

文章编号:1007-2780(2015)06-0979-08

基于 ARM+FPGA 的影像交互与显示系统设计

吴 迪*, 李丙玉, 王晓东
(中国科学院 长春光学精密机械与物理研究所, 吉林 长春 130033)

摘要:为促进航空测绘信息获取的数字化、一体化、实时化,本文利用 FPGA(Field-Programmable Gate Array,即现场可编程门阵列)并行处理的优势结合 ARM 处理器低功耗高性能的特点,基于 ARM+FPGA 的双核硬件架构实现了影像的交互与显示。该系统以 Linux 操作系统为软件开发平台,以 ARM11 嵌入式处理器为硬件核心、FPGA 作为协处理器,采用 FPGA 片内 FIFO(First Input First Output,即先进先出存储器)作为 ARM 处理器与 FPGA 之间的高速通信桥梁,针对 Linux 2.6.36 内核完成了对 FPGA 设备的驱动设计,并基于 Qt 图形用户界面实现了影像的实时显示。测试结果表明,ARM 处理器与 FPGA 之间能够实现 VGA(640×480)图像的高速交互,帧率可达 26 帧/s,最大传输带宽为 182 Mbps。该系统不仅体积小、功耗低、成本低,而且稳定性好、功能强,能够满足航空遥感摄影系统的实时性要求。

关键词:ARM; 嵌入式 Linux; FPGA; 设备驱动; 影像显示
中图分类号:TP368 **文献标识码:**A **doi:**10.3788/JYXS20153006.0979

Design of image interaction and display system based on ARM+FPGA

WU Di*, LI Bing-yu, WANG Xiao-dong
(Changchun Institute of Optics, Fine Mechanics and Physics,
Chinese Academy of Sciences, Changchun 130033, China)

Abstract: To promote the integration of mapping information obtained, image interaction and display are realized based on the dual-core ARM+FPGA hardware architecture in this paper by taking full advantage of the parallel processor FPGA (Field-Programmable Gate Array) and the high-performance ARM processor. The real-time image display is realized based on Qt graphical interface and the driven design of FPGA FIFO (First Input First Output) is completed in the Linux 2.6.36 kernel which uses Linux operating system as the software platform, ARM11 as the core of embedded hardware processor and FPGA as the coprocessor. The test results show that it could achieve high speed VGA (640×480) format image interaction between ARM and FPGA with the maximum frame rate of 26 fps and the maximum transmission bandwidth of 182 Mbps. This system not only has the advantages of small volume, low power consumption, low cost, but also has good performance of stability and functionality. Moreover, it could meet the real-time requirements for the aerial remote sensing system.

Key words: ARM; embedded Linux; FPGA; device driver; image display

收稿日期:2015-03-23;修订日期:2015-04-22.
基金项目:国家重大科学仪器设备开发专项(No.2012YQ160185)
Supported by National Key Scientific Instrument and Equipment Development Projects of China (No. 2012YQ160185)
* 通信联系人,E-mail:wudihope@163.com

1 引 言

目前,随着嵌入式技术与数字技术的高速发展,ARM+FPGA 双核硬件架构倍受青睐,该架构不仅功能分明而且结构紧凑。结合 ARM-Linux 和 FPGA 各自特点的系统在数字系统设计中得到了广泛应用。

FPGA 能够实现复杂的高速逻辑。它克服了定制电路的不足,可以在不更改外围电路的情况下,通过对 FPGA 的软件编程实现特定的系统功能。但 FPGA 在分析控制及驱动等方面较为薄弱。ARM 处理器弥补了 FPGA 的上述不足,它是目前嵌入式领域使用最广泛的处理器,具有专用性强、功耗低等优势,且嵌入式 Linux 操作系统具有开放源代码、稳定性强、可裁剪等优点,能够完成对大规模复杂程序的分配调度和协调控制。

