磁共振成像仪通信及脉冲生成模块的设计*

郭桦炜 郑振耀 陈 忠 姚凯文 郑泽寰
(厦门大学电子科学系 福建省等离子体与磁共振研究重点实验室 厦门 361005)

摘 要：针对小型化核磁共振成像仪主控板的功能需要提出新的设计方式，采用 FPGA 开发板作为主控板，以 Nios II 嵌入式软核作为协处理器，通过在其上移植 μC/OS-Ⅱ操作系统及 lwIP 协议栈，实现与计算机的网络通信；使用 Verilog 硬件描述语言编写脉冲序列生成模块，实现对指令的解析以及执行，形成用户要求的脉冲序列。最终测试结果表明，主控板的通信速率以及通过示波器采集得到的脉冲序列均满足成像仪系统要求。而且这种软硬结合的设计方式与以往全软件的设计相比，使成像仪精度更高，性能更加稳定。

关键词：核磁共振；FPGA；网络通信
中图分类号：TP302 **文献标识码**：A **国家标准学科分类代码**：510.8040

Design of MRI communications and pulse generation module

Guo Huawei Zheng Zhenyao Chen Zhong Yao Kaiwen Zheng Zehuan
(Department of Electronic Science, Fujian Key Laboratory of Plasma and Magnetic Resonance, Xiamen University, Xiamen 361005, China)

Abstract: To satisfy the require of the MRI main control board, this paper puts forwards a new design which uses FPGA development board as main control board and uses Nios II as a co-processor. We transplant the μC/OS-II operating system and protocol stack lwIP to achieve the network communication with computers. Then we use the hardware language Verilog to write pulse sequence generation module to achieve the analysis and execution of the command. Eventually it produces user's pulse sequence. Final test results show that the communication rate of the main control board and pulse sequence which is obtained by oscilloscope can satisfy the requirements of the MRI system. Compared with the past of the whole software design, this design make the MRI system higher precision and more stable.

Keywords: nuclear magnetic resonance; FPGA; network communication

1 引 言

核磁共振成像系统主要由计算机、磁体、主控板、发射模块、梯度模块、接收模块和线圈等部分组成[1]，主控板不仅需要与计算机进行通信，也负责生成用户所要求的脉冲序列以及完成对射频模块、梯度模块、接收模块的控制。其中通信及脉冲生成模块设计的好坏严重关系到成像仪的整体性能与最终的成像效果[2]。在以往的设计中，通常是将主控板大多的功能例如与计算机的通信，指令的解释以及脉冲序列的生成都通过微处理器(CPU)来执行。这种设计使得 CPU 负担过重，随着任务的加剧最终导致整体性能的下降。在本文的主控板设计中，CPU 只负责与计算机的网络通信与数据存储，其余功能则是由 FPGA 采用 Verilog 语言编写的硬件模块来执行。两者的分离大大提升了成像系统的性能以及稳定性。

2 主控板整体设计方案

在本文的核磁共振成像系统设计中，选用 Altera 公司的 Stratix III 系列的开发板作为主控板。该开发板所使用的 EP3SL150F1152 芯片是一款低功耗、高性能、大容量的高端 FPGA 芯片，逻辑资源丰富，而且还带有嵌入式 Nios II 软核处理器，完全满足设计的需要。如图 1 所示为主控板的整体结构。

从图 1 可以看出，主控板的设计可以分为两个部分。

第 1 部分是嵌入式软件设计，以开发板自带的 Nios II 作为协处理器，采用 Quartus II 中的 SOPC builder 软件在 FPGA 上构建了嵌入式开发平台[3]。该平台的硬件结构如图 2 所示。之后在其上移植 μC/OS-Ⅱ的操作系统以及 lwIP TCP/IP 协议栈，实现与计算机软件的网络通讯及数据交换，将计算机通过网口发下来的命令数据存储在双口

