

本科实验报告

课程名称： **FPGA应用**

主讲教师： 夏书峰

学院、系： 微电子学院

专 业： 电子科学与技术类

班 级： 微2005

学 号：

姓 名：

联系电话：

电子邮箱：

2021～2022学年第3学期

实验成绩记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 内容 | 学时 | 满分 | 得分 |
| 1 | Vivado/ModelSim实践 | 4 | 15 |  |
| 2 | 时序逻辑设计1 — RTC | 4 | 20 |  |
| 3 | Xilinx嵌入式逻辑分析仪 | 4 | 15 |  |
| 4 | Xilinx嵌入式处理器实践 | 4 | 20 |  |
| 5 | 时序逻辑设计2 — FSM | 8 | 30 |  |
|  |  |  |  |  |
| 总计 | | 24 | 100 |  |

主讲教师签字：

实验五《状态机逻辑设计》实验报告

【实验内容】

用状态机实现UART-Lite逻辑的设计，编写测试向量进行仿真；再设计一个控制UART-Lite收发数据的状态机，接收计算机发来的固定数目字节，计算累加和并返回给计算机，编写测试向量进行仿真。最后在PYNQ-Z2板上验证设计，用计算机上的串口调试软件与PL部分主控状态机通过UART-Lite双向通信

【设计文件与实验过程】

项目主要分为两个部分：控制状态机和UART-Lite外设。

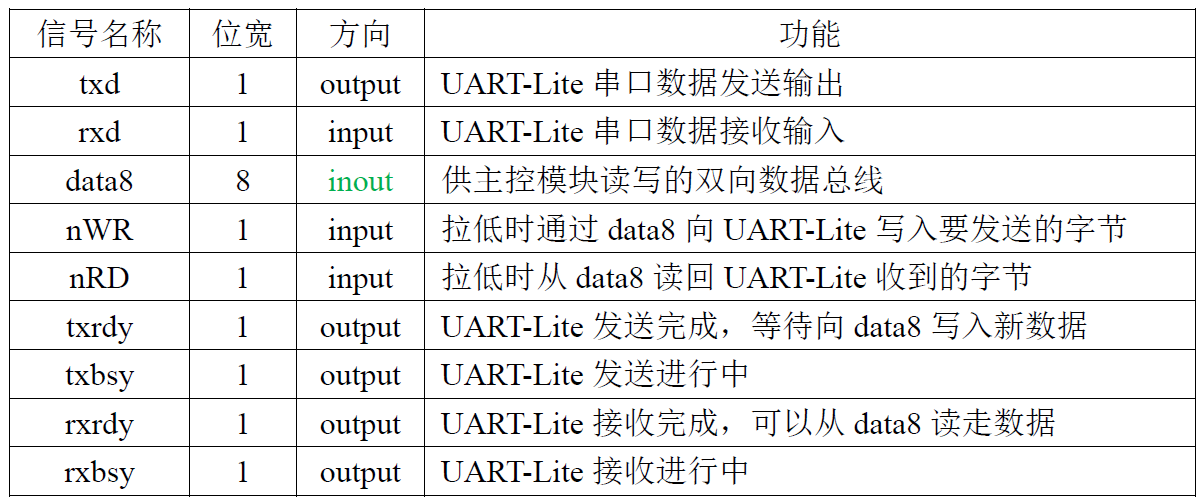
控制状态机源文件为【control.v】

UART-Lite的RTL实现源文件包括【uart\_clk\_gen.v】、【uart\_rx.v】、【uart\_tx.v】、【uart\_top.v】

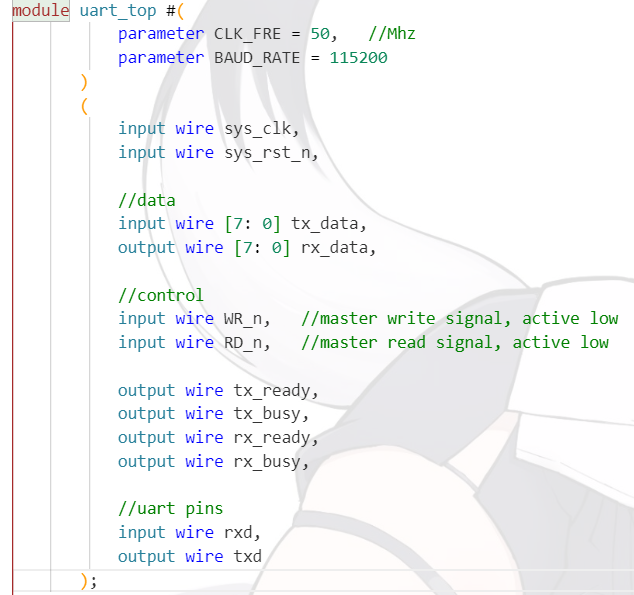
两套RTL统一由顶层文件【topFrame.v】例化，按照项目要求实现PS-UART-FSM系统。

控制状态机部分设置了五个状态，分别管理串口接收-运算-串口发送，从而实现了要求的接收固定数目字节-计算累加和-返回计算机的任务。

UART-Lite外设实现了基本的串口收发功能，并按照要求引出了部分接口



其中data8改成了更易于实现的output rx\_data和input tx\_data分别表示串口接收端发送给控制状态机的8位数据和控制状态机发送给串口发送端的8位数据。具体信号如下所示：



下面逐文件进行介绍

【uart\_clk\_gen.v】、【uart\_rx.v】、【uart\_tx.v】、【uart\_top.v】

四个文件分别为：UART-Lite模块的采样、发送时钟生成；UART接收模块；UART发送模块；UART顶层模块。

【uart\_clk\_gen.v】代码如下所示：

module uart\_clk\_gen#(

    parameter CLK\_FRE = 50, //clock frequency(MHz)