该架构中 FPGA 主要负责高速数据处理,ARM 处理器负责实现人机交互、界面显示及外部通信功能^[1]。传统的 I²C、SPI 等串行总线通信仅能实现 ARM 与 FPGA 之间的低速通信,本系统采用 FPGA 片内 FIFO 实现二者之间的高速数据通信,且 Linux 2. 6. 36 版本内核在内存管理和设备驱动方面表现得十分出色。本文基于 ARM-Linux 软硬件架构完成了与 FPGA 数据通信的接口设计与驱动设计,并采用 Qt GUI 设计了人机交互图形界面,最终实现高速影像信息的实时显示。

2 系统硬件架构

2.1 ARM+FPGA 双核架构

系统以 ARM11 处理器 S3C6410 为硬件核心,负责影像的交互与显示。采用 DDR RAM 和 NAND Flash 作为系统存储器、SD 卡存储图像文件,1. 2 V 外部电源供电,最大主频 667 MHz,且内置图像加速器并扩展了 SIDI 功能,能够满足较高图像处理性能的要求,4. 3 in LCD 液晶屏用以显示实时影像,并配备了方便人机交互的触摸屏。

FPGA 采用 Altera 公司的 Cyclone IV 系列的 EP4CE40F23C8 芯片。它采用 FBGA 封装,具有 484 个引脚、39 600 个 LEs(Logic Elements,逻辑单元)、126 个 18×18 乘法器、总 RAM 位数

1 161 216 bits、329 个用户 I/O,是一款高性能、低功耗、低成本的现场可编程门阵列,能够满足本系统对影像处理的要求^[2]。

嵌入式 ARM 处理器接收 FPGA 从 Camera Link 接口获取并处理的影像,并将该影像实时显示在 LCD 液晶屏上。系统硬件架构如图 1 所示。

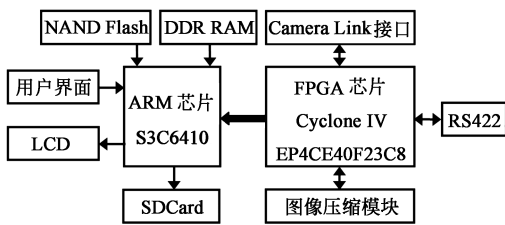


图 1 系统硬件架构图

Fig. 1 Structure diagram of the system hardware

2.2 ARM 与 FPGA 接口设计

ARM 处理器与 FPGA 之间的数据通信主要采用串行传输和并行传输两种方式。采用 I²C 接口或者 SPI 接口的串行通信方式传输速度较慢;而并行方式通过总线连接,在 ARM 端口配置地址总线、数据总线和控制总线,并和 FPGA 相应的信号相连,这种传输方式具有速度快且传输稳定的优势。本系统采用总线方式连接 EP4CE40F23C8 和 S3C6410,二者的硬件连接图如图 2 所示。

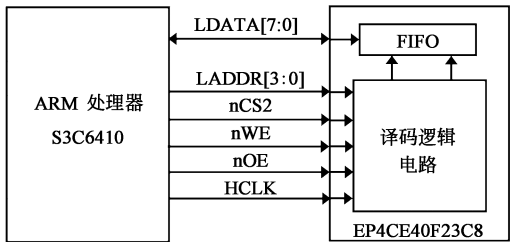


图 2 ARM 与 FPGA 硬件连接图

Fig. 2 Diagram of ARM and FPGA hardware connection

在图 2 中,EP4CE40F23C8 作为外部存储器挂载在 S3C6410 的存储器总线上。将 S3C6410 的数据总线 LDATA[7 : 0] 连接到 FPGA 片内 FIFO,FIFO 常用作数据缓存,和 RAM 不同的是,FIFO 不需要关心读写地址,从而简化操作。此外,还需将 S3C6410 的地址总线、片选信号线、读/写使能信号线及时钟信号线分别与 EP4CE40F23C8 的相应引脚相连。

3 Linux 系统下的 FPGA 驱动设计

由于 Linux 内核中并没有 FPGA 片内 FIFO 的设备驱动,因此,为实现 ARM 处理器与 FPGA 之间的数据通信,须在 Linux 系统下完成对 FPGA 片内 FIFO 的驱动开发。