收稿日期：2016-03
* 基金项目：国家自然科学基金(11175149)资助项目

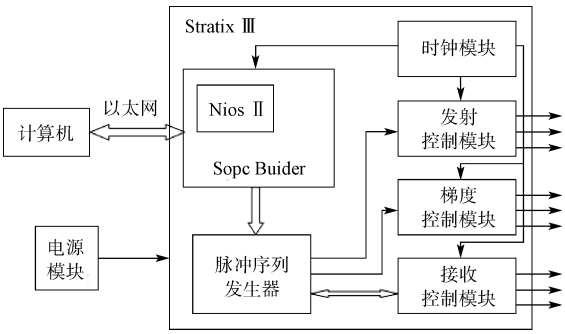

图 1　主控板整体结构

RAM 中，并将采集到的 FID 数据通过网口回传给计算机进行图像处理。采用网线与计算机进行通信的方式与其他的通信方式相比能提供更高速，更定的传输速度以及可进行远距离传输。

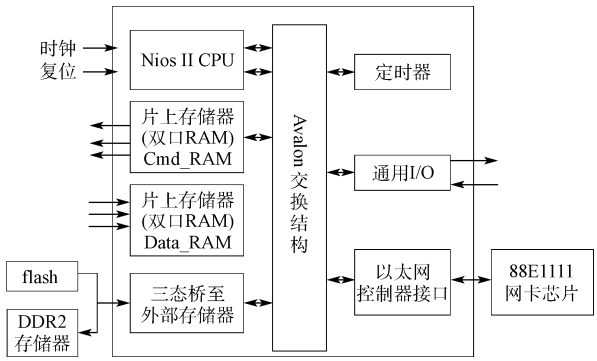

图 2　SOPC builder 嵌入式开发平台结构

第 2 部分是硬件设计。主要是采用 Verilog 硬件语言编写脉冲序列生成模块、发射板控制模块、梯度板控制模块等电路板控制模块。通过脉冲序列生成模块对存储在 RAM 中的数据进行解析，实现对射频模块、梯度模块等模块的信号控制，最终生成用户所要求的脉冲序列[4]。其中发射板控制模块等电路板控制模块需根据具体电路板的性能以及时序要求进行编写，本文不对此进行说明。

这种设计结构也将主控板的功能明确地划分为两个部分。嵌入式设计实现与计算机网络通信的功能。而硬件设计则实现了对接收到的命令进行解析并生成用户要求的脉冲序列。这种主控板设计结构的优点在于：

1)缓解 CPU 运行压力。CPU 只负责与计算机的网络通信。

2)采用硬件的方式执行命令可以提升系统的性能以及稳定性。硬件的设计支持并行操作，且每条命令的执行时间是一定的。

3)有利于成像仪系统多通道的的扩展。由于各通道间是相互独立的，只需对相应的硬件部分进行拷贝即可实现多通道的扩展。

3　通信模块设计

3.1　μC/OS-Ⅱ移植

μC/OS-Ⅱ是一款开源的多任务实时操作系统，具有可移植性、可固化、可裁剪、占先式、中断管理等特点。完全满足成像仪系统的需要，而且其源代码是公开的，方便移植。因此，选用 μC/OS-Ⅱ作为嵌入式设计的操作系统。

μC/OS-Ⅱ的文件系统结构如图 3 所示。μC/OS-Ⅱ的代码源文件可以分为上层应用程序相关的配置文件、与处理器无关的核心功能实现文件和处理器有关的移植文件[5]。

