

电子技术项目设计——串行通信接口

1. 整体实现思路

本实验主要的模块有

uart_rx：串行接收数据

uart_tx：串行发送数据

led：用FPGA上的LED显示数据信息

seg2：用FPGA上的晶体管显示数据信息

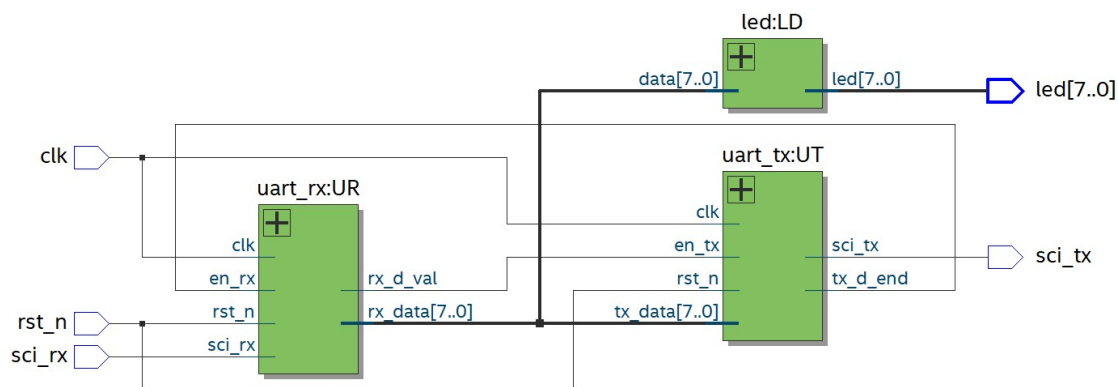
clk_1k：产生1kHz的时钟，给晶体管作为显示切换

alu：计算模块

data_ctl1：控制特定数据的接收和各单元的使能

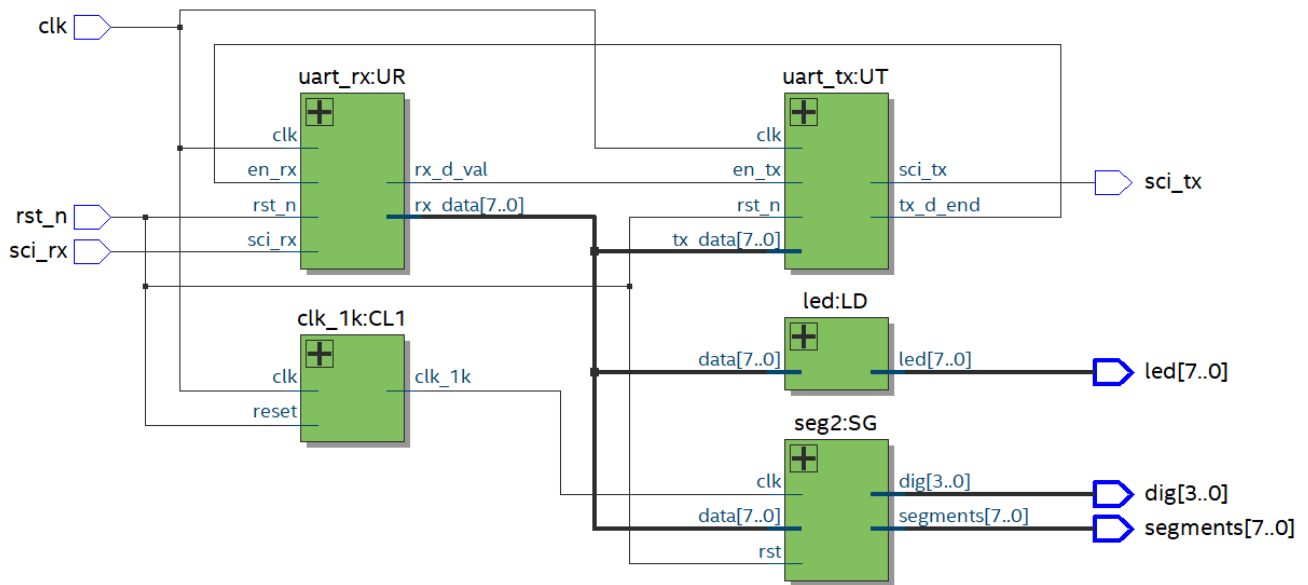
在任务4.1-2中，我们需要接收完数据后立即发送，于是可以设计出顶层模块的电路图为：

