USB2.0 通信例程

黑金动力社区 2018-02-02

1 简介

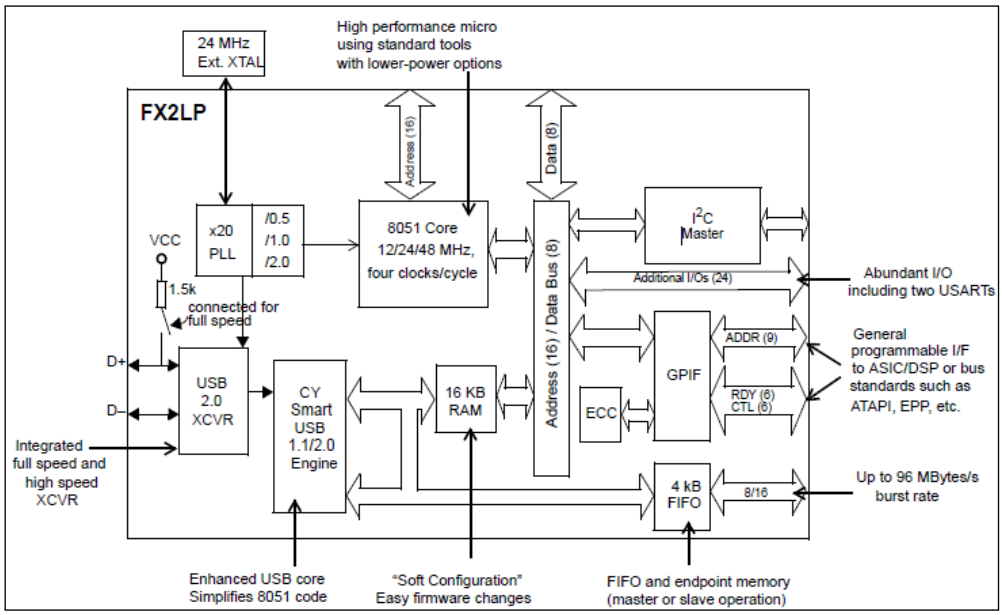
本教程介绍如何使用 Cypress 公司 EZ-USB FX2LP 系列的 CY7C68013A 驱动芯片进行 USB 数据传输，文中从底层 FPGA、驱动层固件（CY7C68013A 固件）及上位机测试软件设计方面阐述了 USB 开发的整个流程。

2 实验原理

2.1 CY7C68013A 原理介绍

CY7C68013A 是一款含高速 USB2.0 通信协议及兼容的 8051 控制 CYPRESS 公司的 USB2.0 控制芯片。在此芯片中含 16KB 的内部 RAM，不需外加存储器就可以直接运行固件程序。CY7C68013A 的内部 8051 处理器的工作频率可以为 48Mhz, 24Mhz, 12Mhz, AX7102 开发板中使用了 24Mhz 晶体作为时钟输入。

CY7C68013A 内部集成四个 FIFO，大小共为 4K 字节,用户可以通过设置让其工作在 Master 或 Slave 模式。CY7C68013A 含有一个 I2C 接口用于连接外部的 EEPROM， 用户先把写好的固件程序.iic 文件下载到 EEPROM。CY7C68013A 上电启动时会自动读取 EEPROM 的程序到内部的 RAM 开始运行。CY7C68013A 的内部结构图如下：

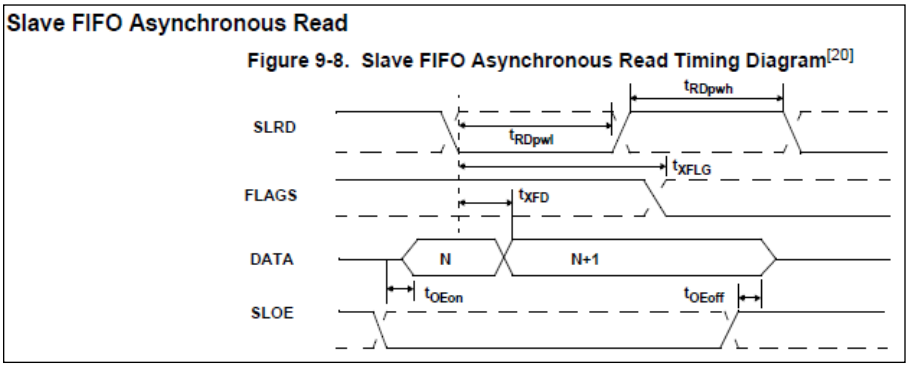


如果要了解更详细的信息请参考 CY7C68013A 的 Datasheet.

