实验十三：SPI 通信实验——基于 SPI 总线的 ARM 与 FPGA 通信

一、 实验目的与意义

1、掌握 SPI 通信协议及实现方法。

2、掌握 QuartusII 的使用方法。

二、 实验设备及平台

1、iCore4 双核心板。

2、Blaster（或相同功能）仿真器。

3、JLINK（或相同功能）仿真器。

4、Micro USB 线缆。

5、Keil MDK 开发平台。

6、Quartus 开发平台。

7、电脑一台。

三、 实验原理

1. SPI 简介

SPI 是串行外设接口（Serial PeripheralInterface）的缩写。SPI，是一种高速的，全双工，同步的通信总线，并且在芯片的管脚上只占用四根线，节约了芯片的管脚，同时为 PCB 的布局上节省空间，提供方便，正是出于这种简单易用的特性，如今越来越多的芯片集成了这种通信协议。SPI 的通信原理很简单，它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以（单向传输时）。也是所有基于 SPI 的设备共有的，它们是 SDI（数据输入）、SDO（数据输出）、SCLK（时钟）、CS（片选）。

SPI 硬件接口：

MISO ：主设备数据输入，从设备数据输出

MOSI ：主设备数据输出，从设备数据输入

SCLK ：时钟信号，由主设备产生

CS ：从设备片选信号，由主设备控制

2. SPI 功能说明

SPI 时钟极性和相位:

CPOL 决定时钟空闲时的稳定电平，对主/从都有效

CPOL=0：空闲时低电平

CPOL=1：空闲时高电平

CPHA 决定数据采样时刻

CPHA=0：第一个时钟延开始采样 MSBit

CPHA=1：第二个时钟延开始采样 MSBit

SPI 总线四种工作方式 SPI 模块为了和外设进行数据交换，根据外设工作要求，其输出串行同步时钟极性和相位可以进行配置，时钟极性（CPOL）对传输协议没有重大的影响。如果 CPOL=0，串行同步时钟的空闲状态为低电平；如果 CPOL=1，串行同步时钟的空闲状态为高电平。时钟相位（CPHA）能够配置用于选择两种不同的传输协议之一进行数据传输。如果 CPHA=0，在串行同步时钟的第一个跳变沿（上升或下降）数据被采样；如果 CPHA=1，在串行同步时钟的第二个跳变沿（上升或下降）数据被采样。SPI 主模块和与之通信的外设备时钟相位和极性应该一致。如图 13-1 所示

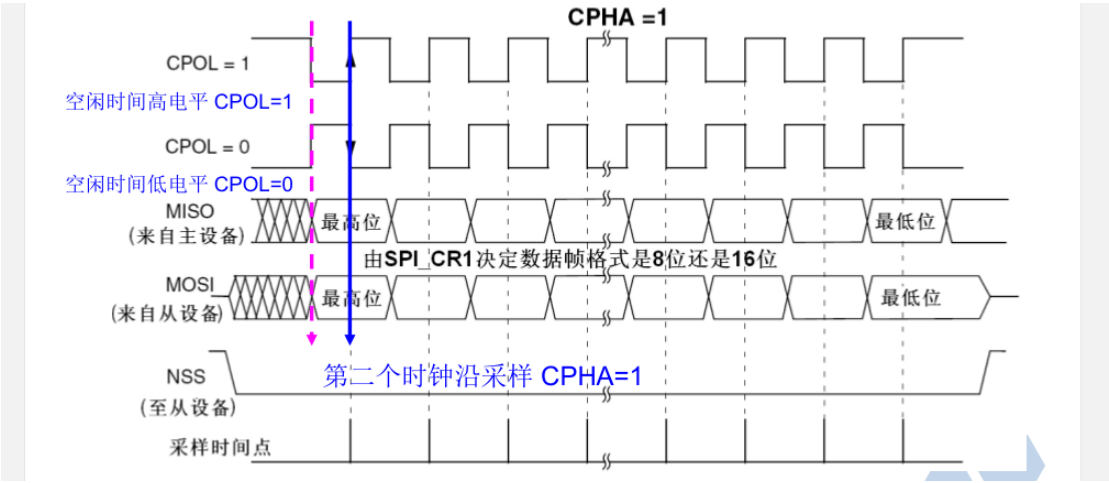


图 13-1

3．SPI 通信指令表

指令名称	字节 1	字节 2	字节 3	字节 4
器件 ID	01h			
写数据长度	02h	A15~A8	A7~A0	

写数据	04h	A15~A8	A7~A0	数据（直至写完所有数据）
读数据长度	05h	A15~A8	A7~A0	
读数据	07h	A15~A8	A7~A0	数据（直至读完所有数据）
读错误信息	08h			