4.1



(无晶体管显示)

4.2

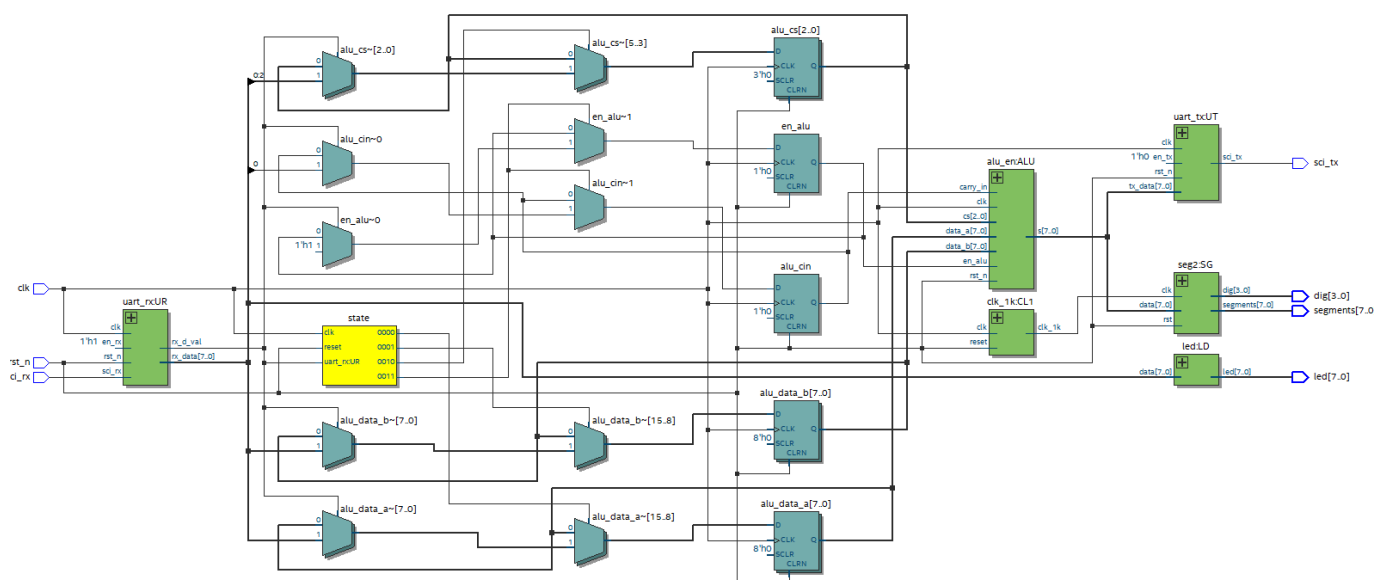


(有晶体管显示)