FPGA 和 USB 芯片 CY7C68013A 的数据通信是通过异步读写 CY7C68013A 的四个端口（EP2，EP4，EP6，EP8）的 FIFO 的数据来实现的，其中 FPGA 为主设备，CY7C68013 的端口为从设备。为了更好的理解下面的内容，我们这边有必要先来了解一下 CY7C68013A 的端口 FIFO 的异步读写时序。

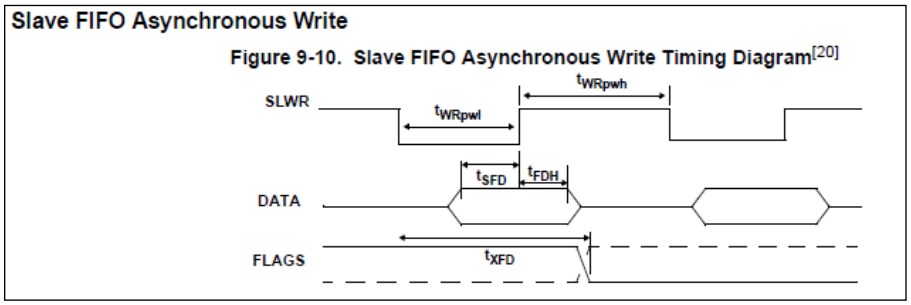
CY7C68013A FIFO 异步读：

外部系统控制 FIFOADDR0 和 FIFOADDR1 地址线的电平选择端口，拉低 SLOE 信号后，选择的端口会把 FIFO 的数据输出到数据总线。外部系统拉低 SLRD 读取数据。



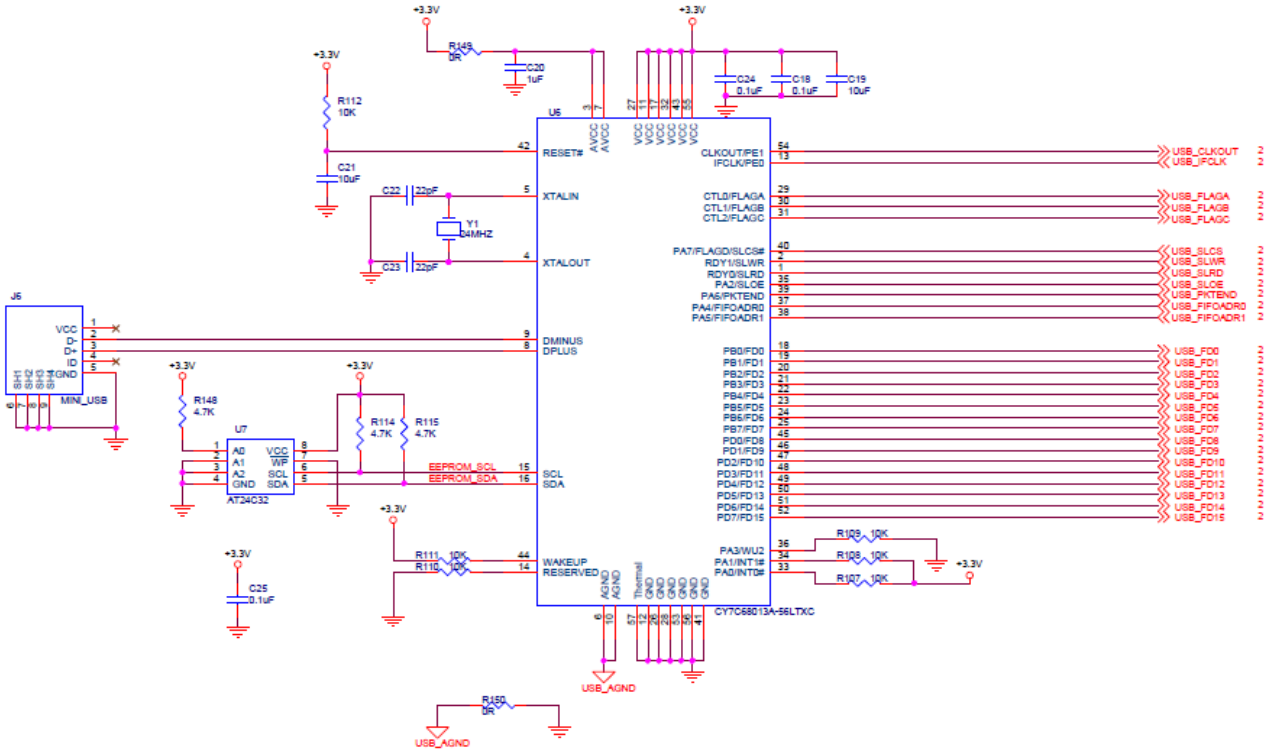
CY7C68013A FIFO 异步写：

外部系统控制 FIFOADDR0 和 FIFOADDR1 地址线的电平选择端口，再输出要写的数据到数据总线，拉高 SLWR 信号，SLWR 的上升沿把数据存入 FIFO 中。



2.2 硬件原理图

如下为 AX7102 开发板的 USB 部分原理图，USB 驱动芯片采用 CY7C68013A，驱动芯片与 AT24C32 存储芯片相连，将 USB 的配置信息存储于芯片中。以下为开发板上的 CY7C68013A 的接口电路：



