摘 要

本系统是基于可见光的通信装置，包括电源模块、FPGA 编码模块、FPGA 解码模块、发射接收模块。模拟信号经 AD 转换传输到 FPGA 可编程器件进行编码调制，减少噪声干扰，使用大功率白光 LED 作为系统的通信光源，利用硅光电池接收，接收后通过 FPGA 解调及 DA 转换成模拟信号放大驱动 8 欧姆的喇叭后放音。

关键词：可见光通信；大功率白光 LED；数字传输；FPGA 调制解调；硅光电池

Abstract： This system is a communication device based on visible light. Analog signal through the AD conversion is transferred to the FPGA programmable devices are coded modulation, reduce noise interference, the use of high power white LED as light source of the communication system, using silicon light battery receiving, receiving through demodulation and DA FPGA converted into analog signal amplification to drive 8 ohm speaker playback.

Keyword： Visible light communication, digital transmission, modulation and demodulation of FPGA, Silicon photocel

目录

一. 方案选择与证论

1.1 通信方式

方案一：数字通信：数字调制驱动电路，用 FPGA 可编程逻辑器件作为数据处理的核心，数据编码后以 RS232 协议驱动电路，经过施密特触发器滤除噪声后，由 ADM3202AN 电平转换，经过场效应管 2SK2232 驱动 LED 发出光信号。本方案有较高的速度以及很高的精度，抗干扰能力强，失真小。

方案二：模拟调制解调：要求带宽足够宽，且需要较高的发射频率，效率较低，波形易失真且易受干扰；难以实现发挥部分两路传输信号及 M 序列。

综上所述知，我们小组选择方案一，这样设计更方便，更容易满足要求。

1.2 发送模块

方案一：莫斯管直接驱动 LED。方案简单，外围器件少，但失真较严重

方案二：编码后的数据经过施密特触发器滤除噪声后，到达集成电路元件 ADM3202AN，经过 ADM3202AN 的电平转换后，由场效应管 2SK2232 驱动 LED 发出光信号。

所以经过分析比较，本次设计选用方案二来实现。

1.3 接收模块

方案一：红外接收头接收，经运放放大后直接输出。

方案二：硅光二极管接收到空间信道中传输的光信号，LM7171 和 74HC14 对接收波形的整形和滤除噪声，送到后续解码和控制单元 FPGA

比较：方案一简单易行，但接收效率低，失真率高。方案二对频率要求高，但接收失真小。

经分析比较，本设计选用方案二。

1.4 调制解调模块

硬件：混频器和解调器。对信号要求高，易受噪声干扰。

软件：AD/DA+FPGA 调制解调。信号编码，噪声影响小，但对软件要求高

所以经过分析比较，本次设计选用方案二来实现。

综上所述，得出总体系统框图

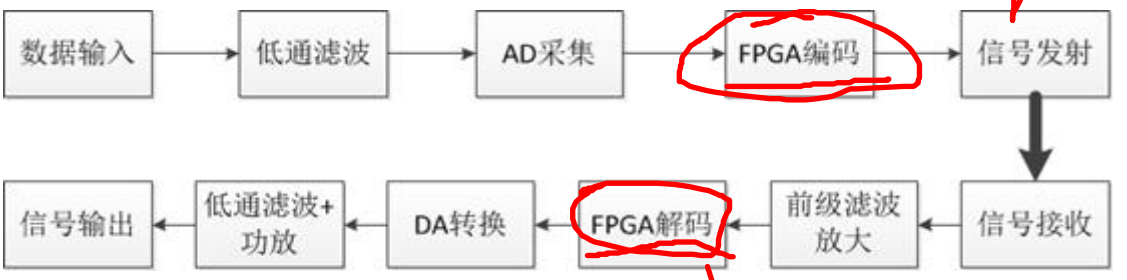

图 1 总体系统框图

二. 理论分析与计算

2.1 白光 LED 通信分析与比较

发光二极管，通常称为 LED，它的发光是由于半导体材料中电子的移动。由于 LED 没有灯丝，因此不会被烧坏，寿命长。白光 LED 的优点除了发光效率高、电压驱动低、低耗电量、良好的稳定性、寿命长、可靠安全、利于环保等普遍特点外，还包括显色性好、高光效、光通量大等诸多优点。此外白光 LED 的响应时间很短，一般为纳秒级。因此，白光 LED 的调制频率很高，适合作为室内照明通信系统的光源。