Linux 设备驱动程序是连接底层设备和上层应用程序的纽带。Linux 系统将设备以设备文件的形式挂载在根文件系统的 /dev 目录下,系统对设备的操作可简化成对设备文件的操作,如打开 open()、关闭 close()、读 read() 和写 write() 等^[3]。驱动设计的层次结构图如图 3 所示。

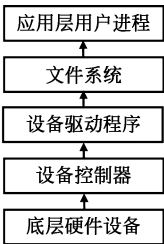


图 3 驱动设计的层次结构图
Fig. 3 Diagram of the driver hierarchy

Linux 系统具有以下 3 种设备:包括字符设备、块设备及网络设备^[4]。Linux 把 FPGA 的片内 FIFO 视为特殊的字符型设备,即杂项设备。由于硬件连接上将 FPGA 挂载在 S3C6410 的存储器总线上,用户空间通过内存映射方式实现对 FPGA 的访问。因此,上层应用程序可以直接访问设备内存,从而大大提高数据传输速度。

本文在 Linux 系统下实现了 FPGA 的设备驱动,总结得到以下步骤:

- (1)编写设备驱动程序,并添加到内核源码目录;
- (2)在 Kconfig 文件中添加上述设备驱动程序所对应的编译配置选项;
- (3)通过内核配置工具 menuconfig,将该硬件配置编译为模块;
- (4)把内核配置选项和真正的硬件驱动联系起来,在 Makefile 文件中添加相应的编译条目;
- (5)执行 #make modules 编译生成动态加载模块;
- (6)将编译好的驱动模块下载到目标机,执行命令 #insmod mini6410_fpga.ko 动态加载驱动

模块;

(7)交叉编译测试程序,验证驱动设计是否成功。

3.1 文件操作接口

Linux 系统开机启动后 FPGA 设备驱动的加载流程如图 4 所示。

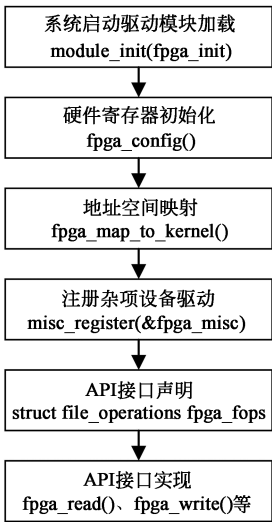


图 4 FPGA 驱动加载流程图
Fig. 4 Flowchart of FPGA driver loading

首先,定义 file_operations 类型的结构体变量 fpga_fops 用于存储内核驱动模块对设备操作的函数指针,包括 open、release、read、write 以及 ioctl,分别对应应用用户空间的接口函数^[5]。

```
static struct file_operations fpga_fops = {
    .owner = THIS_MODULE,
    .open = fpga_open,
    .release = fpga_release,
    .read = fpga_read,
    .write = fpga_write,
    .unlocked_ioctl = fpga_ioctl,
};
```

然后,定义杂项设备(即主设备号为 10 的特殊字符型设备)结构体变量 fpga_misc,

```
static struct miscdevice fpga_misc = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = DEVICE_NAME,
    .fops = &fpga_fops,
};
```

其中主要包括 3 个域:分别是次设备号、设备名称与 file_operations 类型结构体变量的引用,

为增加程序的可读性和可移植性,这里将设备名定义为宏: #define DEVICE_NAME “fpga”。

3.2 设备驱动的初始化

设备驱动的初始化包括三部分: S3C6410 外部存储器硬件寄存器的初始化、将设备由物理空间映射到 Linux 内核虚拟空间及向内核注册杂项设备。

3.2.1 硬件寄存器初始化

由于 EP4CE40F23C8 芯片被当作外部存储器连接在 S3C6410 的存储器总线上,对硬件寄存器的初始化可以通过对 SROM_BW 总线宽度等待控制寄存器进行配置实现。Linux 系统必须先调用 ioremap() 函数将物理地址映射到虚拟地址空间,该函数原型为: void * ioremap(unsigned long phys_addr, unsigned long size)^[6];