CY7C68013A 和 FPGA 连接的管脚情况如下：

USB_CLKOUT-------- PIN:U18 USB _FD2---------- PIN:W19

USB _IFCLK---------- PIN:Y21 USB _FD3---------- PIN:Y19

USB_FLAGA---------- PIN:R18 USB _FD4---------- PIN:Y18

USB_FLAGB ---------- PIN:R14 USB _FD5---------- PIN:V22

USB_FLAGC ---------- PIN:P14 USB _FD6---------- PIN:U22

USB_SLCS ---------- PIN:P16 USB _FD7---------- PIN:T18

USB_SLWR ---------- PIN:R19 USB _FD8---------- PIN:R17

USB_SLRD ---------- PIN:P19 USB _FD9---------- PIN:R16

USB_SLOE ---------- PIN:N13 USB _FD10---------- PIN:P15

USB _ PKTEND --------- PIN:P20 USB _FD11---------- PIN:N17

USB _ FIFOADR0------- PIN:N14 USB _FD12---------- PIN:P17

USB _ FIFOADR1------- PIN:N15 USB _FD13---------- PIN:U16

USB _ FD0---------- PIN:Y22 USB _FD14---------- PIN:T16

USB _ FD1---------- PIN:W20 USB _FD15---------- PIN:U17

因为 CY7C68013A 工作在 Slave FIFO 模式，所以时钟信号 USB_CLKOUT 和 USB_IFCLK 在本实验中没有使用。

在开发板中，我们已经为用户配置了 CY7C68013 固件程序，其中设置 EP2，EP4 的 FIFO 为数据输出，EP6，EP8 的 FIFO 为数据输入，每个端口含有两个 512 字节的 FIFO，如果一个 512 字节 FIFO 被读空或写满，硬件会自动切换到另一个 512 字节 FIFO。另外 FLAGA~C 引脚的功能也是在固件程序中配置，其中控制信号和状态信号都是低电平有效。如果用户需要改变 CY7C68013A 的工作模式，端口或信号的配置，可以修改 CY7C68013A 的固件来实现。

3 程序设计

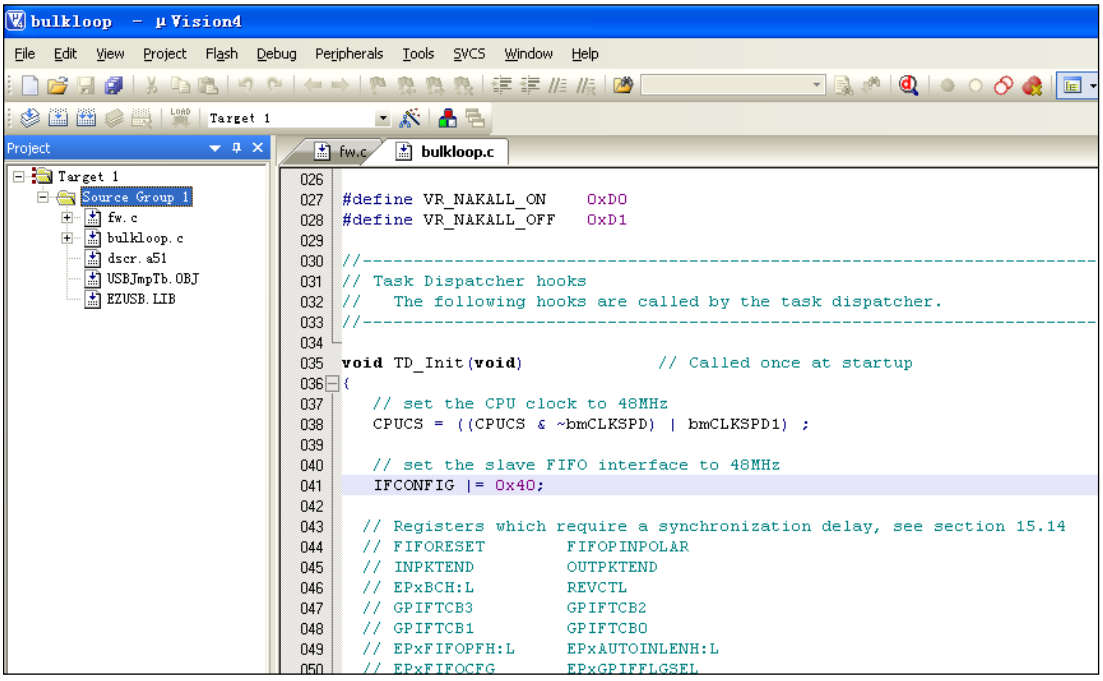
USB通信软件的开发包含两部分，一部分为CY7C6803A的固件设计，另一部分为在FPGA的verilog测试程序。因为我们已经在开发板上为大家在EEPROM中配置了CY7C68013A的固件程序，关于CY7C68013A的固件的设计和开发请参看“\05_软件工具及驱动\USB2.0驱动\USB固件程序”目录下的"黑金开发板CY7C68013A固件程序说明.pdf"文档,我们在这里只作简单的介绍。