表 1-1 可见光通信与红外无线通信比较

典型特性	红外光	白光
信号光源	红外 LED，红外 LD	白光 LED
工作波长	800-900nm	380-789nm
光谱宽度	< 5nm(LD)	25-100nm
调制带宽	几十 KHz 到几百 MHz（LED） 几十 KHz 到几十 GHz（LD）	几十 KHz 到几百 MHz
线路布局	需另架设红外通信光源和线路	兼顾照明，简化了布局
信道速率	理论速率 100Mb/s	最初上线速率 10Mb/s
发射功率	需限制发射功率	通常情况下无需限制
码间串扰	多径效应引起码间串扰	多径效应引起码间串扰
阴影效应	容易受其它遮挡物影响	可消除阴影效应

与红外通信相比，可见光通信更具优势与应用前景。

2.2可见光发射的基本框图

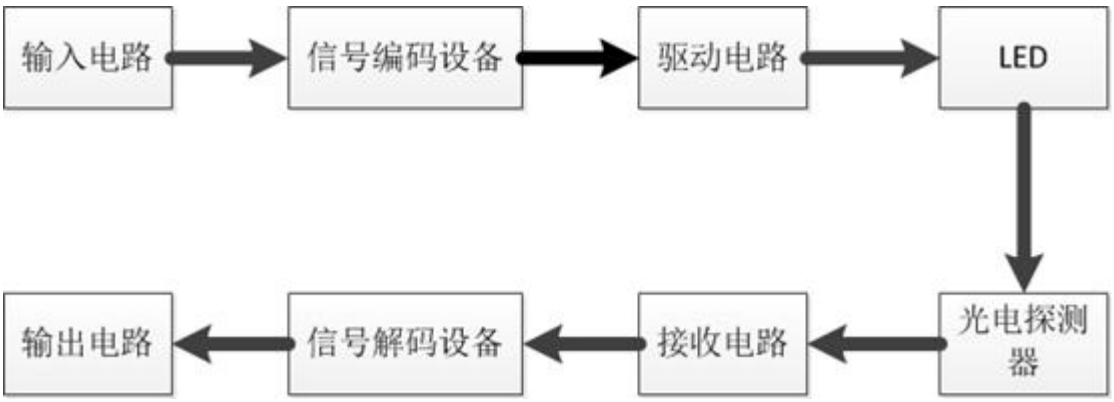

图 2 基本框图

三. 硬件电路设计

3.1 发射模块

LED 的驱动电路使用高速 MOS 管的开关作用，使得传输的数据加载在直流分量上通过 LED 发射。

原理图见 附录一 图 4 发射电路

3.2 接收模块

LED 发出的光经空间信道传输会产生很大衰减，接收电路需要尽可能的放大现有信号，以最大的幅度和最稳定的频率将此光信号发送给 FPGA。LM7171 和 74HC14 对接收波形进行整形和滤除噪声的作用。

原理图见 附录一 图 5 接收电路

3.3 电源模块

有 220-24V 变压器将 220 交流变为 24V 交流，经过全波整流滤波，最后由 LM7824 稳压输出。

原理图见 附录一 图 6 电源电路

四. 软件设计

可见光通信易受自然光的干扰，而数字通信将模拟信号编为数字量，在传输过程中受自然光等其他条件影响小，因此引入 FPGA 对模拟信号进行编码，食用串口通信协议送到 LED 灯，接收端 FPGA 接收解码，存储缓冲一定数据后送出，经 DA 还原为模拟信号。

