

《数字逻辑与数字系统》实验报告

学院 智能与计算学部 年级 2023 级 班级 4

姓名 张宇晨 学号 3023210076

实验项目名称 UART 串口的设计与实现

大模型名称 GPT-5.2 问答形式 线上 本地

是否建立知识库 是 否

一. 实验目的

- 掌握基于 SystemVerilog HDL 的时序逻辑电路建模方法。
- 掌握串口的工作原理。
- 掌握计数器设计方法。
- 掌握移位寄存器设计方法。
- 掌握有限状态机的设计方法。

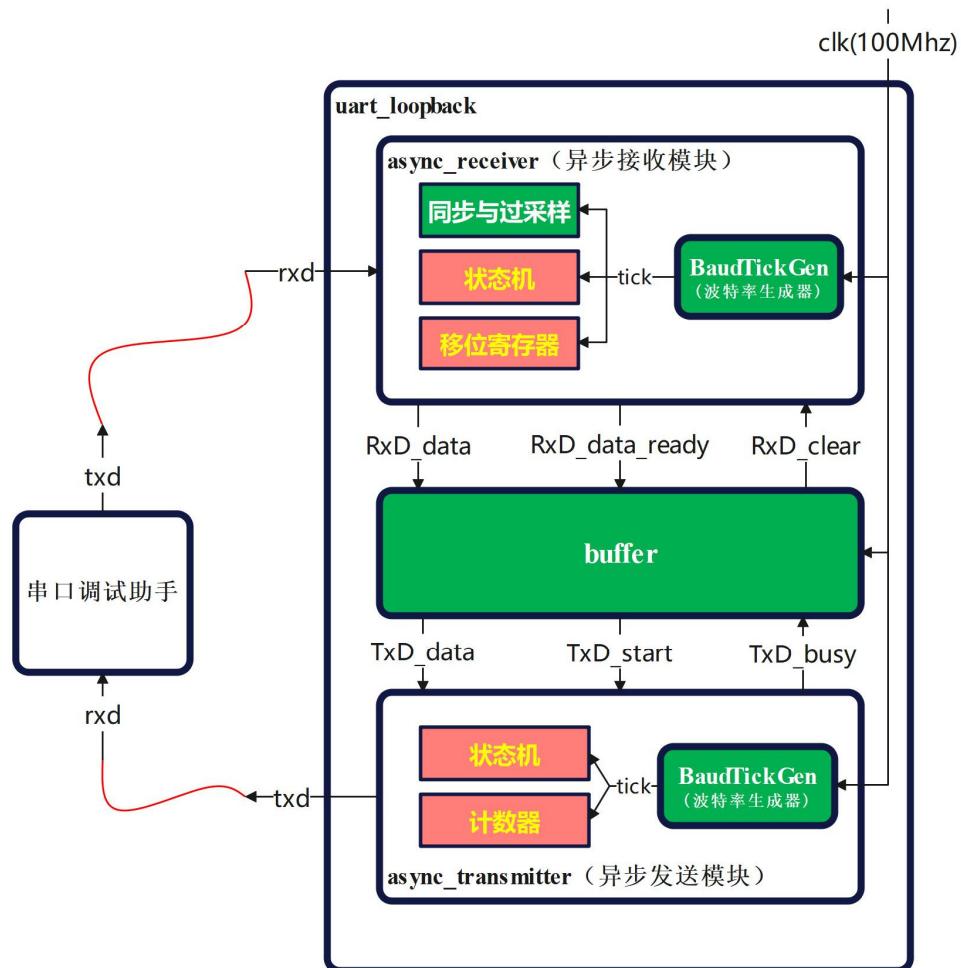
二. 实验内容

本实验的任务是设计并实现 UART 串口数据的接收模块和发送模块，最后通过回环测试（loopback 测试）。所谓回环测试就是发送端发送什么数据，接收端就接收什么数据，这也是常用的一种测试手段。如果回环测试成功，则说明从发送端到接收端之间的数据链路是正常的，以此来验证数据链路的畅通。

整个回环测试流程为：

- 异步接收模块从串口调试助手中接收串行数据，组装成并行数据后输送
到缓冲区模块。
- 缓冲区模块再将数据原封不动地输送到异步发送模块。
- 异步发送模块将来自缓冲区的并行数据拆分为串行数据后输送给串行
调试助手。

本实验采用代码补全形式完成，仅需补全下图中红色矩形（异步接收模块的状态机和移位寄存器、异步发送模块的状态机和计数器）的代码，需要补全的位置详见框架代码中的“TODO”注释信息。