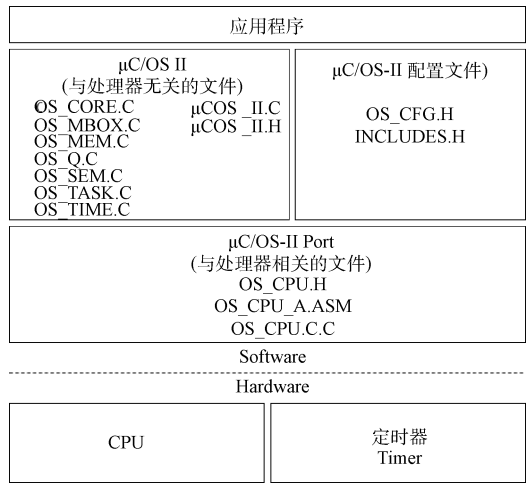

图 3　μC/OS-Ⅱ文件系统结构

对于 μC/OS-Ⅱ的系统移植主要是修改与处理器相关的文件部分，即主要对 OS_CPU. H、OS_CPU. C、OS_CPU. S 进行修改[6]。其中：

1)在 OS_CPU. H 中重定义了与编译器相关的数据类型、宏定义以及堆栈数据类型。

```
/* This is the definition for Nios II.    */
typedef unsigned char   BOOLEAN;
typedef unsigned char   INT8U;
……
typedef unsigned int    OS_STK;
#define   OS_STK_GROWTH   1       /* Stack grows from HIGH to LOW memory */
```

在这个文件中还宏定义了两个在移植过程中非常重要的函数。

```
#define   OS_CPU_SR alt_irq_context
#define   OS_ENTER_CRITICAL()   cpu_sr = alt_irq_disable_all ()
#define   OS_EXIT_CRITICAL()   alt_irq_enable_all (cpu_sr)
```

OS_ENTER_CRITICAL()关中断和 OS_EXIT_CRITICAL()开中断是操作系统进行原子操作时不可缺少

的环节。这里直接调用 Nios II 自带的开关中断函数。

2)在 OS_CPU.C 中实现堆栈初始化函数以及一系列在应用程序中将要用到的 Hook 函数。只需修改 OSTaskStkInit()初始化堆栈函数,其他的 Hook 函数在本系统设计中没有用到,所以无需修改。

3)在 OS_CPU.S 中需使用汇编语言实现任务启动函数、任务切换函数。这部分的修改可以说是整个操作系统移植的核心部分也是最难的部分,因为需要对 Nios II CPU 进行直接的操作。主要实现如下几个函数:

OSStartHighRdy():运行优先级最高的就绪任务。获得运行优先级最高的就绪任务的指针。

OSCtxSw():任务级的任务切换函数。该函数可以获得任务指针并保护现场。

OSIntCtxSw():中断级的任务切换函数。可以获得任务指针和保护现场。

在对 CPU 的操作完成之后,距离整个移植的完成还差最关键的一步,就是系统时钟的设置。需要开启定时器中断设置并绑定时钟中断函数。

alt_alarm_start(&alarm,1, OSTickISR,NULL);

其中 OSTickISR()是处理时钟中断函数,内部只需调用系统自带函数 OSTimeTick。

3.2 lwIP 协议栈移植

lwIP(light weight IP)是瑞士计算机科学院的 Adam Dunkels 等开发的一套开放式 TCP/IP 协议栈源代码[7]。在本文的设计中,将 lwIP 移植到 μC/OS-Ⅱ操作系统上。实现主控板与计算机软件的 TCP/IP 通信。

lwIP 协议栈在设计的时候就考虑到了移植问题,已经把所有的与硬件、操作系统、编译器等有关的部分全部独立出来,放在.\src\arch 目录下,因此,lwIP 在 μC/OS-Ⅱ上的移植主要是修改这个目录下的文件。主要对以下几个接口进行移植。

3.2.1 与 CPU 或编译器相关的接口

在.\src\arch\include\arch 目录下,cc.h、perf.h 中有一些与 CPU 或编译器相关的定义,如数据类型、存储模式的选择、字的高低位顺序等。这些都应与之前移植 μC/OS-Ⅱ时定义的参数保持一致。