1. CY7C68013A 的驱动安装

在使用 USB2.0 数据通信之前必须安装 CY7C68013A 的驱动，关于驱动的如何安装请参考\05_软件工具及驱动\USB2.0 驱动目录下的"WIN7 Cypress USB 驱动安装说明”一文。

2. CY7C68013A 固件设计

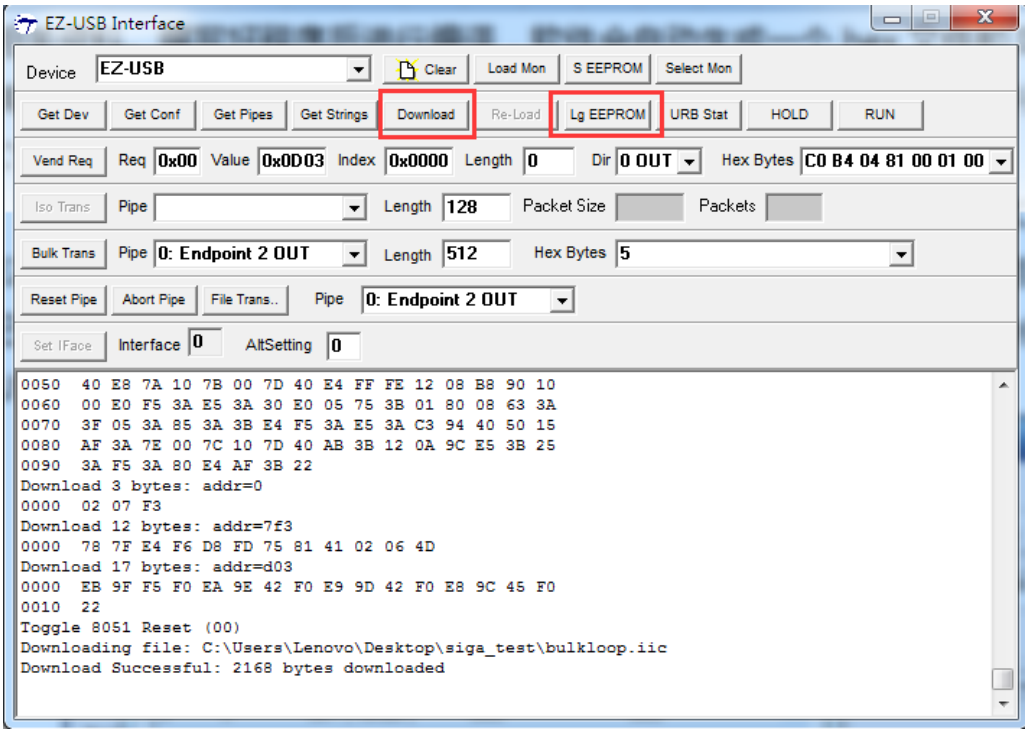
USB2.0 CY7C68013A 的固件程序可以用在 Keil uVision 开发环境中完成。用户需要的话，可以在我们提供的源码基础上进行二次开发。在开发板附带的光盘中有 Keil uVision 2 的安装软件，Keil uVision 2 的开发环境如下。



Keil uVision 2 的环境的使用在这里不作详细介绍，用户根据需要可以查看相关资料。编写好程序后进行编译，软件会自动生成一个 hex 文件和 IIC 文件，其中 hex 文件是下载到 CY7C68013

的 RAM 中的目标文件，重新上电后消失；IIC 文件是固化到 CY7C68013 外部的 EEPROM 中的目标文件，重新上电后不消失。

用我们提供的 USB 线连接 PC 的 USB 口和开发板上的 USB 接口(J6)后，我们可以通过 EZ-USB Interface 工具下载 Bit 文件或 IIC 文件。其中 Download 按钮是下载 hex 文件,Lg EEPROM 按钮是选择下载 IIC 文件。



3. FPGA Verilog 程序设计

在本测试程序中，会循环查询 CY7C68013A 的端口 2 的 FIFO 是否有数据，如果 CY7C68013A 的端口 2 的 FIFO 不空(PC 发送数据到端口 2)，程序会读取端口 2 的 FIFO 数据然后把它发送回到 CY7C68013A 端口 6 的 FIFO 中。