4.1 软件流程图

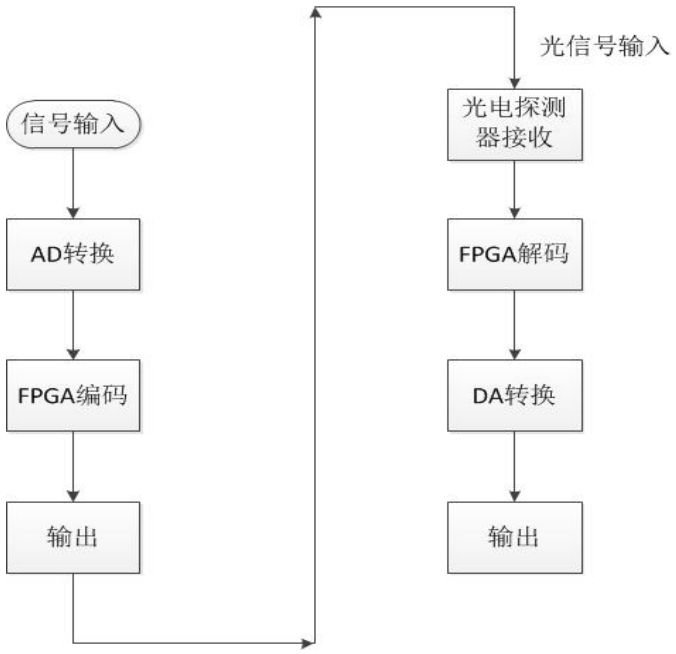

图 3 程序流程图

4.3 系统软件

见附录三

五. 测试方案和测试结果

测试仪器见　附录二.元器件列表

5.1 基本部分电压、电流的测试

表 1 电压电流测试

U1	24V
I1	1.01A

U2	10.2V
I2	0.99A

5.2　发射 300-8000Hz 频率的信号测试

表 2 信号接收测试

输入频率	波形是否正常	波形有无失真	幅值
300	是	无	0.43
1000	是	无	0.45
8000	是	无	0.41

5.3　A、B 路模拟信号波形测试

表 3 双路传输测试

		频率(Hz)	接收装置电压有效值（V）
A 路传输信号	模拟信号	300-8000	0.42
	单音信号	2000	0.43
		8000	0.45
	方波	<1000	0.42
	锯齿波	<1000	0.41
B 路传输信号	模拟信号	300-8000	0.43
	单音信号	2000	0.45
		8000	0.42
	方波	<1000	0.41
	锯齿波	<1000	0.44

5.4 M 序列测试

发射与接收均无明显码间串扰

误差分析：

1．受自然光影响产生偏差

2．连接线较长，影响信号传输

3．器件对高频响应不理想

六．结论

（1）本系统完成了题目的基本部分和发挥部分的全部要求。实验测试结果表明,本装置通信性能稳定，失真少。

（2）双信号编码传输

（3）1MHZ 传输频率，信道宽，传码率高

总结

本系统设计使用FPGA作为信息处理核心实现了大功率白光LED的可见光通信的设计。在设计中，我们既考虑了系统的通信能力，又考虑了系统中芯片的稳定性。通过测试，系统不但完成了题目要求，而且还扩展了相应功能。经过几天不断的改进电路和程序，一点点的攻克难关，最终很好的完成了设计，在比赛过程中，首先要感谢指导老师对我们的帮助和指导，也要感谢我们的队员，有了他们，我们才能更好的完成设计。对自身能力来说也是一种质的提高，也充分体现了团队合作的重要性。在以后的学习和生活中我们还需继续努力，不断学习，不断改善自己，以创造更好的成绩。