3.2.2 与操作系统 μC/OS-Ⅱ相关的接口

lwIP 将与操作系统相关函数都放在了 sys_arch.h 及 sys_arch.c 文件中。需要实现的是 sys_sem_t 信号量一系列函数、sys_mbox_t 消息一系列函数、定时器函数以及 lwIP 线程创建函数。由于 μC/OS-Ⅱ也存在信号量、消息机制,所以在对信号量函数以及部分消息函数的实现皆可通过对 μC/OS-Ⅱ的类似函数进行重新封装得到[8]。但有如下几个函数需要另外编写:

1)由于 μC/OS-Ⅱ中没有对消息队列中的消息进行管理,所以需要另外定义一个消息队列结构。

```
typedef struct{
    OS_EVENT    * pQ;
    void  * pvQEntries[MAX_QUEUE_ENTRIES];}
sys_mbox_t;
```

对于消息队列本身的管理利用 μC/OS-Ⅱ自己的 QSQ 操作完成,使用 μC/OS-Ⅱ的内存管理模块实现对消息的创建、使用、删除和回收等操作即可实现了 lwIP 消息队列功能。

2)lwIP 中每个与外界网线连接的线程都对应一个 sys_timeout 结构队列。sys_timeout 和 sys_timeouts 结构体已在 sys.h 中定义,且对该队列的数据操作也有 lwIP 负责,只需实现找到当前线程使用的 sys_timeouts 结构指针的函数。

```
struct sys_timeouts  * sys_arch_timeouts(void){
u8_t currPrio;
    currPrio = (OSPrioCur-LWIP_START_PRIO);
    return &timeoutlist[currPrio];   }
```

3)μC/OS-Ⅱ提供了创建新任务函数 OSTaskCreate,因此只需将该函数封装一下即可实现 lwIP 的线程创建函数 sys_thread_new。但 lwIP 中的 thread 并没有 μC/OS-Ⅱ优先级的概念,所以在实现时需事先为 lwIP 创建的线程分配好优先级。

```
sys_thread_t sys_thread_new(void ( *  thread)(void
* arg), void  * arg, u8_t lwIP_prio){
OSTaskCreate(thread,arg,(void * )&sys_stack[lwIP
_prio][LWIP_STACK_SIZE-1],LWIP_START_PRIO+
lwIP_prio);
  return lwIP_prio;);
```

3.2.3 网络设备驱动程序

在本文的设计中采用的网络芯片是 Marvell 公司推出的 88E1111 以太网收发芯片。在为该芯片编写驱动程序时需要参考 /src/netif/ethernetif.c 文件,因为 lwIP 设计者为方便移植提供了网络驱动的模板。具体的底层驱动结构如图 4 所示。

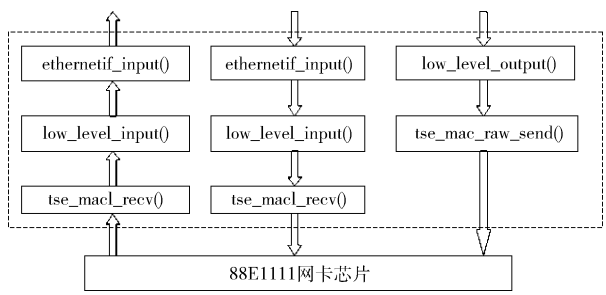

图 4 lwIP 底层驱动结构

其中,ethernetif_init()、ethernetif_input()作为对外接口实现网卡芯片的初始化以及以太网数据包的读取。low_level_output()函数通常在 ethernetif_init()执行时与相应的 netif 结构体绑定实现以太网数据包的发送。而 tse_mac

_init()、tse_max_recv()、tse_mac_raw_send()则是实现网卡初始化、数据包收发功能的最底层函数，需要与 88E1111 网卡芯片寄存器进行直接的操作[9]。