其中, phys_addr 为要映射的外设 I/O 起始物理地址; size 表示要映射的空间大小,其返回值为对应的内核虚拟地址。这里调用如下代码实现地址映射: memcfg_addr_vir = ioremap(0x70000000, 0x20); 其中, 0x70000000 是 SROM_BW 总线宽度等待控制寄存器的物理地址, 返回值 memcfg_addr_vir 是它在 Linux 内核空间对应的虚拟地址。

将 SROM_BW 的 8~11 位(对应 BANK2)设置为 0101, 表示不使用 BU/LB, WAIT 信号使能, 并将 BANK2 的数据总线宽度设置为 16 位^[7]。

3.2.2 将设备由物理空间映射到内核虚拟空间

由于 ARM 处理器并没有为已知外设 I/O 内存资源的物理地址预定义虚拟地址, 因此, 驱动程序并不能直接通过物理地址访问 I/O 内存资源, 而必须将它们映射到内核虚拟空间。

FPGA 的 I/O 内存资源的物理地址由硬件决定。本架构选择 S3C6410 的 nCS2(即 BANK2), 起始地址为 0x20000000。这里将 FPGA 映射到 linux 内核的虚拟地址定义为 fpga_addr_vir, 然后调用 ioremap() 将 SROMC BANK2 的地址映射到 Linux 内核, 映射空间大小为 128M, 代码如下: fpga_addr_vir = ioremap(0x20000000, 0x07FFFFFF);

3.2.3 向内核注册杂项设备

在模块加载函数 module_init() 中通过调用 misc_register() 实现向 Linux 内核注册该杂项设备。

因此, 在使用 insmod 模块加载命令时, 即可自动地调用该函数, 完成该设备向内核的注册。

3.3 设备的读写

在设备读/写函数中, 通过系统内核函数 copy_to_user() 和 copy_from_user() 分别实现从内核空间到用户空间和从用户空间到内核空间的数据拷贝功能。

3.3.1 从 FPGA FIFO 中读取数据

在读函数 fpga_read(struct file * filp, char * user * buf, size_t size, loff_t * ppos) 中, 首先调用 ioread16() 函数将数据从 FPGA 中读出, 代码如下:

```
fpga_buf = ioread16((void *) (fpga_addr_vir + nowpos * 2));
```

其中 fpga_addr_vir 为 FPGA 的虚拟地址, fpga_buf 为 unsigned int 指针类型, 将其转换成 char 类型指针后, 调用 copy_to_user() 将从 FPGA 中读取的数据从内核空间拷贝到用户空间, 主要代码如下:

```
temp[0] = fpga_buf >> 8; // 高 8 位
temp[1] = fpga_buf & 0xff; // 低 8 位
if (copy_to_user(buf, temp, 2)) {
    return -EFAULT;
}
```

3.3.2 向 FPGA FIFO 中写数据

在 fpga_write(struct file * filp, const char __user * buf, size_t size, loff_t * ppos) 函数中, 首先, 调用 copy_from_user() 函数将数据从用户空间拷贝到内核空间中, 这里 char 类型指针变量 temp 指向内核空间; 然后, 将该缓冲区中的数据复制到 fpga_buf 缓冲区, 并调用 iowrite16() 将其写入到 FPGA 的 FIFO 中, 主要代码如下:

```
if (copy_from_user(temp, buf, 2)) {
    return -EFAULT;
}
```

```
fpga_buf = temp[0] << 8 | temp[1];
iowrite16(fpga_buf, (void *) (fpga_addr_vir + nowpos * 2));
```

3.3.3 读写进程互斥功能的实现

多个进程同时对 FPGA 进行读写操作时, 会造成 FPGA FIFO 中数据的不确定性。因此, 在驱动程序的 FPGA 读写函数中, 采用信号量实现读/写进程的互斥, 从而保证同一时间内只有一个

进程对 FPGA FIFO 进行读或写操作^[8]。

首先,在驱动程序中定义一个信号量 struct semaphore sem;然后,在设备驱动初始化函数中调用 sema_init(&sem, 1)将该信号量初始化为 1;在执行读/写功能前要先获取信号量,再调用函数 down_interruptible(&sem);最后,读/写操作完成,调用 up(&sem)释放信号量,告知 Linux 操作系统的其他进程使用 FPGA 的 FIFO。