三. 实验原理与步骤（请给出所有提示语和回答）

在使用大模型之前，已经画出了状态机，并实现了发送模块（省略框架代码）。

start-----

```

localparam D0 = 4'b1000, D1 = 4'b1001, D2 = 4'b1010, D3 = 4'b1011,
D4 = 4'b1100, D5 = 4'b1101, D6 = 4'b1110, D7 = 4'b1111,
STOP = 4'b0010;

...
always_ff @(posedge clk)

```

```

begin

    case(TxD_state)

        4'b0000: if(TxD_start) TxD_state <= 4'b0100;

        4'b0100: if (BitTick) TxD_state <= D0;      // START -> D0

        D0: if (BitTick) TxD_state <= D1;          // D0 -> D1

        D1: if (BitTick) TxD_state <= D2;          // D1 -> D2

        D2: if (BitTick) TxD_state <= D3;          // D2 -> D3

        D3: if (BitTick) TxD_state <= D4;          // D3 -> D4

        D4: if (BitTick) TxD_state <= D5;          // D4 -> D5

        D5: if (BitTick) TxD_state <= D6;          // D5 -> D6

        D6: if (BitTick) TxD_state <= D7;          // D6 -> D7

        D7: if (BitTick) TxD_state <= STOP;         // D7 -> STOP

        STOP: if (BitTick) TxD_state <= 4'b0000;   // STOP -> IDLE

        default: if(BitTick) TxD_state <= 4'b0000;

    endcase

end

logic [3:0] bit_cnt;

always_ff @(posedge clk) begin

    if (TxD_ready && TxD_start) begin

        bit_cnt <= 3'd0;

    end

    else if (BitTick) begin

        if (TxD_state >= D0 && TxD_state <= D7) bit_cnt <= bit_cnt + 3'd1;

    end

end

always_comb begin

    if (TxD_state == 4'b0000) TxD = 1'b1;

```

```

else if (TxD_state == 4'b0100) TxD = 1'b0;

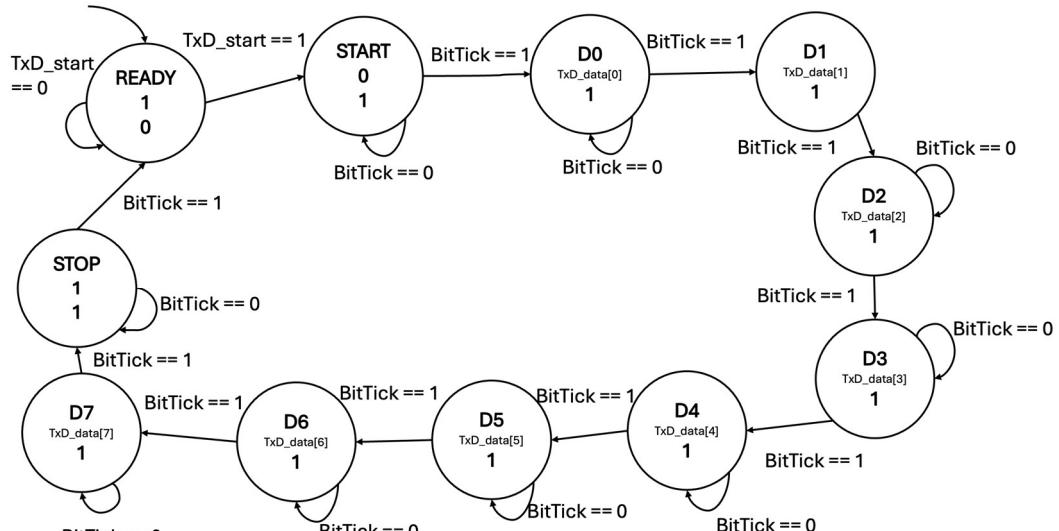
else if (TxD_state == STOP) TxD = 1'b1;

else TxD = TxD_data[bit_cnt];

end

end-----

```



第二行为 TxD 输出, 第三行为 TxD_busy

下面是具体的问答过程。

User: 实现 UART 的接收模块, 根据 tb_async_receiver.sv 文件的测试, 修改 async_receiver.sv, 使之通过测试。

注意:

1. 在实现状态机过程中, 必须使用同步、滤波后的 RxD_bit 信号和过采样得到的最佳采样点 sampleNow 来更新状态机。
2. 接收模块的核心功能之一是将串行数据组装为并行数据, 这可以通过移位寄存器实现。