3.3　网络通信测试

完成网络芯片的驱动程序开发后，为了检测移植是否成功，可以通过 Ping 命令来测试 ICMP 协议是否正常工作。通过 iperf 软件来测试 ARP、TCP、IP 协议是否正常工作。iperf 是一个网络性能测试工具，可以测试 TCP 和 UDP 的带宽质量并向用户报告带宽、延迟抖动和数据包丢失情况[10]。测试所得到的结果如图 5、6 所示。

```
C:\Documents and Settings\Administrator>ping 192.168.16.24

Pinging 192.168.16.24 with 32 bytes of data:

Reply from 192.168.16.24: bytes=32 time<1ms TTL=64
Reply from 192.168.16.24: bytes=32 time<1ms TTL=64
Reply from 192.168.16.24: bytes=32 time<1ms TTL=64
Reply from 192.168.16.24: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.16.24:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

图 5　Ping 测试结果

```
C:\Documents and Settings\Administrator>iperf -c 192.168.16.24 -p80 -i 5 -t 50
------------------------------------------------------------
Client connecting to 192.168.16.24, TCP port 80
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1872] local 192.168.16.100 port 49888 connected with 192.168.16.24 port 80
[ ID] Interval       Transfer     Bandwidth
[1872]  0.0- 5.0 sec  7.57 MBytes  12.7 Mbits/sec
[1872]  5.0-10.0 sec  7.51 MBytes  12.6 Mbits/sec
[1872] 10.0-15.0 sec  7.52 MBytes  12.6 Mbits/sec
[1872] 15.0-20.0 sec  7.55 MBytes  12.7 Mbits/sec