参考文献：

1. 《德州仪器高性能单片机和模拟器件在高校中的应用和选型》

2. 黄智伟．《全国大学生电子设计竞赛训练教程》．电子工业出版式社，2005 年第 1 版

4. 夏宇闻，黄然等．《Verilog　SOPC 高级实验教程》．北京航空航天出版社，2009 年第一版

5. 童诗白，华成英．《模拟电子技术基础》．高等教育出版社，2009 年第四版

6．王松武．《电子创新设计与实践》．国防工业出版社，2005 年第一版

7．《On Amps For Everyone Third Edition》　Bruce Carter　Ron Mancini　人民邮电出版社

附录一 模块电路

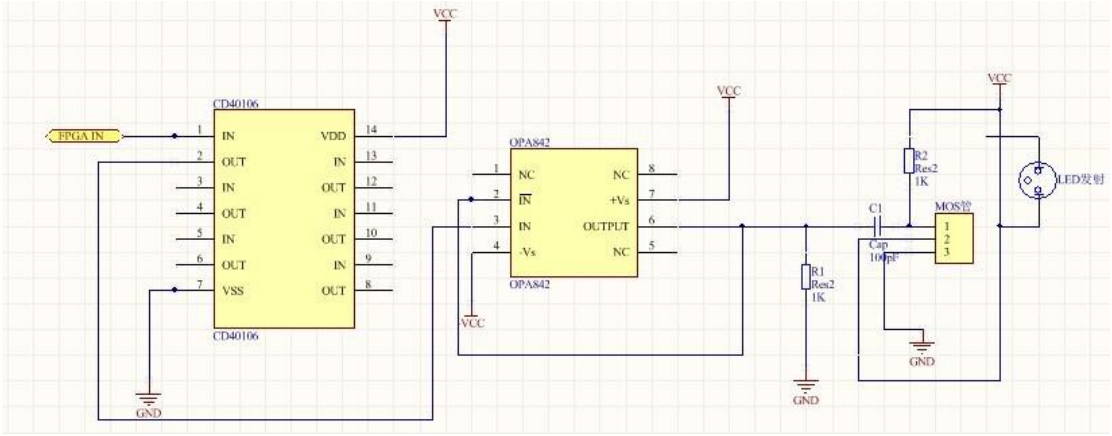

图 4　发射电路

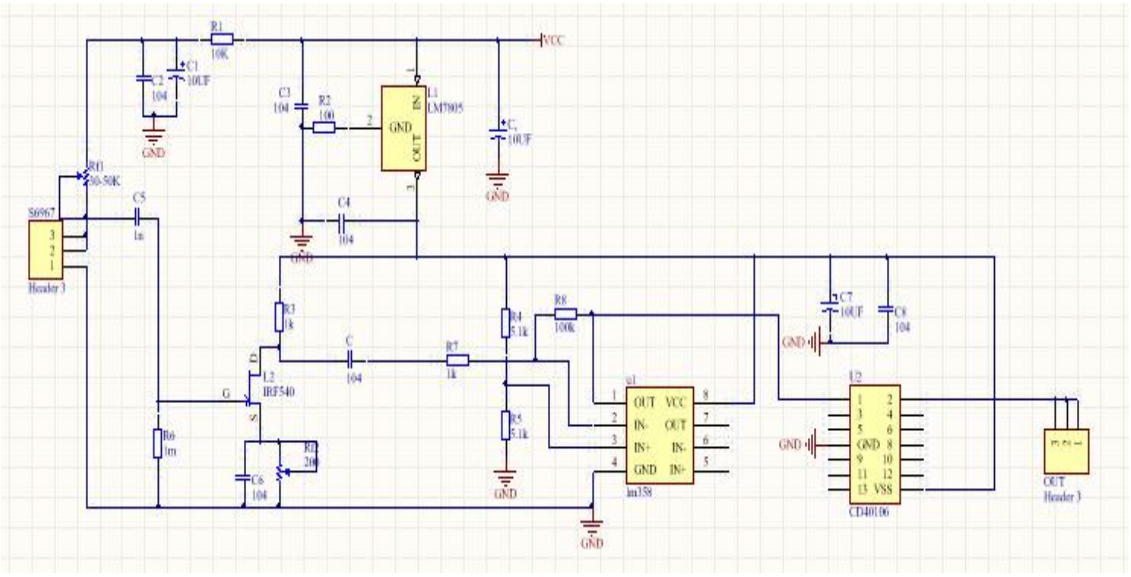

图 5 接收电路

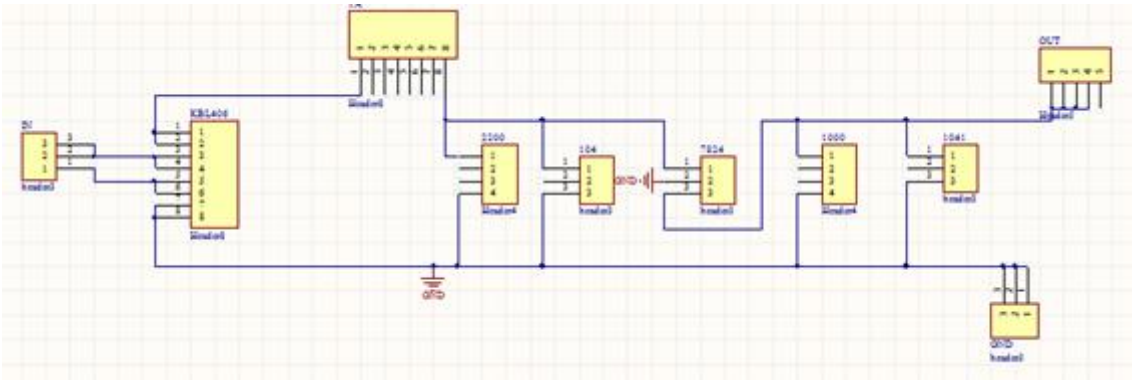

图 6 电源模块