    parameter BAUD\_RATE = 115200 //serial baud rate

)(

    input wire sys\_clk,

    input wire sys\_rst\_n,

    output wire uart\_rx\_clk,

    output wire uart\_tx\_clk

);

localparam RX\_CLK\_SCALER = 16;

localparam RX\_CLK\_COUNT\_MAX = (CLK\_FRE\* 1\_000\_000) / (BAUD\_RATE \* RX\_CLK\_SCALER); //uart oversampling clk

localparam TX\_CLK\_COUNT\_MAX = (CLK\_FRE\* 1\_000\_000) / BAUD\_RATE; //base baud rate clk

localparam RX\_CNT\_WIDTH = $clog2(RX\_CLK\_COUNT\_MAX);

localparam TX\_CNT\_WIDTH = $clog2(TX\_CLK\_COUNT\_MAX);

reg [RX\_CNT\_WIDTH - 1:0] rx\_cnt = 0;

reg [TX\_CNT\_WIDTH - 1:0] tx\_cnt = 0;

assign uart\_rx\_clk = (rx\_cnt == 'd0);

assign uart\_tx\_clk = (tx\_cnt == 'd0);

always @(posedge sys\_clk) begin

    if(!sys\_rst\_n) begin

        rx\_cnt <= {RX\_CNT\_WIDTH{1'b0}};

    end

    if (rx\_cnt == RX\_CLK\_COUNT\_MAX[RX\_CNT\_WIDTH - 1:0]) begin

        rx\_cnt <= {RX\_CNT\_WIDTH{1'b0}};

    end

    else begin

        rx\_cnt <= rx\_cnt + 1;

    end

end

always @(posedge sys\_clk) begin

    if(!sys\_rst\_n) begin

        tx\_cnt <= {TX\_CNT\_WIDTH{1'b0}};

    end

    if (tx\_cnt == TX\_CLK\_COUNT\_MAX[TX\_CNT\_WIDTH - 1:0]) begin

        tx\_cnt <= {TX\_CNT\_WIDTH{1'b0}};

    end

    else begin

        tx\_cnt <= tx\_cnt + 1;