1.编写 usb 测试程序 usb_test.v

```
`timescale 10ns/1ns
//////////////////////////////////////////////////////////////////////////////////
// Module Name:    usb_test
// Description: If the FIFO of EP2 is not empty and
```

```
//          the EP6 is not full, Read the 16bit data from EP2 FIFO
//          and send to EP6 FIFO.
//////////////////////////////////////////////////////////////////////////////////
module usb_test(
   input sys_clk_p,
   input sys_clk_n,
   input reset_n,                  //FPGA Reset input
   output reg [1:0] usb_fifoaddr,      //CY68013 FIFO Address
   output reg usb_slcs,              //CY68013 Chipset select
   output reg usb_sloe,               //CY68013 Data output enable
   output reg usb_slrd,              //CY68013 READ indication
   output reg usb_slwr,               //CY68013 Write indication
   inout [15:0] usb_fd,              //CY68013 Data
   input usb_flaga,                 //CY68013 EP2 FIFO empty indication; 1:not empty; 0: empty
   input usb_flagb,                 //CY68013 EP4 FIFO empty indication; 1:not empty; 0: empty
   input usb_flagc                  //CY68013 EP6 FIFO full indication; 1:not full; 0: full

   );

reg[15:0] data_reg;

reg bus_busy;
reg access_req;
reg usb_fd_en;          //控制 USB Data 的方向

reg [4:0] usb_state;
reg [5:0] i;

parameter IDLE=5'd0;
```

```
parameter EP2_RD_CMD=5'd1;

parameter EP2_RD_DATA=5'd2;

parameter EP2_RD_OVER=5'd3;

parameter EP6_WR_CMD=5'd4;

parameter EP6_WR_OVER=5'd5;


////////////////差分时钟转换成单端时钟//////////////////////

   wire sys_clk_ibufg;

    IBUFGDS #

        (

         .DIFF_TERM    ("FALSE"),

         .IBUF_LOW_PWR ("FALSE")

         )

        u_ibufg_sys_clk

          (

           .I  (sys_clk_p),

           .IB (sys_clk_n),

           .O  (sys_clk_ibufg)

           );


/* Generate USB read/write access request*/

always @(negedge sys_clk_ibufg or negedge reset_n)

begin

  if (~reset_n ) begin

    access_req<=1'b0;

  end

  else begin

    if (usb_flaga & usb_flagc & (bus_busy==1'b0))    //如果 EP2 的 FIFO 不空，EP6 的 FIFO 不满，而且状态为 idle

      access_req<=1'b1;                    //USB 读写请求
```

```
    else
        access_req<=1'b0;
  end
end

/* Generate USB read and write command*/
always @(posedge sys_clk_ibufg or negedge reset_n)
 begin
  if (~reset_n) begin
              usb_fifoaddr<=2'b00;
      usb_slcs<=1'b0;
                    usb_sloe<=1'b1;
      usb_slrd<=1'b1;
                    usb_slwr<=1'b1;
      usb_fd_en<=1'b0;
              usb_state<=IDLE;
  end
  else begin
            case(usb_state)
     IDLE:begin
                                usb_fifoaddr<=2'b00;
           i<=0;
                                usb_fd_en<=1'b0;
                        if (access_req==1'b1) begin
                                    usb_state<=EP2_RD_CMD;          //开始读 USB EP2 FIFO 的数据
                                    bus_busy<=1'b1;                 //状态变忙
                                end
           else begin
                                    bus_busy<=1'b0;
```

```
                                        usb_state<=IDLE;
                                end
    end
    EP2_RD_CMD:begin          //发送 EP2 端口 FIFO 的数据读命令，先拉低 OE 信号，再拉低 RD 信号
                                if(i==11) begin
        usb_slrd<=1'b1;
        usb_sloe<=1'b0;                  //OE 信号变低
         i<=i+1'b1;
      end
      else if(i==35) begin
        usb_slrd<=1'b0;                //RD 信号变低
        usb_sloe<=1'b0;
         i<=0;
                                        usb_state<=EP2_RD_DATA;
      end
      else begin
                                        i<=i+1'b1;
                                end
    end
                EP2_RD_DATA:begin                  //读取 EP2 中的数据
      if(i==35) begin
        usb_slrd<=1'b1;                //RD 信号变高,读取数据
        usb_sloe<=1'b0;
         i<=0;
                                        usb_state<=EP2_RD_OVER;
                                data_reg<=usb_fd;                  //读取数据
      end
      else begin
```

```
        usb_slrd<=1'b0;
        usb_sloe<=1'b0;
                                    i<=i+1'b1;
                            end
                end
                EP2_RD_OVER:begin
    if(i==19) begin
        usb_slrd<=1'b1;              //OE 信号变高,读取数据完成
        usb_sloe<=1'b1;
         i<=0;
                                usb_fifoaddr<=2'b10;
                                    usb_state<=EP6_WR_CMD;
    end
    else begin
        usb_slrd<=1'b1;
        usb_sloe<=1'b0;
                                    i<=i+1'b1;
                            end
                end
                EP6_WR_CMD:begin                        //EP6 端口写入数据，slwr 变低后，再变高
    if(i==35) begin
        usb_slwr<=1'b1;
        i<=0;
                                usb_state<=EP6_WR_OVER;
    end
    else begin
         usb_slwr<=1'b0;
                                    usb_fd_en<=1'b1;            //数据总线改为输出
```

```
                                        i<=i+1'b1;
                                end
                        end
                        EP6_WR_OVER:begin                                   //EP6 写完成
        if(i==19) begin
          usb_fd_en<=1'b0;
                                bus_busy<=1'b0;
          i<=0;
                                usb_state<=IDLE;
        end
        else begin
                                 i<=i+1'b1;
                               end
                        end
    default:usb_state<=IDLE;
    endcase
           end
 end


assign usb_fd = usb_fd_en?data_reg:16'bz;      //USB 数据总线输入输出改变


endmodule
```