3.4 驱动配置与移植

首先,将编写好的驱动程序复制到内核源码/home/linux-2.6.36/drivers/char/目录下,并修改 Kconfig 文件,添加 FPGA 驱动的编译配置选项:

```
config MINI6410_FPGA
tristate "FPGA Support for Mini6410"
depends on CPU_S3C6410
help
    this is fpga driver.
```

其次,进入图形化的内核配置界面,在 Device Drivers → Character devices 页面下选中 Kconfig 文件所添加的模块。FPGA 驱动配置如图 5 所示。

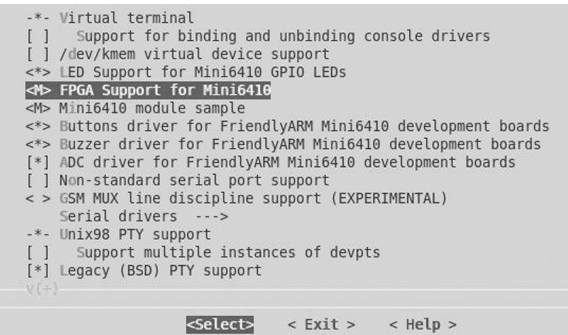


图 5 FPGA 驱动配置图
Fig. 5 Diagram of FPGA driver configuration

再次,在 Makefile 中添加如下语句,把内核配置选项和真正的 FPGA 驱动源代码联系起来: obj-\$(CONFIG_MINI6410_FPGA) += mini6410_fpga.o,并执行 # make modules 命令编译得到 mini6410_fpga.ko 动态加载模块。

最后,将编译好的动态模块复制到 ARM 板根文件系统/lib/modules/2.6.36 目录下,并执行 # insmod mini6410_fpga.ko 命令完成模块的动态加载,在/dev 目录下可以看到该设备文件的挂载点,如图 6 所示。

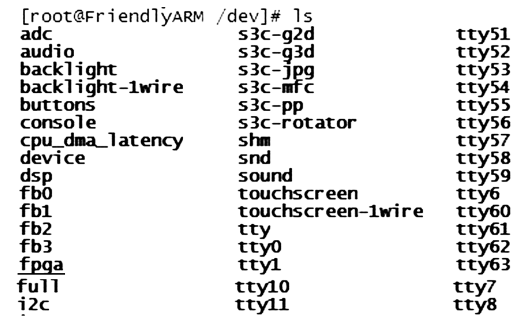


图 6 FPGA 设备挂载点
Fig. 6 Diagram of FPGA device node

3.5 驱动测试

执行 # modinfo 命令查看 kernel 模块信息,包括模块开发者、许可证及内核版本信息,如图 7 所示。

向内核加载 FPGA 设备驱动模块成功后,执

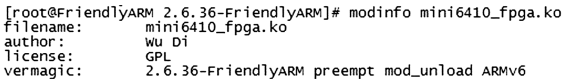
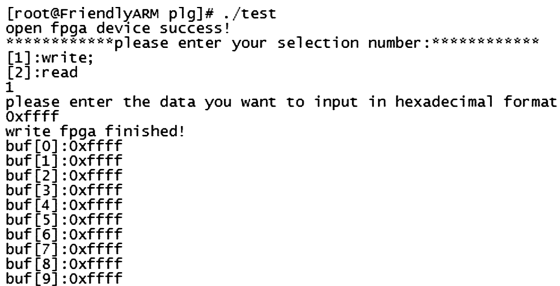
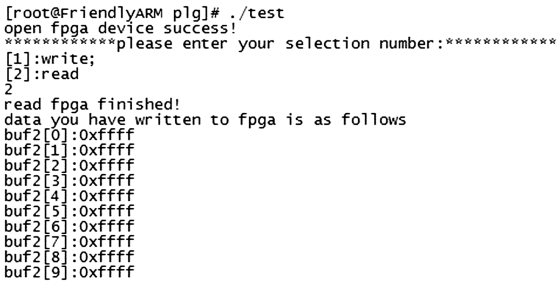