    end

end

endmodule

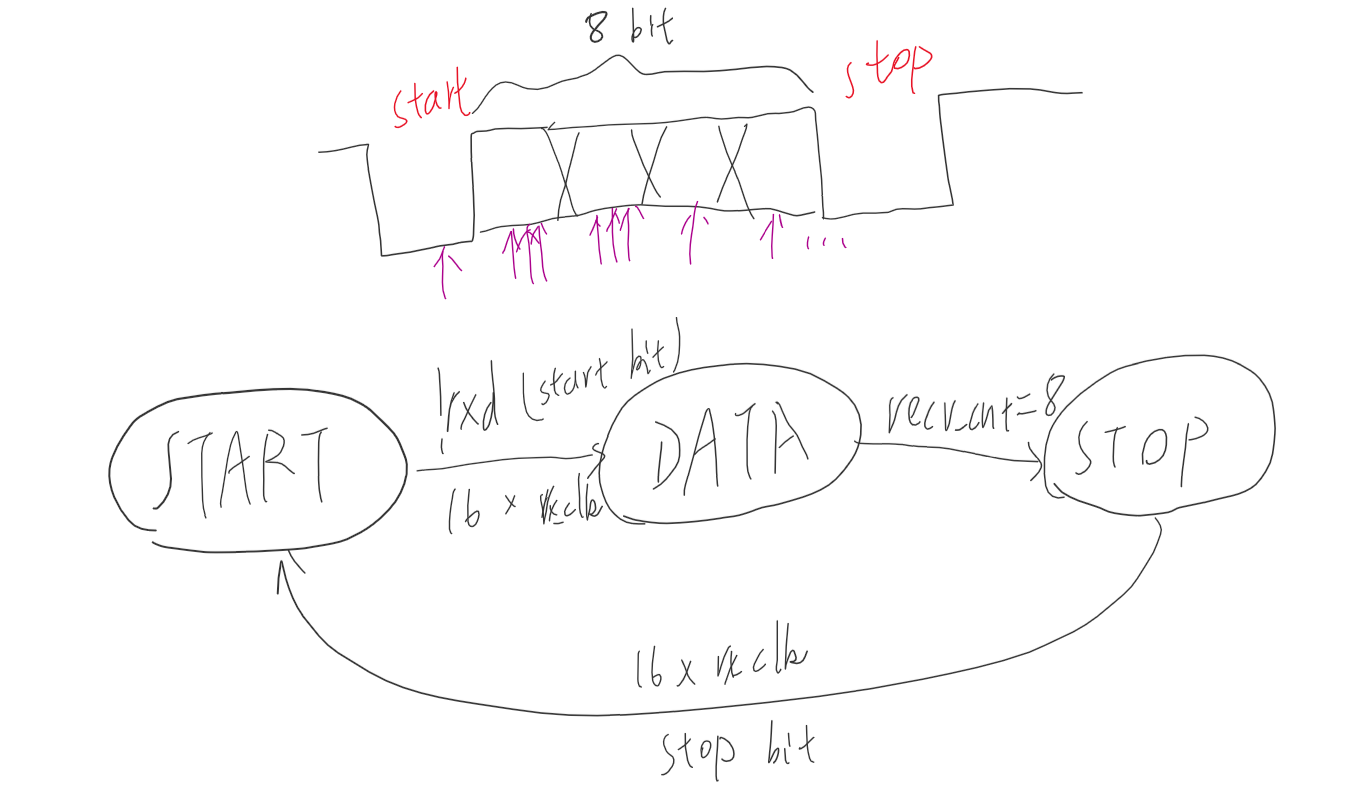
其中建立了两套时钟，分别是发送时钟和采样时钟，发送时钟与所需波特率对应时钟同频率，但采样时钟是其16倍频，也就是说一个发送时钟周期内包含16个采样时钟周期，目的是让接收模块可以进行过采样。