4. 管脚约束

编写 FPGA 的管脚约束文件 usb.xdc。管脚的约束必须和开发板的硬件设计一致，如下所示:

```
set_property CFGBVS VCCO [current_design]

set_property CONFIG_VOLTAGE 3.3 [current_design]

############### clock define###################
```

```
create_clock -period 5 [get_ports sys_clk_p]

set_property PACKAGE_PIN R4 [get_ports sys_clk_p]

set_property IOSTANDARD DIFF_SSTL15 [get_ports sys_clk_p]


###############reset key define###################

set_property PACKAGE_PIN T6 [get_ports rst_n]

set_property IOSTANDARD LVCMOS15 [get_ports rst_n]


############## USB define###################

set_property IOSTANDARD LVCMOS33 [get_ports usb_slcs]

set_property PACKAGE_PIN P16 [get_ports usb_slcs]


set_property IOSTANDARD LVCMOS33 [get_ports usb_sloe]

set_property PACKAGE_PIN N13 [get_ports usb_sloe]


set_property IOSTANDARD LVCMOS33 [get_ports usb_slrd]

set_property PACKAGE_PIN P19 [get_ports usb_slrd]


set_property IOSTANDARD LVCMOS33 [get_ports usb_slwr]

set_property PACKAGE_PIN R19 [get_ports usb_slwr]


set_property IOSTANDARD LVCMOS33 [get_ports usb_flaga]

set_property PACKAGE_PIN R18 [get_ports usb_flaga]


set_property IOSTANDARD LVCMOS33 [get_ports usb_flagb]

set_property PACKAGE_PIN R14 [get_ports usb_flagb]


set_property IOSTANDARD LVCMOS33 [get_ports usb_flagc]

set_property PACKAGE_PIN P14 [get_ports usb_flagc]
```

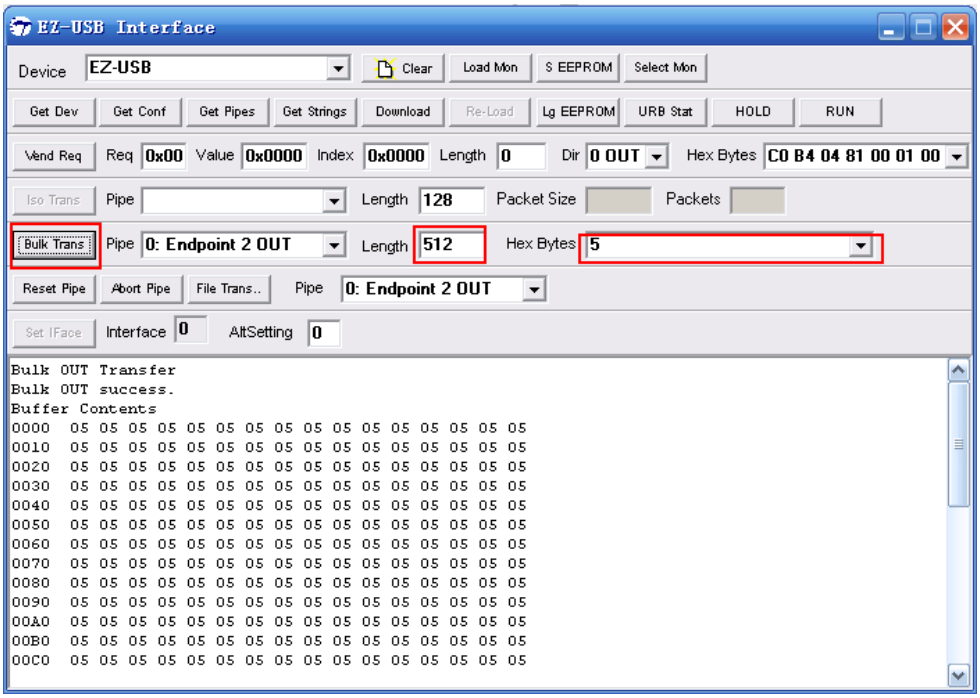
```
set_property IOSTANDARD LVCMOS33 [get_ports {usb_fifoaddr[0]}]
set_property PACKAGE_PIN N14 [get_ports {usb_fifoaddr[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fifoaddr[1]}]
set_property PACKAGE_PIN N15 [get_ports {usb_fifoaddr[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[0]}]
set_property PACKAGE_PIN Y22 [get_ports {usb_fd[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[1]}]
set_property PACKAGE_PIN W20 [get_ports {usb_fd[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[2]}]
set_property PACKAGE_PIN W19 [get_ports {usb_fd[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[3]}]
set_property PACKAGE_PIN Y19 [get_ports {usb_fd[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[4]}]
set_property PACKAGE_PIN Y18 [get_ports {usb_fd[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[5]}]
set_property PACKAGE_PIN V22 [get_ports {usb_fd[5]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[6]}]
set_property PACKAGE_PIN U22 [get_ports {usb_fd[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[7]}]
```

```
set_property PACKAGE_PIN T18 [get_ports {usb_fd[7]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[8]}]
set_property PACKAGE_PIN R17 [get_ports {usb_fd[8]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[9]}]
set_property PACKAGE_PIN R16 [get_ports {usb_fd[9]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[10]}]
set_property PACKAGE_PIN P15 [get_ports {usb_fd[10]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[11]}]
set_property PACKAGE_PIN N17 [get_ports {usb_fd[11]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[12]}]
set_property PACKAGE_PIN P17 [get_ports {usb_fd[12]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[13]}]
set_property PACKAGE_PIN U16 [get_ports {usb_fd[13]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[14]}]
set_property PACKAGE_PIN T16 [get_ports {usb_fd[14]}]


set_property IOSTANDARD LVCMOS33 [get_ports {usb_fd[15]}]
set_property PACKAGE_PIN U17 [get_ports {usb_fd[15]}]
```