表 18.1 SPI 通信指令表

ARM 与 FPGA 通信采用的是半双工式通信，FPGA 通过识别指令完成与 ARM 的交互。

器件 ID 指令为 01h，接下来为两字节的伪指令，第四字节仍为伪指令读取 ID 标志。写数据长度指令 02h，接下来两个字节为写数据的长度，先发高字节，后发低字节，接下来为一个字节的伪指令 00h。

写数据指令为 04h，接下来为两字节的地址指令，后为要写入的数据，数据写入完毕以伪指令 00h 结束数据传输。

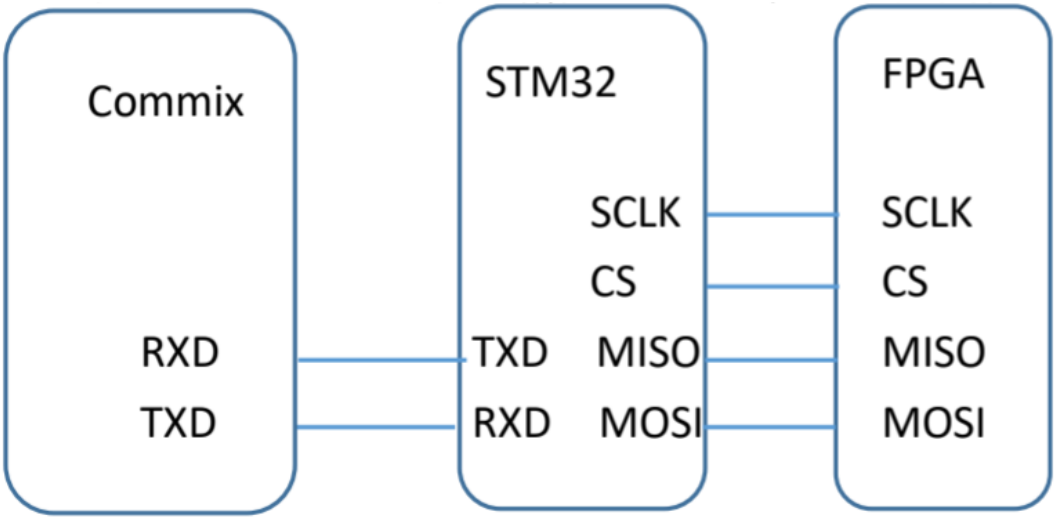
读数据长度指令 05h，接下来两个字节为写数据的长度，先发高字节，后发低字节，接下来为一个字节的伪指令 00h。

读数据指令为 07h，接下来为两字节的地址指令，其后为伪指令 00h 开始读取数据进行数据传输，第五字节以后为要读取的数据。

读错误信息指令 08h，FPGA 接收数据是否出错，先发送两个字节的伪指令，第四字节仍为伪指令读取错误标志信息。

4、本实验实现方式

通过 FPGA 建立的 SPI 模块对外提供的 SCLK、CS、MOSI、MISO 接口与 STM32 的 SPI 相连接，Commix 串口精灵与 STM32 通过串口连接，实现三者之间的通信。本实验中，Commix 串口精灵向 STM32 发送数据，STM32 的 RXD 端口接收数据，然后，通过 TXD 端口把数据发送至 FPGA，STM32 起到一个桥梁作用。程序运行后，FPGA 定时向 STM32 发送数据，经过 STM32 发送至串口精灵显示出来。下图为实验原理图。



四、 代码讲解

本实验基于 ARM+FPGA 构架，通过 ARM 首先发送查询 ID 指令，然后依次通过指令设置写数据的长度、写数据地址、读数据的长度、读数据的地址，然后对比写入数据和读出数据是否一致判断数据是否写入成功，从而基于 SPI 总线实现 ARM 与 FPGA 之间的通信。