在任务4.3 中，我实现了FPGA单次接收4个数据 (data_a, data_b, cs, carry_in), 并将数据传入 ALU 模块中计算出结果。

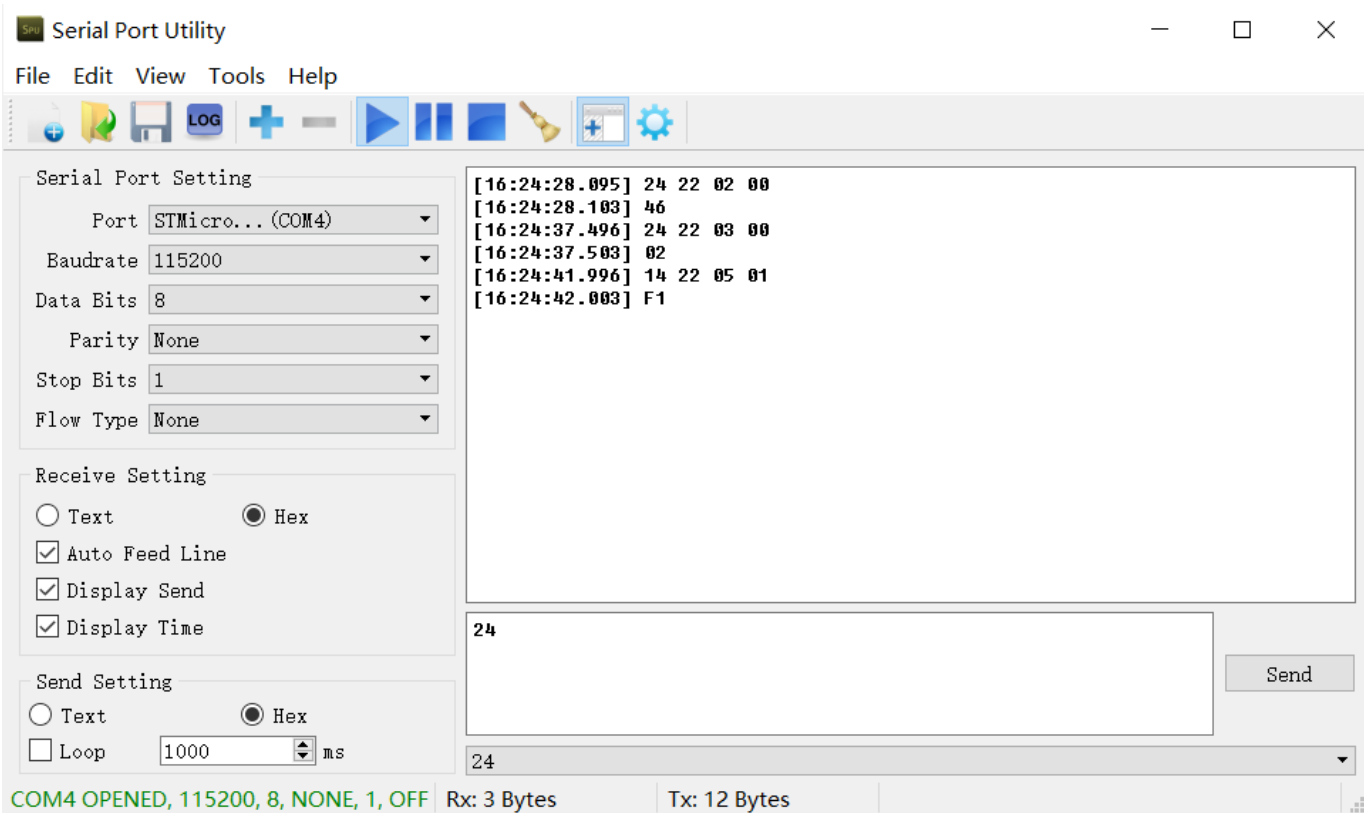
顶层的设计与4.1,2不一样，接收完数据后要按照一定次序存进操作数寄存器中，我们采用状态机来实现。

初代版本：



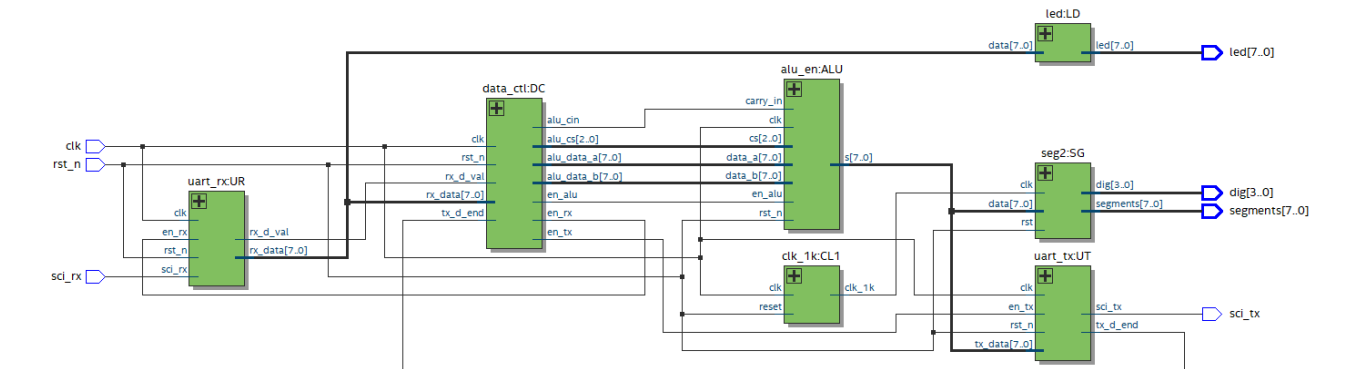
改良：注意到初代版本并未使用数据发送模块，并且顶层文件的没有封装完全。我对这两点进行改进，创新之处在于：