该模块使用两套计数器搭建。

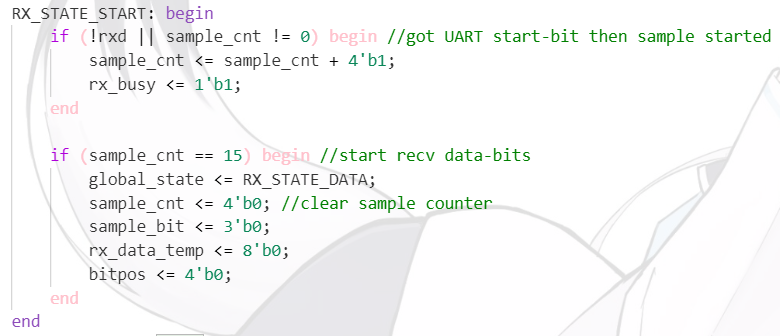
【uart\_rx.v】

该模块用于接收串口数据。状态机由三个状态构成，分别是用于等待和起始位识别的RX\_STATE\_START状态、过采样接收数据位的RX\_STATE\_DATA状态和用于识别停止位并将串口设备复位的RX\_STATE\_STOP状态

状态转移图如下所示

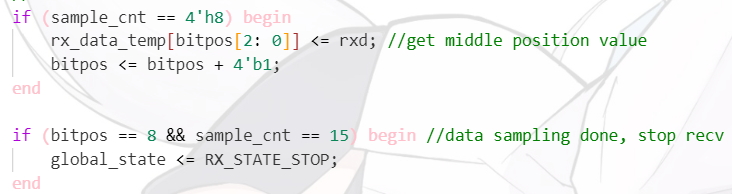


start状态代码如下：



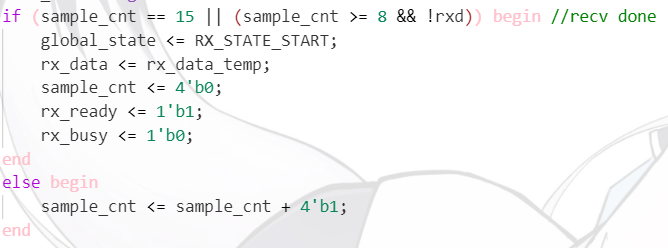
使用了sample\_cnt计数器处理起始位的延时计数，当rxd拉低后16个采样周期，判定开始接收数据位。