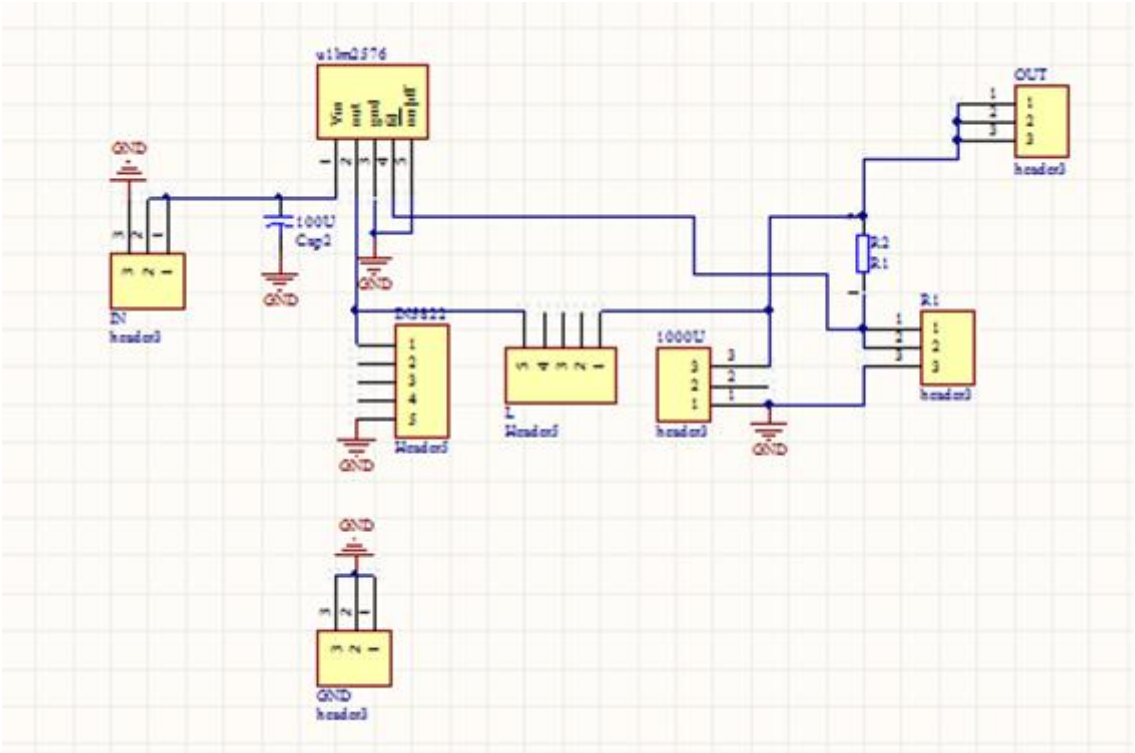

图 7 Lm2576 降压模块

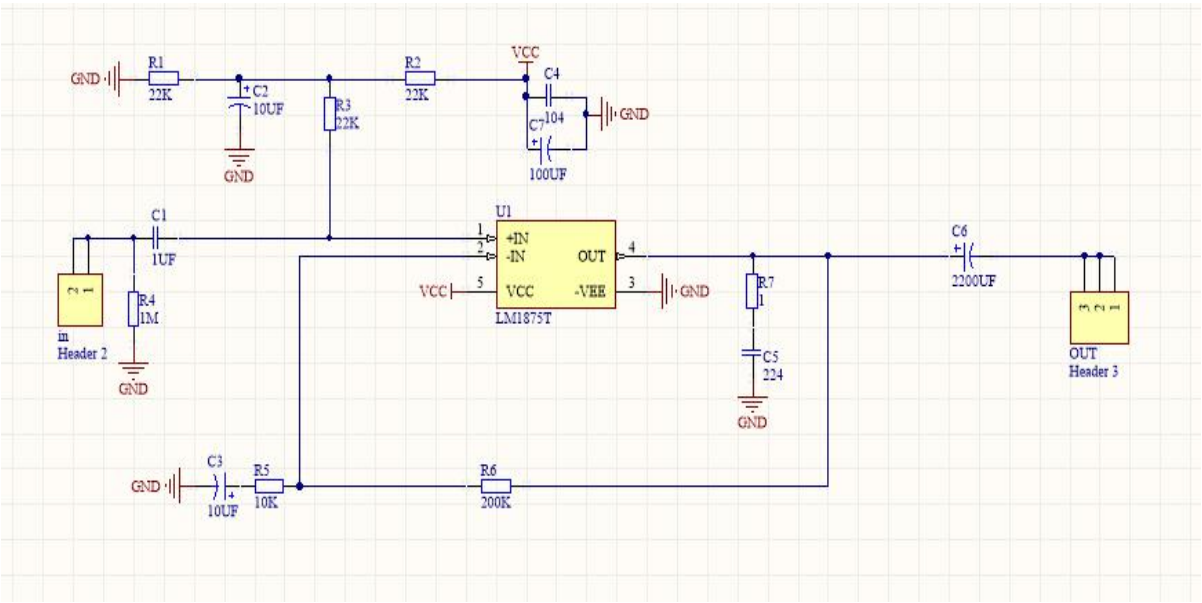

图 8 LM1875 功放

附录二.元器件列表

元器件	个数
24V 变压器	2
LM2576	4
MOS 管	3
10WLED 灯	1

INA128	1
LM386	1
硅光电池 SP-804	1

附录三.部分源程序代码

(1)clk_1M.v:

```
      module clk_1M(clk,rst,clk_1M);
input clk,rst;
output clk_1M;

reg[31:0]cnt;
reg clk_1M;
always@(posedge clk or negedge rst)
begin
    if(!rst)
        begin
            cnt<=1'b0;
        end
    else if(cnt>=7)
        begin
            clk_1M<=~clk_1M;
            cnt<=1'b0;
        end
    else
        begin
            cnt<=cnt+1'b1;
        end
end


endmodule
```

(2)da_data.v

```
module da_data(clk_16M,
done,clk_da,
in_data,
Aout_data,Bout_data
);


input clk_16M,done;
input [15:0]in_data;
```

```
output[7:0]Aout_data;
output[7:0]Bout_data;
output clk_da;

reg clk_da;
always@(posedge clk_16M)
begin
    clk_da<=done;
end

wire[7:0] Aout_data=in_data[7:0];
wire[7:0] Bout_data=in_data[15:8];

endmodule
```