计算的结果通过fpga传输给stm,再传输给PC机，在串口上显示。



改良后的总体结构：

4.3



2.各模块实现

核心模块有四个：uart_tx , uart_rx , alu 和 data_ctl 。

```

uart_bcv.v
state = 2; //发送数据
state = 3; //发送完成

always @(posedge clk or posedge rst_n)
begin
    if (rst_n)
        begin
            tx_i_and <- 1;
        end
    else
        begin
            cnt <- 0;
            state <- 0;
            tx_i_and <- 0;
            tx_i_and <- 1;
        end
    end

    if (state == 0 && en_tx == 1) //允许发送
    begin
        state <- 1;
        tx_i_tx <- start_bit;
    end

    if (state == 1) //发送数据
    begin
        cnt <- cnt + 1;

        if (cnt == 8f)
            begin
                cnt <- 0;
                state <- 2;
                tx_i_and <- 0;
                tx_i_tx <- tx_data[tx_i_and];
            end
        else
            begin
                if (state == 2)
                begin
                    cnt <- cnt + 1;

                    if (cnt == 8f)
                        begin
                            cnt <- 0;
                            tx_i_tx <- tx_data[tx_i_and + 1];
                            tx_i_and <- tx_i_and + 1;
                        end
                    else
                        begin
                            if (tx_i_and == 4'hffff)
                                begin
                                    state <- 3;
                                    tx_i_tx <- end_bit;
                                    tx_i_and <- tx_i_and + 1;
                                end
                            else
                                begin
                                    cnt <- 0;
                                    tx_i_and <- tx_i_and + 1;
                                end
                            end
                        end
                    end
                end
            end
        end

        if (state == 3)
        begin
            cnt <- cnt + 1;

            if (cnt == 8f) //发送完成
            begin
                cnt <- 0;
                state <- 0;
                tx_i_and <- 1;
                tx_i_tx <- 1;
            end
        end
    end

endmodule

uart_alu.v
parameter AND = 3'b000,
OR = 3'b001,
XOR = 3'b010,
SUB = 3'b011,
SLT = 3'b100,
SLUC = 3'b101,
AOC = 3'b110;

always @(posedge clk or posedge rst_n)
begin
    if (rst_n)
        begin
            a <- 0;
            zero <- 1;
            carry_out <- 0;
        end
    else
        begin
            if (op_alu)
                begin
                    carry_out <- 0;

                    case (rx)
                        AND: begin
                            a <- data_a & data_b;
                        end
                        OR: begin
                            a <- data_a | data_b;
                        end
                        XOR: begin
                            a <- data_a ^ data_b;
                        end
                        SUB: begin
                            s <- data_a - data_b;
                        end
                        SLT: begin
                            a <- (data_a < data_b);
                        end
                        SLUC: begin
                            a <- (data_a - carry_in < data_b);
                        end
                        AOC: begin
                            carry_out <- 1;
                            a <- (carry_out, data_a) - data_b - carry_in;
                        end
                    end
                    zero <- (carry_out, a) < data_a & data_b & carry_in;
                end
            end
        end
    end

    // 设置 zero 位
    zero <- (s < 3'b00000000);

end

endmodule

uart_rcv.v
reg [15:0] cnt; //计数器
reg end_tx;

always @(posedge clk or posedge rst_n)
begin
    if (rst_n)
        begin
            rx_data <- 3'h00000000;
            rx_i_and <- 0;
            rx_i_and <- 0;
            state <- 0;
            cnt <- 0;
            end_tx <- 0;
        end
    else
        begin
            if (en_rx == 1 && state == 0) //接收数据
            begin
                rx_i_and <- 0;
                end_tx <- 0;

                if (cnt == start_bit)
                    begin
                        state <- 1;
                    end
                else
                    begin
                        if (state == 1) //接收数据位置
                        begin
                            cnt <- cnt + 1;

                            if (cnt == 8)
                                begin
                                    state <- 0;
                                    rx_i_and <- 0;
                                end
                            else
                                begin
                                    if (cnt == 8f) //接收数据结束
                                    begin
                                        if (cnt == 8f - 1)
                                            begin
                                                cnt <- 0;
                                                state <- 2;
                                                rx_i_and <- 0;
                                                rx_i_and <- 0; //接收数据
                                            end
                                        else
                                            begin
                                                if (state == 2) //接收数据结束
                                                begin
                                                    cnt <- cnt + 1;

                                                    if (cnt == 8f)
                                                        begin
                                                            cnt <- 0;
                                                            state <- 3;
                                                            rx_i_and <- rx_data[rx_i_and];
                                                            rx_i_and <- rx_i_and + 1;
                                                            // state <- 0;
                                                        end
                                                    else
                                                        begin
                                                            state <- 3;
                                                            rx_i_and <- rx_data[rx_i_and];
                                                            rx_i_and <- rx_i_and + 1;
                                                            // state <- 0;
                                                        end
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end

    if (state == 3) //接收数据结束
    begin
        cnt <- cnt + 1;

        if (cnt == 8f)
            begin
                end_tx <- 1;
                state <- 0;
            end
        end
    end

    if (cnt == 8f + tx_i_and) //接收数据结束
    begin
        cnt <- 0;
        state <- 0;
        rx_i_and <- end_tx;
    end
end

endmodule

data_ctl.v
output reg [7:0] alu_data_a;
output reg [7:0] alu_data_b;
output reg [15:0] alu_res;
output reg alu_tx;
output reg en_tx;
output reg en_rx;

reg [15:0] state;

always @(posedge clk or posedge rst_n)
begin
    if (rst_n)
        begin
            state <- 0;
            alu_data_a <- 0;
            alu_data_b <- 0;
            alu_tx <- 0;
            alu_res <- 0;
            en_tx <- 0; //允许发送
            en_rx <- 0; //允许接收
        end
    else
        begin
            case (state)
                0: begin
                    if (en_tx_val)
                        begin
                            alu_data_a <- rx_data;
                            state <- 1;
                        end
                    else
                        begin
                            1: begin
                                if (en_rx_val)
                                    begin
                                        alu_data_b <- rx_data;
                                        state <- 2;
                                    end
                                else
                                    begin
                                        2: begin
                                            if (en_tx_val)
                                                begin
                                                    alu_tx <- rx_data[15:8];
                                                    state <- 3;
                                                end
                                            else
                                                begin
                                                    3: begin
                                                        if (en_rx_val)
                                                            begin
                                                                alu_res <- rx_data[0];
                                                                en_tx <- 1;
                                                            end
                                                        else
                                                            begin
                                                                4: begin
                                                                    en_tx <- 0;
                                                                    en_rx <- 1;
                                                                    state <- 4;
                                                                    begin
                                                                        if (en_tx && tx_i_and)
                                                                            begin
                                                                                en_tx <- 1;
                                                                                en_rx <- 1;
                                                                                state <- 0;
                                                                            end
                                                                        end
                                                                    end
                                                                end
                                                            end
                                                        end
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

endmodule