data状态主要代码如下所示：



每个时钟周期的中位判断一次数据是高电平还是低电平，并将其保存到临时数据寄存器中rx\_data\_temp中，用一个计数器bitpos来确定当前接收的是哪一位。

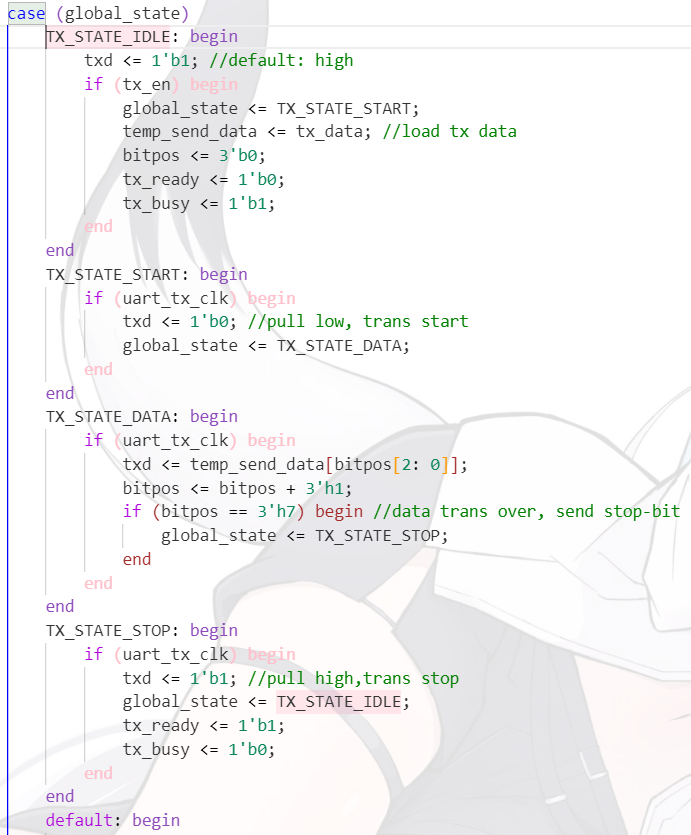
stop状态主要代码如下：



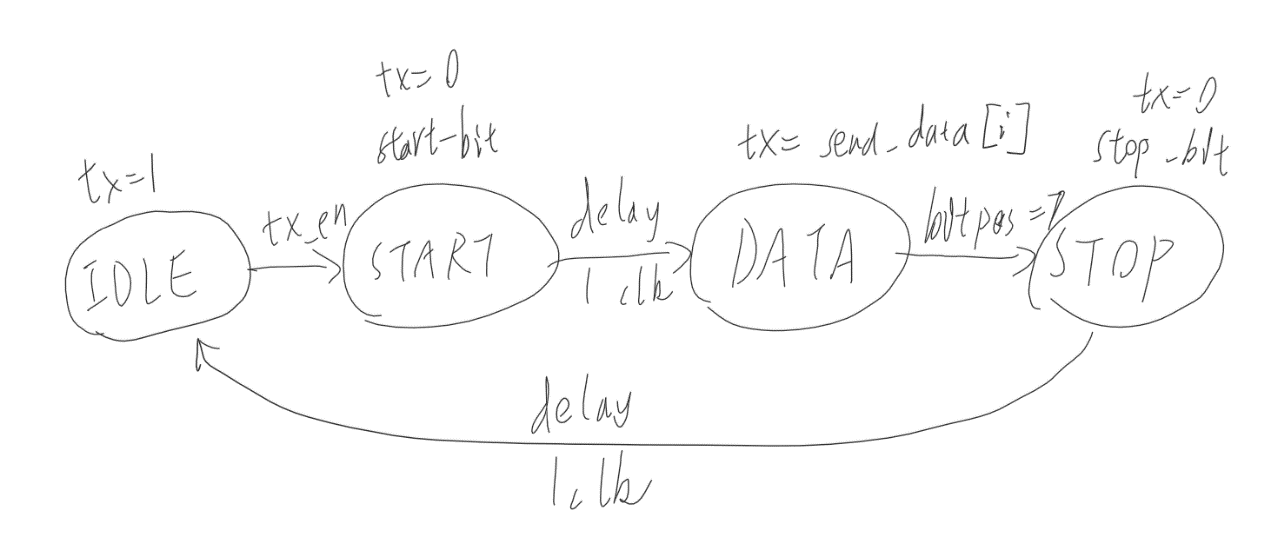
这部分代码与start部分代码类似，都是在计时16个采样周期后判定结束，转换状态