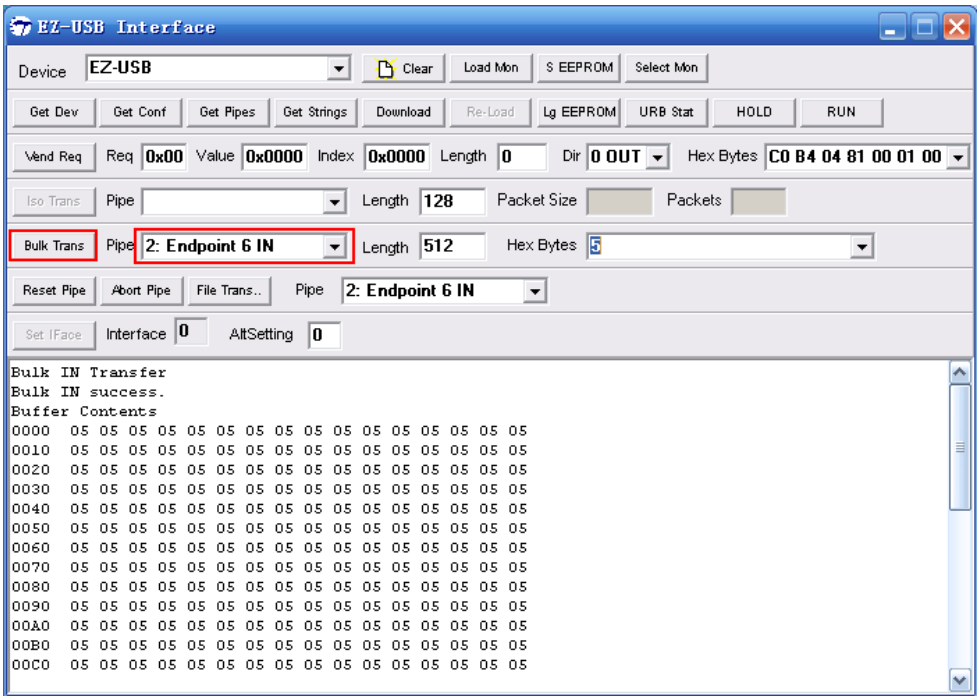
4 下载和试验

编译生成 usb_test.bit 文件。下载 usb_test.bit 到 FPGA，再打开 EZ-USB Interface 的通信工具。

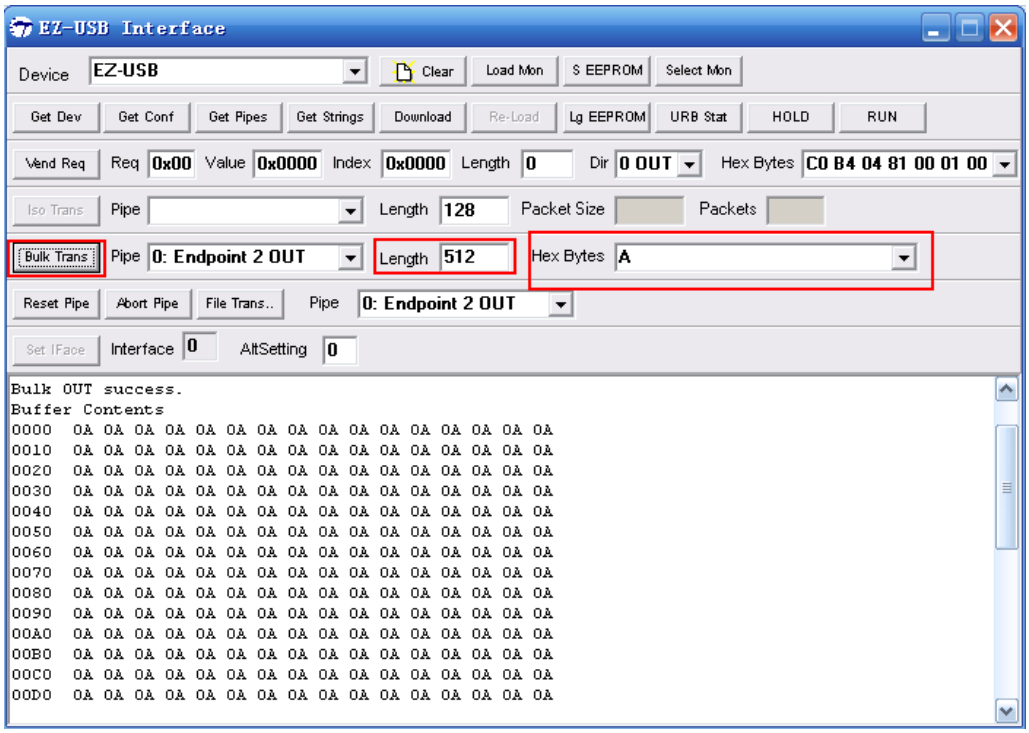
1.发送 512 个数据 5 到端口 2(选择 Endpoint 2 OUT 并点击“Bulk Trans”，发送数据到 Endpoint 2 的 FIFO 中)。