```

4.1-2

[illegible]

4. 遇到的困难

在调试数据的发送时，我遇到了很多困难。首先是示波器没法显示发送波形，并且一直处于低电位，最后发现是**杜邦线内部断路**，更换杜邦线后解决了该问题；其次是发送的数据和接收到的数据不符合，在经过大量的仿真和代码检查时，我终于发现我的一处代码和我要实现的功能并不相同（注释处为未修改的代码）：

```
if (cnt == td)
begin
    cnt <= 0;
    sci_tx <= tx_data[tx_idx + 1];
    tx_idx <= tx_idx + 1;

    if (tx_idx == 4'b0)
    begin
        state <= 3;
        sci_tx <= end_bit;
        tx_idx <= 0;
    end
    // cnt <= 0;
    // tx_idx <= tx_idx + 1;

    // if (tx_idx == 4'b1000)
    // begin
    //     state <= 3;
    //     sci_tx <= end_bit;
    //     tx_idx <= 0;
    // end
    // else sci_tx <= tx_data[tx_idx];
end
```

我以为会发生的事：执行完 `tx_idx + 1` 后，`tx`线会显示 `tx_data[tx_idx + 1]`的值；

然而仿真结果告诉我：`tx`线实际显示的是`tx_data[tx_idx]`的值，这就会导致我的波形整体多了一位（第0位），导致数据传输出错。原因在于`begin-end`中的寄存器都是非阻塞赋值，也就是`tx_idx`和`sci_tx`会在同一时间赋值，`sci_tx`被赋值时还没有+1，所以应该使用`tx_data[tx_idx + 1]`。

除此之外，我还遇到了示波器使用和`reset`等方面的问题，在老师助教的帮助下和我仔细的检查下终于解决了问题。

总结

经过本次项目，我充分学习并应用了FPGA的语法和编程思维，并进行了多硬件的调试，锻炼了我发现问题、解决问题的能力。

本次项目的成果展示：

<https://cloud.tsinghua.edu.cn/f/827e955522dd4622bed9/>