3. 不要修改文件编码，不要增加任何注释，直接在原文件增加代码即可。

GitHub Copilot:

```
start-----  
...  
  
always_ff @(posedge clk)  
  
case(RxD_state)  
  
 4'b0000: if(~RxD_bit) RxD_state <= `ifdef SIMULATION 4'b1000  
`else 4'b0001 `endif;  
  
 4'b0001: if(sampleNow) RxD_state <= 4'b1000;  
  
 4'b1000: if(sampleNow) RxD_state <= 4'b1001;  
  
 4'b1001: if(sampleNow) RxD_state <= 4'b1010;  
  
 4'b1010: if(sampleNow) RxD_state <= 4'b1011;  
  
 4'b1011: if(sampleNow) RxD_state <= 4'b1100;  
  
 4'b1100: if(sampleNow) RxD_state <= 4'b1101;  
  
 4'b1101: if(sampleNow) RxD_state <= 4'b1110;  
  
 4'b1110: if(sampleNow) RxD_state <= 4'b1111;  
  
 4'b1111: if(sampleNow) RxD_state <= 4'b0010;  
  
 4'b0010: if(sampleNow) RxD_state <= 4'b0000;  
  
 default: if(sampleNow) RxD_state <= 4'b0000;  
  
endcase  
  
  
always_ff @(posedge clk)  
begin  
  
  if(RxD_state==4'b0000 && ~RxD_bit)  
    RxD_data <= 8'h00;  
  
  else if(sampleNow)  
    begin
```

```

if(RxD_state>=4'b1000 && RxD_state<=4'b1111)

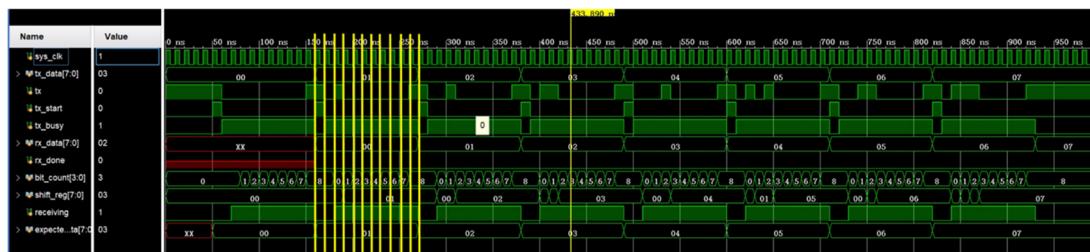
    RxD_data <= {RxD_bit, RxD_data[7:1]};

end

end
end-----

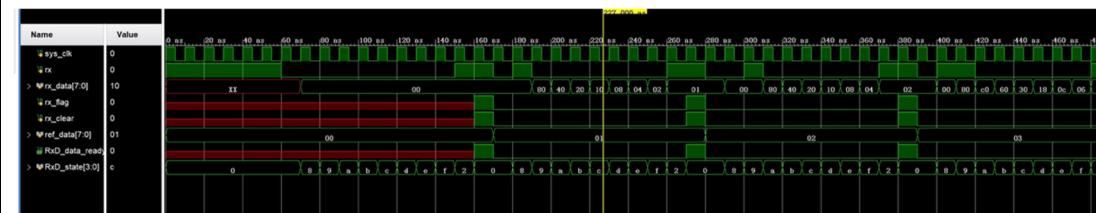
```

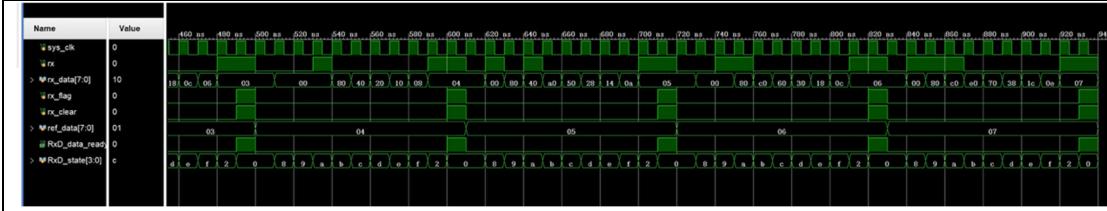
四. 仿真与实验结果（注：仿真需要给出波形图截图，截图要清晰，如果波形过长，可以分段截取；实验结果为远程 FPGA 硬件云平台的截图）



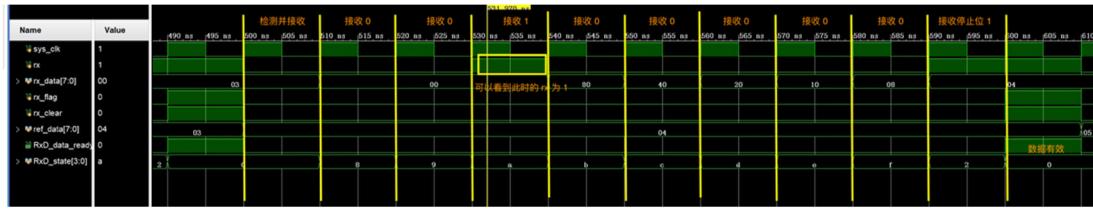
上图为发送模块，选择发送 0x01 数据的那一段进行分析。每两个黄线之间表示一个状态，分别为：