1、实现 ARM 与 FPGA 通信，对 FPGA 而言，其关键就在于 SPI 时序的模拟，实现 SPI 数据的接收与发送，实现数据与传输信号之间的串并转换。FPGA 首先接收 ARM 指令，然后解析指令，存储相应的信息与数据，并根据指令需求将相应的指令数据放到 SIMO 总线上，等待 ARM 读取，从而实现两者之间的数据交互。SPI 时序的硬件语言描述如下：

```
////////////////按字节接收 SPI 发送过来的数据/////////////
///////接收模块///////
reg[3:0]i;
reg[7:0]data_in;
reg [39:0]temp_data,data;

always@(posedge spi_clk or negedge rst_n)
    if(!rst_n)
        begin
            i <= 4'd0;
            temp_data <= 40'd0;
            data <= 40'd0;
            data_in <= 8'd0;
        end
    else case(i)    //从高位开始接收数据，每 8 个 spi_clk 时钟接收一个 Byte
        4'd0:
            begin
```

```
                i <= i + 1'd1;
                data_in <= {data_in[6:0],spi_mosi};
                temp_data <= {temp_data[31:0],data_in};
                if(data_in == 8'd13)
                    begin
                        data <= temp_data;
                    end
                else
                    begin
                        data <= data;
                    end
            end
        4'd1,4'd2,4'd3,4'd4,4'd5,4'd6:
            begin
                i <= i + 1'd1;
                data_in <= {data_in[6:0],spi_mosi};
            end
        4'd7:begin
                i <= 4'd0;
                data_in <= {data_in[6:0],spi_mosi};
            end
        default: i <= 4'd0;
    endcase

/*对比接收数据*/
reg [2:0]led;

always@(posedge clk_25m or negedge rst_n)
    if(!rst_n)
        begin
            led <= 3'b101;
        end
    else if (data == ledr)
            led <= 3'b011;                  //红灯亮
    else if (data == ledg)
            led <= 3'b101;                              //绿灯亮
    else if (data == ledb)
            led <= 3'b110;                              //蓝灯亮

assign {led_red,led_green,led_blue} = led;

//--------------------------delay----------------------------//
    reg spi_clk_r;
    always@(posedge clk_25m or negedge rst_n)
```

```
        if(!rst_n)
            begin
                spi_clk_r <= 1'd1;
            end
        else
            spi_clk_r <= spi_clk;

//--------------------------spi_miso-----------------------------//
/*发送模块*/
reg [39:0]data_out;
reg [5:0]j;
reg spi_out;
always@(negedge spi_clk_r or negedge rst_n)
    if(!rst_n)
        begin
            data_out <= hello;
            spi_out <= 1'd0;
            j <= 6'd0;
        end
    else case(j)  //连续 40 个 spi_clk_r 时钟发送"hello"字符串
        6'd0:
            begin
                {spi_out,data_out[39:1]} <= data_out;
                j <= j + 1'd1;
            end
        6'd39:
            begin
                {spi_out,data_out[39:1]} <= data_out;
                data_out <= hello;
                j <= 6'd0;
            end
        default:
            begin
                {spi_out,data_out[39:1]} <= data_out;
                j <= j + 1'd1;
            end
    endcase
```