图 7 FPGA 设备驱动信息图
Fig. 7 Diagram of FPGA device driver information



(a)向 FPGA 中写入数据
(a)Test of writing data to FPGA



(b)从 FPGA 中读取数据
(b)Test of reading data from FPGA

图 8 设备驱动测试图
Fig. 8 Diagram of device driver testing

行设备驱动测试程序,图 8(a)实现了 ARM 处理器向 FPGA FIFO 中连续写入从键盘输入的数据,图 8(b)实现了 ARM 处理器从 FPGA FIFO 中读取写入数据的操作。结果表明,S3C6410 能够实现对 FPGA FIFO 的正确读写,保证了数据交互的正确性与稳定性。

4 Qt GUI 图形界面设计

系统将处理后的图像数据写入 FPFA 片内 FIFO,然后 ARM 处理器读取该图像数据,并基于 Qt GUI 设计了人机交互界面,实现影像数据在 LCD 液晶屏上的实时显示。

Qt GUI 图形界面的设计步骤如图 9 所示。

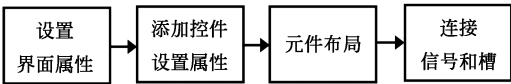


图 9 图形界面的设计步骤

Fig. 9 Design steps of the graphical interface

本设计的人机交互界面包括终端登录界面和影像显示界面两部分,主要使用了 Qt API 的 QLabel 类、QPixmap 类以及 QPushButton 类,并为终端添加了开机启动界面。程序设计完成后,将编译好的 Qt 程序下载到 ARM 板,并将执行该程序的 SHELL 命令写入开机启动脚本中,从而实现终端的开机自启动功能,登录界面如图 10 所示。



图 10 终端登录界面

Fig. 10 Terminal login interface

终端登录界面类被定义为 Log。在该界面的设计中,将图 10 中“Welcome Log in”对应的按钮 QPushButton * logbut 的 SIGNAL(clicked()) 信号与槽函数 SLOT(test())相关联,并在槽函数

test()中执行当前对象的隐藏函数 hide()和影像显示界面的显示函数 show(),即实现了从登陆界面到影像显示界面的切换,主要代码如下:

```
logbut = new QPushButton;//定义登陆按钮

/* 设置按钮属性,包括其尺寸、图标 */
QPixmap pix("/image/log.png");
logbut->setIcon(pix);
logbut->setIconSize(QSize(100,50));
logbut->setFixedSize(QSize(100,50));
connect(logbut, SIGNAL(clicked()), this, SLOT(test())); //连接信号和槽
...

/* 界面切换槽函数 test() */
voidLog::test(){
img.show();//影像显示界面显示
this->hide();//当前登录界面隐藏
}
```

影像显示界面类被定义为 Img。为实现系统实时获取测绘信息的后续查看功能,ARM 处理器将接收的影像保存为适合 QLabel 类加载的 .bmp 格式文件,将该文件重命名为当前系统时间,并保存在 SD Card 中。因而后续查看时不仅可以获得图像信息还可以获取记录该影像的时间信息。在该界面的设计中,ARM 处理器将从 FPGA 接收到的影像数据加载在 QLabel 上,并采用定时器事件不断刷新加载到 QLabel 上的影像数据,从而实现基于 Qt GUI 的影像显示。由于人眼视觉暂留特性,当连续的图像以大于 25 帧/s 的速率显示时,即可达到人眼对平滑流畅视觉效果的要求^[9]。影像显示界面的实现流程如图 11 所示。

该界面实现的具体步骤如下:

首先,在头文件中定义影像显示界面类 Img 成员变量,如下:

```
QPushButton * recvbutton;//接收按钮
QPushButton * stopbutton;//停止显示按钮

QLabel * label;//用于加载实时影像
QPixmap pm;
```

然后,在影像显示子界面类 Img 实现源文件中将接收按钮 recvbutton 的 SIGNAL(clicked()) 信号和槽函数 SLOT(mytimer())相关联,代码