[1872] 20.0-25.0 sec  7.54 MBytes  12.6 Mbits/sec
[1872] 25.0-30.0 sec  7.51 MBytes  12.6 Mbits/sec
[1872] 30.0-35.0 sec  7.52 MBytes  12.6 Mbits/sec
[1872] 35.0-40.0 sec  7.53 MBytes  12.6 Mbits/sec
[1872] 40.0-45.0 sec  7.52 MBytes  12.6 Mbits/sec
[1872] 45.0-50.0 sec  7.53 MBytes  12.6 Mbits/sec
[1872]  0.0-50.0 sec  75.3 MBytes  12.6 Mbits/sec
```

图 6　iperf 测试结果(50 MHz 工作频率)

通过以上测试，说明 lwIP 协议可在 μC/OS-Ⅱ上正常工作。由于主控板的工作频率只有 50 MHz，无法充分发挥网卡芯片的性能。但通过 iperf 的测试结果，可以看到数据传送速度达到十多兆每秒，满足了成像仪系统与计算机的通信要求。而且经过测试，随着工作频率的增大，网络传输速度也随之增大。

4　脉冲序列生成模块设计

如图 7 所示为主控板整体功能实现流程图。其中软件部分即通信模块已经实现了与计算机的网络通信与命令的存储。主控板还需对计算机端发下来的数据进行解析并产生相应的脉冲序列。在本文的设计中，这之后的功能实现都采用硬件方式实现，即利用 FPGA 可编程逻辑的概念，使用 Verilog 硬件描述语言编写脉冲序列生成模块[11]。

从图 7 硬件部分可以看出，脉冲序列生成模块需完成以下处理：

1) 各硬件模块的初始化。主要是对发射板控制模块、

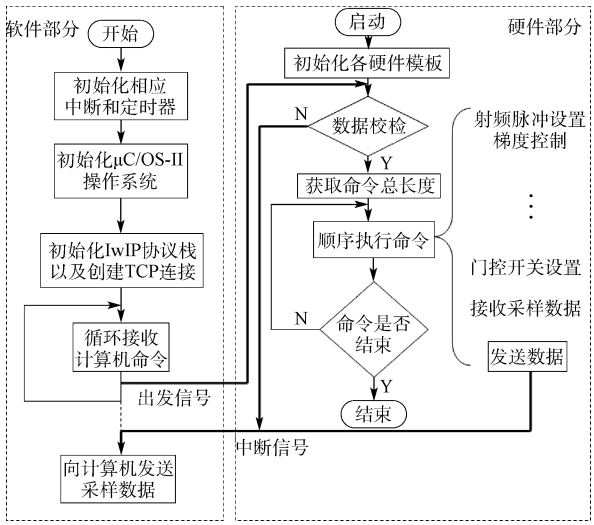

图 7　主控板整体功能流程

梯度板控制模块、接收板控制模块等模块的初始化操作。使整个成像仪系统处于初始化状态。这通常是在成像仪刚开机时或是复位键被按下时执行的。

2) 数据的读写操作。当软件部分接收到一整套的命令数据时发出信号通知脉冲序列生成模块开始从 RAM 中读取数据处理。由于在嵌入式平台搭建时采用了双口 RAM 的设计，所以硬件部分和软件部分可以对同一 RAM 进行读写操作，这省去了数据的复制时间。

3) 对接收到的数据进行校验。在这里需要对数据的头几个字节进行标识验证。保证接收到的数据是可被执行的命令。

4)获取整个命令的总长度。这为后面命令的循环执行确定了停止条件。

