addr.sin_addr));
if(send(connfd,"successfully connect\n",20,0)==-1)
                    //send 函数用来控制对套接字的写操作
    {
        fprintf(stderr,"send error! \n");
        exit(1);
    }
                    //显示连接成功，开始接收客户端信息
    while((len1=recv(connfd,str,100,0)) > 0)
                    //recv 函数用来控制对套接字的读操作
    {
        len1=recv(connfd,str,100,0);
        printf("\n 收到字符数:%d\n",len1);
        str[len1]=0;
        printf("Received from client:%s\n",str);
    }
    close(sockfd);              //Close 函数用来关闭一个
                                套接字描述符
}
```

(2)启动服务器

程序编写好后，重新编译，生成 image.bin 文件，下载该文件。

打开超级终端，系统启动后输入用户名与密码，进入 petalinux 系统执行下列命令：

ls /bin 回车

petaserver 回车

可以看到服务器端启动语句输出：

SOCKET: Creating socket..done.

SOCKET: start bind socket..done.

SOCKET: start listen..done.

这时打开 PC 机客户端，客户端使用成都众山科技有限公司提供的 TCP/UDP Socket 调试工具 V2.3，点击 TCP Client 按钮，在弹出的窗口中输入服务器 IP 地址：192.168.0.10，端口:8000，点击连接按钮，可以看到当 PC 机客户端向开发板上服务器端发出连接请求时，服务器端通过 PC 机超级终端输出：

SOCKET:start accept..server:got connection from 192.168.0.1

Successfully connect

同时在 PC 机 socket 客户端回显:Successfully connect；

客户端向服务器发送 hello petalinux；

超级终端显示服务器端已经接收到客户端发来的信息，屏幕输出为:15(接收到的字节长度)和 hello petalinux(接收到的内容)。

至此说明客户端与服务器端完成了网络的连通，后续就可以接入外围设备对系统进行进一步的开发与完善了。

本文简单介绍了基于 Petalinux 的嵌入式系统的开发与移植过程以及客户端与服务器端 Socket 的创建过程，通过在客户端及服务器端创建 Socket 实现了 PC 机与 Petalinux 操作系统的实时网络通信，实验证明 Petalinux 的稳定性和实时性较好，为接下来在该系统上进行网络应用程序开发打下了基础，能够满足进一步实验需求。

参考文献

[1] 欧文盛.ARM 嵌入式 Linux 应用实例开发[M].北京：中国电力出版社，2008.

[2] 薛慧敏,武传华,路后兵,等.基于 MicroBlaze 的 Petalinux 嵌入式操作系统移植[J].微计算机信息,2011,27(8):109-110.
[3] 薛小刚,葛毅敏.Xilinx ISE9.x FPGA/CPLD 设计指南[M].北京:人民邮电出版社,2007.
[4] IT 同路人.Linux 标准学习教程[M].北京:人民邮电出版社,2008.

(收稿日期:2011-12-26)

作者简介:

杨谢,男,1987年生,硕士研究生,主要研究方向:嵌入式系统开发。

武传华,男,1963年生,教授,研究生导师,主要研究方向:软件无线电。

路后兵,男,1979年生,讲师,主要研究方向:嵌入式系统开发。