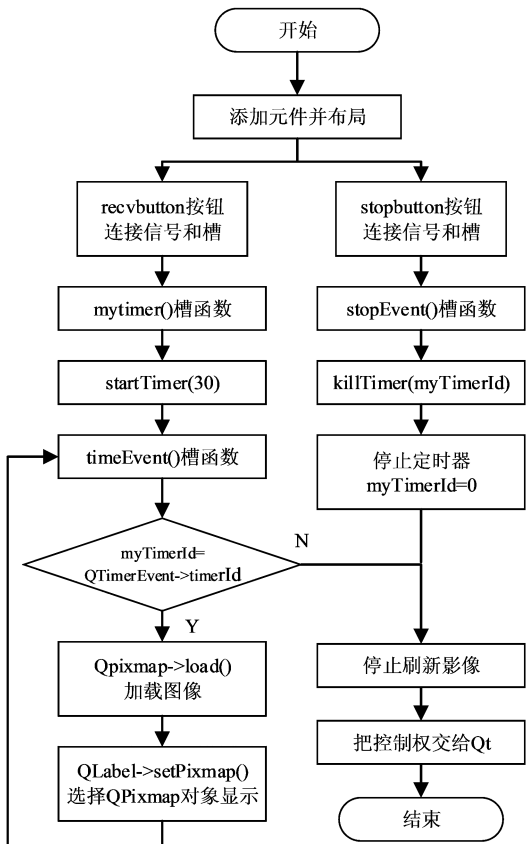


图 11 影像显示界面的实现流程图

Fig. 11 Flowchart of image display interface

如下:

```
connect(recvbutton, SIGNAL(clicked()),
this,SLOT(mytimer()));
```

在槽函数 mytimer()中调用 startTimer()函数来开启定时器,并设定定时器定时间隔为 30 ms。startTimer()函数将返回一个 int 类型的定时器 ID,即全局变量 int myTimerId,代码如下:

```
voidImg::mytimer(){
myTimerId= startTimer(30);
}
```

当定时时间达到 30 ms 时,全局变量 myTimerId 和 QTimerEvent * event->timerId()相等,即触发定时器事件的发生。在 Img 类的 timerEvent()槽函数中,首先调用 QPixmap->load()加载图像,然后调用 QLabel->setPixmap()选择一个 QPixmap 类对象显示,实现代码如下:

```
void Img:: timerEvent (QTimerEvent * e-
vent)
{
```

```
if (event->timerId() == myTimerId)
{
    pm. load("/image/recv. bmp");
    label->setPixmap(pm);
}
else
    QWidget::timerEvent(event);
}
```

在 stopbutton 停止显示按钮的槽函数 stopEvent()中调用 killTimer()函数停止定时器,并将全局变量 myTimerId 设置为 0,因此定时器事件将不再被触发,此后停止刷新加载在 label 上的影像,将控制权交给 Qt。影像显示界面如图 12 所示。

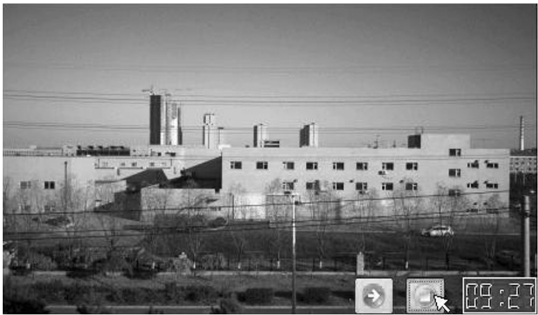


图 12 影像显示界面

Fig. 12 Image display interface

5 结 论

为实现航空测绘信息获取的数字化、实时化,本文结合嵌入式处理器 S3C6410 与 FPGA 的各自优势,提出了一种基于 ARM+FPGA 双核硬件架构的高速影像交互与显示系统的解决方案^[10]。该系统采用并行总线方式实现 ARM 与 FPGA 之间的高速数据通信,利用 Linux 2. 6. 36 内核完成了对 FPGA 设备的驱动设计,实现了高速影像数据交互;并在 ARM-Linux 平台上设计了基于 Qt 的人机交互界面,用以实现高速影像的实时显示。测试结果表明,该系统实现了 VGA (640×480)格式的高速影像交互,帧率最大可达 26 帧/s,传输带宽最大为 182 Mbps。该系统不仅体积小、功耗低、成本低,而且结构紧凑、专用性强、稳定性好、性能佳,能够满足测绘信息获取的实时性要求。