5)命令的执行。主要是根据命令及参数对发射板控制模块、梯度控制模块等模块进行配置。从而实现用户要求的脉冲序列。

最后生成的脉冲序列生成模块的原理图如图 8 所示。其中 cmd_gen 为主模块，recv 为采样数据接收模块，其他模块则是根据发射板控制模块等控制模块所需配置参数的不同而生成的，方便命令的执行。图 9 所示为用 modelsim 仿真软件对硬件的仿真结果，整体时序以及幅值都达到了预期的要求。图 10 所示为最终示波器采集的图像，与仿真结果相一致，满足成像仪脉冲序列要求。而且经过测试，脉冲宽度的精度可达到 μs 级别，与全软件的设计方式相比，极大地提升了成像仪的性能。

5　结　　论

采用了新的核磁共振成像仪主控板的设计方式。以 Nios II 作为协处理器以软件的方式实现与计算机的通信，再使用 Verilog 硬件描述语言以硬件的方式实现了脉冲序列的生成。经最终仿真与测试结果表明，这种设计相比于

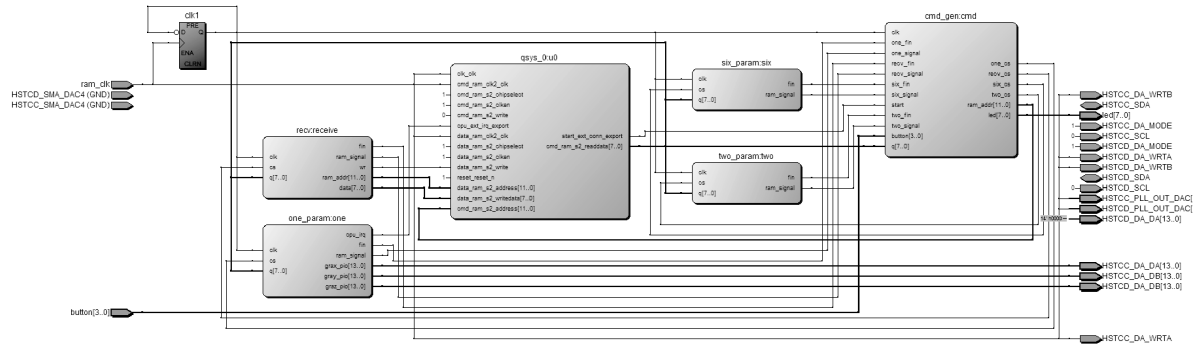

图 8 脉冲序列生成模块原理

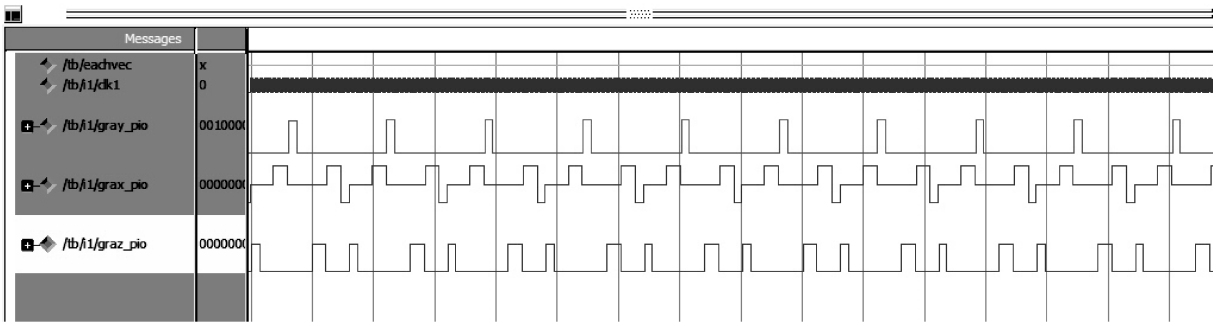

图 9 硬件设计仿真结果

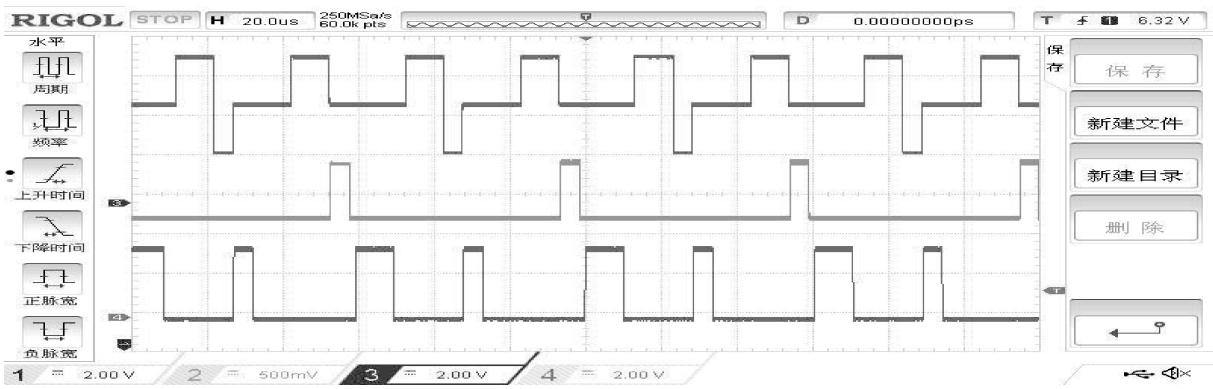

图 10 示波器输出结果

采用全软件的设计方式使得成像仪的性能更加稳定、精度更高。

参考文献

[1] 甘泉,曹国平,王俊,等. 脉冲核磁共振成像实验仪原理及其应用[J]. 医疗卫生装备,2010(8):111-114.

[2] 肖颖,郑振耀,蔡淑慧,等. 基于 FPGA 小型化谱仪控制台的设计[J]. 波谱学杂志,2012,29(3):346-353.

[3] 马俊,尉广军. 基于 SOPC 的以太网远程数据采集系统设计[J]. 电子设计工程,2012,20(4):69-72.

[4] 林君,王琳,杨宇,等. 核磁共振接收线圈谐振优化分析与标定[J]. 仪器仪表学报,2015,36(1):17-25.

[5] 郝玉胜,逯玉兰. 基于 UC/OS-II 的嵌入式网络通信模块[J]. 计算机与现代化,2014(2):142-145.

[6] 覃琴,宋海军,李长俊. 基于 μC/OS-Ⅱ和 NicheStack 的嵌入式以太网接口的设计[J]. 国外电子测量技术,2013,32(5):57-59.

[7] 彭求实,郑振耀,陈忠,等. 基于 STM32 的波谱仪接受通道的设计[J]. 电子测量术,2014,37(7):76-79.

[8] 王永伟,刘岩俊. 嵌入式网络控制系统设计与实现[J]. 国外电子测量技术,2014,33(9):50-53.

(下转第 129 页)