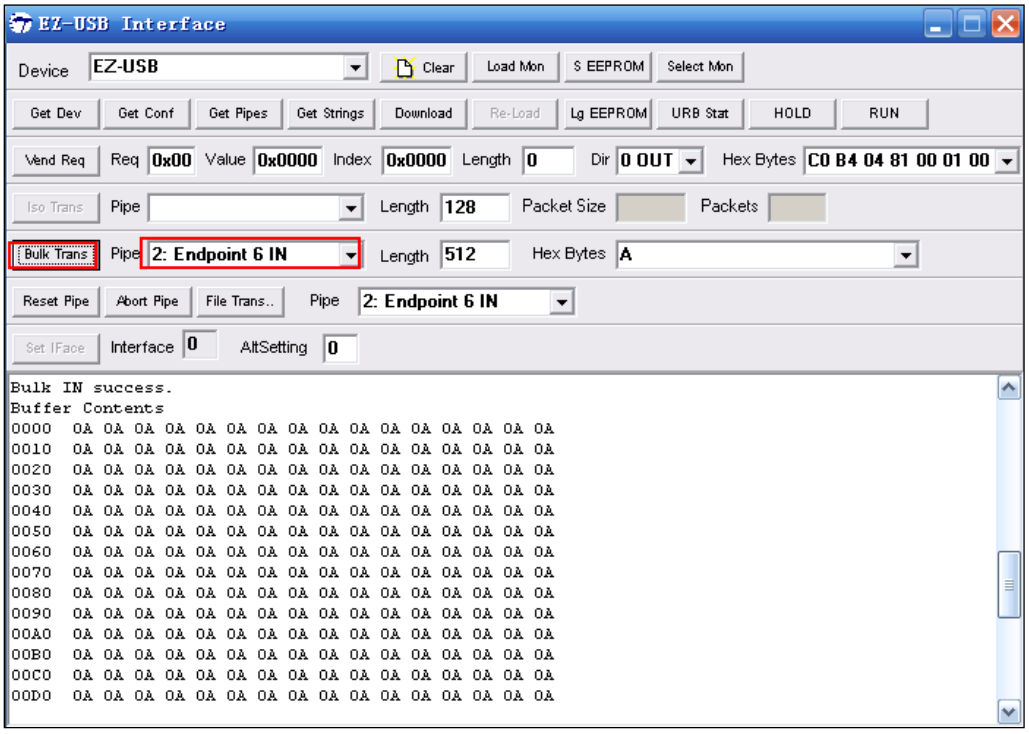
FPGA 的 USB 通信程序会自动读取 Endpoint 2 的 FIFO 数据，并写入到 Endpoint 6 的 FIFO 中。在 EZ-USB Interface 的通信工具中选择 Endpoint 6 IN，并点击“Bulk Trans”。512 个数据 5 显示在窗口中。



2. 我们再发送 512 个数据 A 到端口 2(选择 Endpoint 2 OUT 并点击“Bulk Tran”，发送数据到 Endpoint 2 的 FIFO 中)。



在 EZ-USB Interface 的通信工具中选择 Endpoint 6 IN，并点击“Bulk Trans”。512 个数据 A 显示在窗口中。



到此为止，USB2.0 数据通信的实验就讲完了，这部分的内容比较多，需要掌握的知识也比较多，很多东西需要大家好好的去体会和摸索。另外本实验只实现 CY7C68013 的异步 Slave 模式的数据通信，如果用户想提高 FPGA 和 CY7C68013 的通信速度，可以尝试修改 CY7C68013 的固件程序（设置 IFCONFIG 寄存器的 Bit3 为 0），使 USB 芯片工作在同步 Slave 模式。关于如何如何设置 CY7C68013 寄存器及同步 Slave 工作模式的读写时序大家可以参考“EZ-USB_TRM.pdf”和 CY7C68013 的芯片手册。