【uart\_tx.v】

该部分代码较简单，直接根据协议手册确定状态机，主要代码如下：



主要分成四个状态，状态转移图如下：



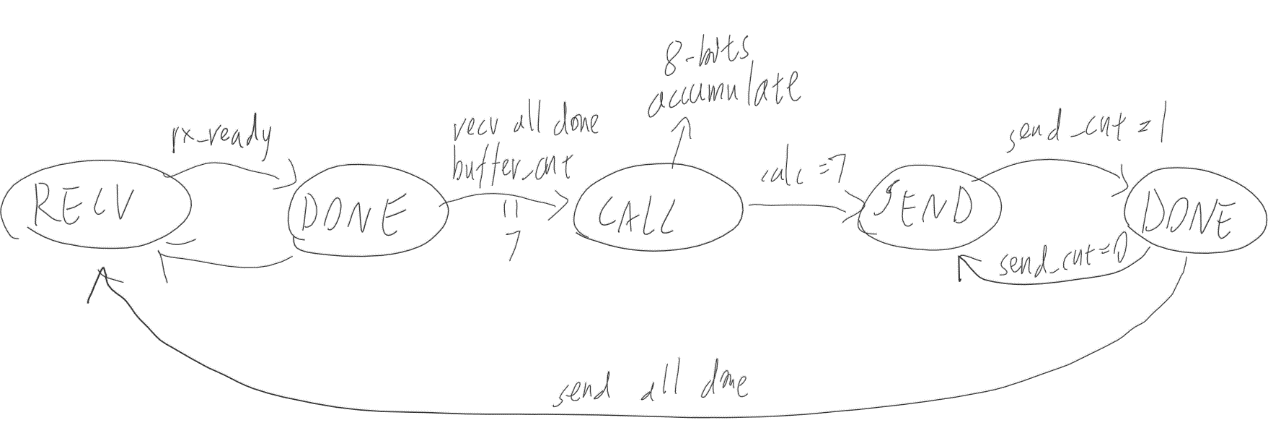
其中idle状态等待发送指令，当tx\_en指令位拉高时，表示串口发送开始，状态转换到start，发送起始位拉低txd，延迟一个发送周期后进入data状态，随后根据计数器bitpos依次发送0到7位，当所有位发送完毕后，转换成stop状态发送停止位，循环往复。

接收发送模块的代码均使用一段式状态机，编写快速简单，适合具有简单逻辑的串口和收发结构实现，但依赖于综合器优化，使用资源可能较多。

三个模块统一被【uart\_top.v】例化，组合成UART-Lite IP。

【control.v】

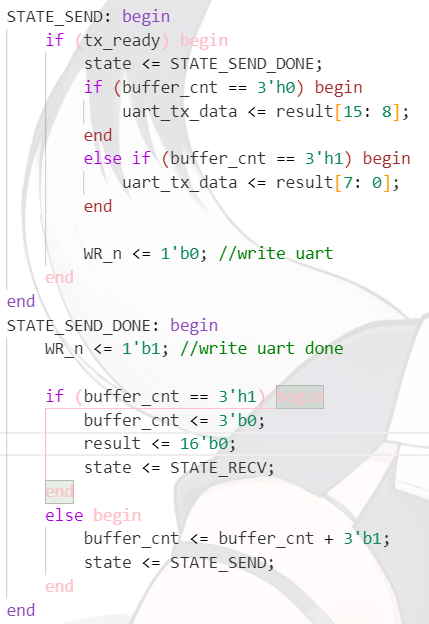
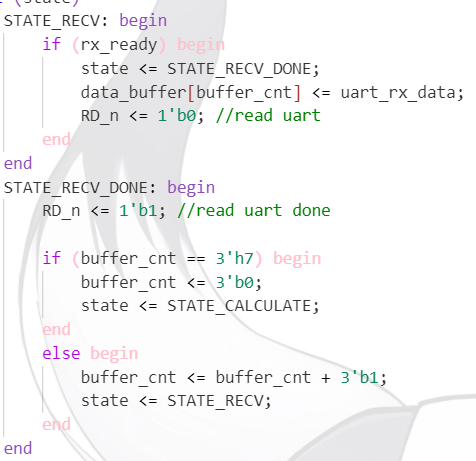
该模块实现了发送-接收控制状态机，状态转移图如下所示：



模块默认位于接收状态，当有输入时，状态机在recv和recv\_done状态之间切换来接收每一字节数据，接收完预定的8字节数据后，进入calculate状态进行累加，累加完毕时保存得到的累加和到缓存，并切换到发送状态，状态机在send和send\_done之间切换来发送每一字节数据。

使用该逻辑不仅可以进行数据的累加，还可以进行CRC校验等复杂操作，只需要在中间的calculate状态引入不同的计算模块即可。但由于时间限制，并未能完成扩展任务，仅进行了主要模块的验证。

关键的发送和接收逻辑代码如下所示：



【结果分析】

在本次实验中，完成了必要的UART-Lite设计和主控模块设计，能够进行收取上位机下传的8字节数据累加并给出16位（2字节）的回传数据，基本实现了要求。

上板演示如下（演示视频）所示：