参 考 文 献:

- [1] 徐正平, 许永森. S3C2440A 在步进电机控制机器人机交互中的应用[J]. 液晶与显示, 2015, 30(1): 70-76.
Xu Z P, Xu Y S. Application of S3C2440A in human-computer interaction of stepping motor controller [J]. *Chinese Journal of Liquid Crystals and Display*, 2015, 30(1): 70-76. (in Chinese)
- [2] 宋亚军, 许廷发, 倪国强, 等. 基于 Virtex-4 FPGA 的低功耗图像融合系统[J]. 光学精密工程, 2007, 15(6): 935-940.
Song Y J, Xu T F, Ni G Q, *et al.* Low power image fusion system based on Virtex-4 FPGA [J]. *Opt. Precision Eng.*, 2007, 15(6): 935-940. (in Chinese)
- [3] 宋宝华. Linux 设备驱动开发详解[M]. 北京: 人民邮电出版社, 2008.
Song B H. *Details for Embedded Linux Device Drivers Development* [M]. Beijing: The People's Posts and Telecommunications Press, 2008. (in Chinese)
- [4] 王亚庭. 基于 ARM 与 FPGA 的高速数据采集技术研究[D]. 北京: 北京交通大学, 2007.
Wang Y T. *Research on High speed data acquisition technology based on ARM&FPGA* [D]. Beijing: Beijing Jiaotong University, 2007. (in Chinese)
- [5] 赵同样. 嵌入式 Linux 系统驱动研究与开发[D]. 哈尔滨: 中国地震局工程力学研究所, 2012.
Zhao T X. *Study and development of embedded Linux system driver* [D]. Harbin: Institute of Engineering Mechanics, China Earthquake Administration. 2012. (in Chinese)
- [6] Mitchell M, Oldham J, Samuel A. *Advanced Linux Programming* [M]. America: New Riders Publishing, 2001.
- [7] Samsung Electronics. User's Manual S3C6410X RISC Microprocessor [R]. Korea: Samsung Electronics, Inc., 2008.
- [8] 汪益立, 胡旭东, 黄斌. Linux 内核中双口 RAM 驱动开发及在大圆机控制系统中的应用[J]. 现代纺织技术, 2014(4): 13-17.
Wang Y L, Hu X D, Huang B. Development of dual-port RAM driver in Linux kernel and application in control system of circular machine [J]. *Advanced Textile Technology*, 2014(4): 13-17. (in Chinese)
- [9] 华清远见嵌入式培训中心. 嵌入式 Linux 应用程序开发标准教程[M]. 北京: 人民邮电出版社, 2009.
Huaqing Vision Embedded Training Center. *Details for Embedded Linux Application Program Development* [M]. Beijing: The People's Posts and Telecommunications Press, 2009. (in Chinese)
- [10] 卢振华, 郭永飞, 李云飞, 等. 利用 CCD 拼接实现推扫式遥感相机的自动调焦[J]. 光学精密工程, 2012, 20(7): 1559-1565.
Lu Z H, Guo Y F, Li Y F, *et al.* Realization of auto-focus on APRC using CCD stitching [J]. *Opt. Precision Eng.*, 2012, 20(7): 1559-1565. (in Chinese)

作者简介: 吴迪(1988—), 女, 吉林长春人, 硕士, 研究实习员, 主要从事嵌入式系统设计及信息处理方面的研究。

E-mail: wudihope@163.com

李丙玉(1980—), 男, 河北沧州人, 硕士, 助理研究员, 主要从事空间图像传感器成像控制与信息处理技术研究。E-mail: liby0125@126.com

王晓东(1970—), 男, 吉林长春人, 博士, 研究员, 博士生导师, 主要从事光电探测与检测及信息处理方面的研究。E-mail: wangxd@ciomp.ac.cn