五、 实验步骤

1、把仿真器与 iCore4 的 SWD 调试口连接（直接相连或者通过转换器相连）。

2、将 USB-Blaster 与 iCore4 的 JTAG 调试口相连。

3、将跳线帽插在 USB_UART。

4、把 iCore4（USB_UART）通过 Micro USB 线与计算机相连，为 iCore4 供电。

以上操作如图 13-2 所示：

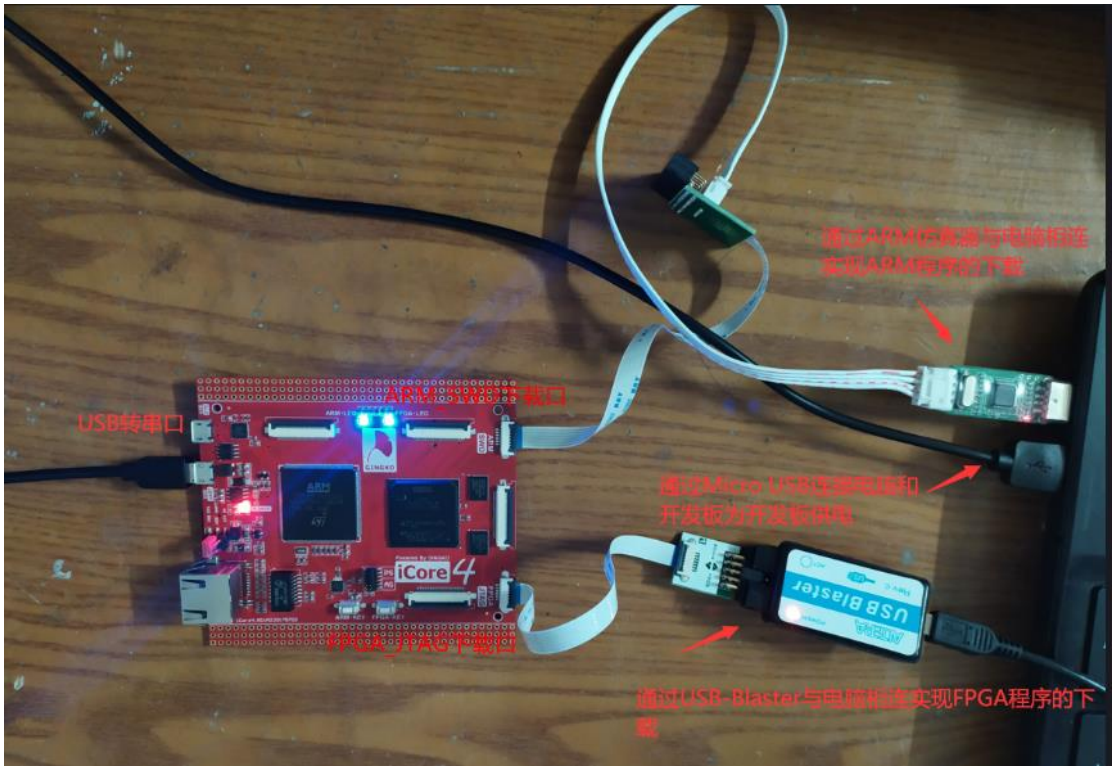


图 13-2

5、打开串口精灵，找到对应口打开，如下图。

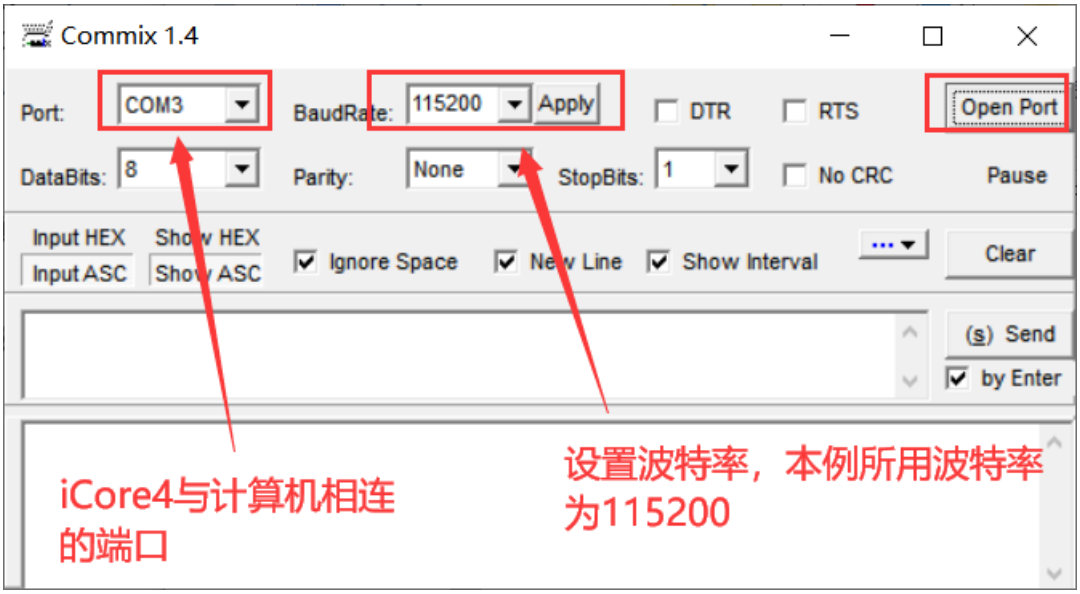


图 13-3

6、打开 KeilMDK 开发环境，并打开实验工程。

7、将 ARM 程序下载至 iCore4。

8、打开 QuartusII 开发环境，并打开实验工程。

9、将 FPGA 程序下载至 iCore4

10、输入串口命令，观察实验现象。

六、 实验现象

在 Commix 上发送命令后，对应的 ARM 和 FPGA 的 LED 灯亮，同时接收显示：hello

串口命令发送格式	ARM_LED 现象	FPGA_LED 灯现象
LEDR\CR\LF	红灯亮	红灯亮
LEDG\CR\LF	绿灯亮	绿灯亮
LEDB\CR\LF	蓝灯亮	蓝灯亮