1. 起始阶段：分别为 IDLE/READY 和 START，此时 receiving 均为 1。 tx 分别为 1 和 0。
2. 数据阶段：分别为 D0 到 D7，此时 bit_count 表示发送数据的位数。可以看到只有第 0 位的数据 tx 为 1，其余均为 0。
3. 停止阶段：STOP 状态。此时 tx 为 1。





上面两张是接收模块的仿真结果。



上图是接收 0x04 数据的结果。图中对整个过程进行了分析。



上图是 FPGA 平台串口通信结果。

五. 请评价大模型在本实验中的表现（例如：优势、劣势、使用建议等。）

使用大模型之前最好要明白整个流程，尽可能地提供详细的提示词，这样它的输出才会更加符合要求，否则它大概率会选择一种不符合要求的方案。

六. 附加题

1. 在接收模块中，为什么 RxD 信号要同步和过滤？为什么要进行过采样？

在串口调试助手中，输入的 RxD 信号与接收端时钟 clk 处于异步时钟域关系。为实现 RxD 信号从慢速时钟域（串口调试助手）到快速时钟域（接收端）的可靠传输，需通过两位移位寄存器进行同步处理，通常称为“打两拍”。

过采样在工程中的作用是抗干扰：一方面，起始位的检测是接收过程的起点，其准确性至关重要，传输线路上的毛刺或噪声可能产生一个短暂的负脉冲，若只采样一次容易被误判为起始位。另一方面，在数据位中心进行采样是可靠的，这是因为一个位的电平在位的中心区域是最稳定的，边沿区域则因信号跳变而充满不确定性。

虽然进行了过采样，但仍不能直接选取某一个采样点的值来作为最终采样值，这是因为信号在传输过程中可能受到噪声干扰，导致个别采样点出现错误。即使直接选择中点采样，一旦这个点恰好受到噪声干扰，就会导致整个数据位采样错误。为了抵抗这种瞬时噪声，提高系统的鲁棒性，我们需要进行滤波。

2. 在发送模块中，输出 TxD 信号是基于计数器实现的，是否可以改为用移位寄存器来实现？在接收模块中，输出 RxD_data 信号是基于移位寄存器实现的，是否可以改为用计数器实现吗？请给出实现关键代码。

可以。

用移位寄存器实现发送模块：

```
start-----  
logic [9:0] TxD_shift;  
  
always_ff @(posedge clk) begin  
    case (TxD_state)
```

```

4'b0000: begin

    if (TxD_start) begin

        // LSB 先发: 把 start(0) 放到 bit0, 这样 TxD_shift[0]先输出

        TxD_shift <= {1'b1, TxD_data, 1'b0}; // {stop, data[7:0], start}

        TxD_state <= 4'b0100;

    end

end


4'b0100: if (BitTick) TxD_state <= D0;

// 每个数据位状态到来时, 在 BitTick 处移位一次

D0, D1, D2, D3, D4, D5, D6, D7: begin

    if (BitTick) begin

        TxD_shift <= {1'b1, TxD_shift[9:1]}; // 右移, MSB 补 1

        TxD_state <= TxD_state + 4'd1;          // D0->D1->...->D7

    end

end


STOP: if (BitTick) TxD_state <= 4'b0000;

default: if (BitTick) TxD_state <= 4'b0000;

endcase

end


// TxD 输出逻辑: 完全由移位寄存器最低位决定 (无需 bit_cnt)

always_comb begin

    if (TxD_state == 4'b0000) TxD = 1'b1;      // 空闲

    else TxD = TxD_shift[0]; // 其余状态输出当前最低位

end

end-----

```

用计数器实现接收模块：

```
start-----  
  
logic [2:0] rx_cnt;  
  
  
always_ff @(posedge clk) begin  
    // 检测到起始位：清零数据和计数器  
    if (RxD_state==4'b0000 && ~RxD_bit) begin  
        RxD_data <= 8'h00;  
        rx_cnt <= 3'd0;  
    end  
    else if (sampleNow) begin  
        if (RxD_state >= 4'b1000 && RxD_state <= 4'b1111) begin  
            // UART 是 LSB 先到：第 1 个数据位写 RxD_data[0]  
            RxD_data[rx_cnt] <= RxD_bit;  
            rx_cnt <= rx_cnt + 3'd1;  
        end  
  
        // 进入停止位时，下一帧重新从 0 计  
        if (RxD_state == 4'b0010) begin  
            rx_cnt <= 3'd0;  
        end  
    end  
end-----
```