(3)my_uart_rx.v

```
module my_uart_rx(clk,rst,rs232_rx,rx_data_out,clk_bps,bps_start);
input clk;// 50MHz 主时钟
input rst;//低电平复位信号
input rs232_rx;   // RS232 接收数据信号
input clk_bps;    // clk_bps 的高电平为接收或者发送数据位的中间采样点

output bps_start;//接收到数据后，波特率时钟启动信号置位
output[3:0] rx_data_out;//接收数据寄存器，保存直至下一个数据来到
//output rx_int; //接收数据中断信号,接收到数据期间始终为高电平


reg rs232_rx0,rs232_rx1,rs232_rx2,rs232_rx3; //接收数据寄存器，滤波用
always @ (posedge clk or negedge rst) begin
    if(!rst) begin
           rs232_rx0 <= 1'b1;
           rs232_rx1 <= 1'b1;
           rs232_rx2 <= 1'b1;
           rs232_rx3 <= 1'b1;
       end
    else begin
           rs232_rx0 <= rs232_rx;
           rs232_rx1 <= rs232_rx0;
           rs232_rx2 <= rs232_rx1;
           rs232_rx3 <= rs232_rx2;
       end
end
```

```
wire neg_rs232_rx;   //表示数据线接收到下降沿
assign  neg_rs232_rx  =  rs232_rx3&&rs232_rx2&&~rs232_rx1&&~rs232_rx0;
   //接收到下降沿后 neg_rs232_rx 置高一个时钟周期

reg bps_start;
reg[3:0] num;
reg rx_int;


always@(posedge clk or negedge rst)
   if(!rst)
   begin
      bps_start<=1'b0;
      rx_int<=1'b0;
   end
   else if(neg_rs232_rx)//接收到串口接收线的下降沿
   begin
      bps_start<=1'b1;
      rx_int<=1'b1;
   end
   else if(num==4'd12)//接受完数据位
   begin
      bps_start<=1'b0;
      rx_int<=1'b0;
   end

reg[7:0] rx_data;
reg a; //数据移位标志

always @ (posedge clk or negedge rst)
 begin
   if(!rst)
       begin
          a<= 1'b0;
          num <= 4'd0;
          rx_data <= 8'd0;
       end
   else if(rx_int)
      begin  //接收数据处理
          if(clk_bps)
              begin  //读取并保存数据,接收数据为一个起始位，8bit 数据，
一个结束位
```

```
                    a<= 1'b1;
                    num <= num+1'b1;
                    if(num<=4'd8)
                    rx_data[7]<= rs232_rx;  //锁存 9bit(1bit 起始位,8bit
数据)
                end
            else if(a) //数据移位处理
                begin
                    a<= 1'b0;
                    if(num<=4'd8) rx_data<=rx_data>>1'b1; //移位 8 次,第
1bit 起始位移除，剩下 8bit 正好时接收数据
                    else if(num==4'd12) num <= 4'd0;   //接收到 STOP 位后
结束,num 清零  end
                end
        end
end

assign rx_data_out=rx_data[3:0];
endmodule
```

(4)my_uart_tx.v

```
module my_uart_tx(clk_1M,rst,ad_up,Atx_data,Btx_data,rs232_tx);
input clk_1M;
input rst;
input ad_up;
input[7:0] Atx_data;
input[7:0] Btx_data;

output rs232_tx;

reg [4:0] num;
reg rs232_tx_r;
reg [1:0]state;
always @(posedge clk_1M or negedge rst)
begin
    if(!rst)
        begin
                num<=5'd0;
                rs232_tx_r<=1'b1;
                state<=1'b0;
            end
    else
```

```
begin
    case(state)
        1'b0:
            begin
               num<=1'b0;
                if(ad_up)
                    state<=1'b1;
                else
                    state<=1'b0;
            end
        1'b1:
            begin
                num<=num+1'b1;
                case(num)
                    5'd0:  rs232_tx_r<=1'b0;
                    5'd1:  rs232_tx_r<=1'b0;
                    5'd2:  rs232_tx_r<=Atx_data[1];
                    5'd3:  rs232_tx_r<=Atx_data[2];
                    5'd4:  rs232_tx_r<=Atx_data[3];
                    5'd5:  rs232_tx_r<=Atx_data[4];
                    5'd6:  rs232_tx_r<=Atx_data[5];
                    5'd7:  rs232_tx_r<=Atx_data[6];
                    5'd8:  rs232_tx_r<=Atx_data[7];

                    5'd9:  rs232_tx_r<=1'b0;
                    5'd10:    rs232_tx_r<=Btx_data[1];
                    5'd11:    rs232_tx_r<=Btx_data[2];
                    5'd12:    rs232_tx_r<=Btx_data[3];
                    5'd13:    rs232_tx_r<=Btx_data[4];
                    5'd14:    rs232_tx_r<=Btx_data[5];
                    5'd15:    rs232_tx_r<=Btx_data[6];
                    5'd16:    rs232_tx_r<=Btx_data[7];

                    5'd17:    rs232_tx_r<=1'b1;
                    5'd18: begin
                                rs232_tx_r<=1'b1;
                                state<=1'b0;
                            end
                    default :
                              begin
                                rs232_tx_r<=1'b1;
                                state<=1'b0;
                            end
```

```
                        endcase
                    end
                endcase
            end
end

assign rs232_tx=rs232_tx_r;
endmodule
```

附录四.实物图