SPI协议_Verilog实现

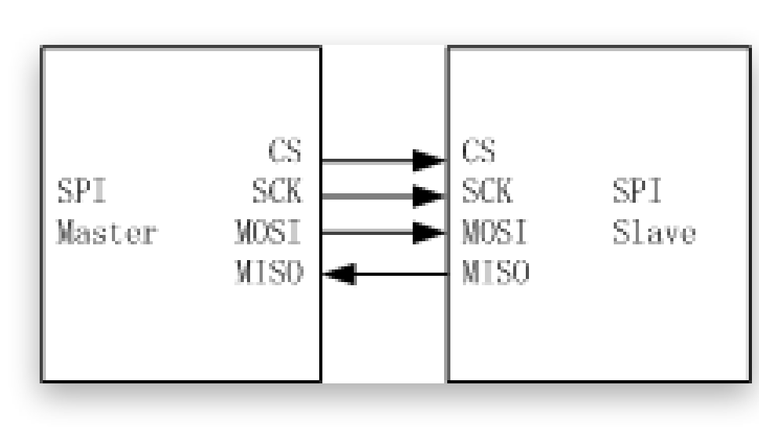
2020-06-03

spi verilog

SPI协议_Verilog实现

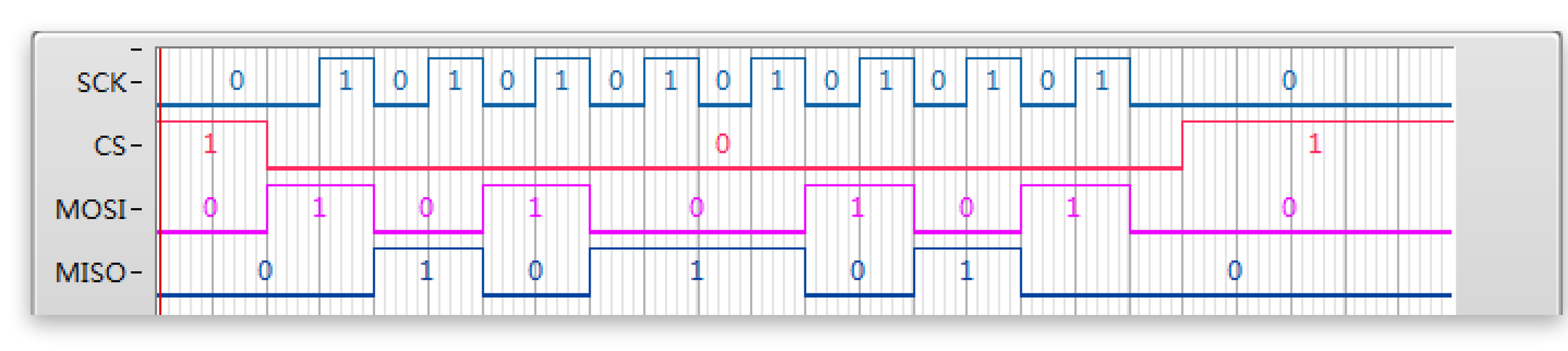
概述：
通过Verilog代码+仿真的形式来理解SPI的时序，此处只写了主机发送，从机接收的代码，后待续。。。
SPI协议简介
●SPI接口介绍
SCK：时钟信号，由主设备产生，所以主设备SCK信号为输出模式，从设备的SCK信号为输入模式。
CS：使能信号，由主设备控制从设备，，所以主设备CS信号为输出模式，从设备的CS信号为输入模式。
MOSI：主设备数据输出，从设备数据输入，所以主设备MOSI信号为输出模式，从设备的MOSI信号为输入模式。
MISO：主设备数据输入，从设备数据输出，所以主设备MISO信号为输入模式，从设备的MISO信号为输出模式。

●SPI接口连接图

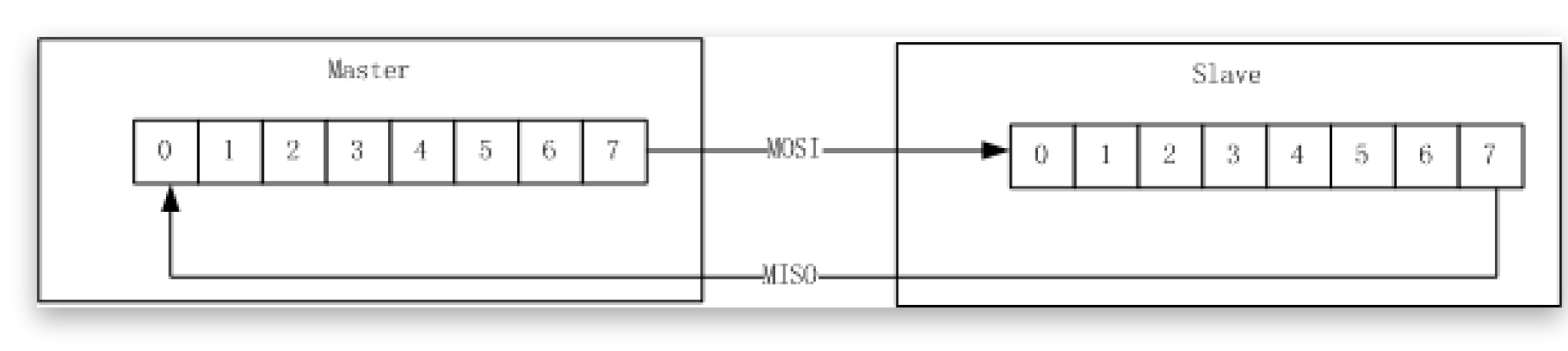


注意：MOSI和MISO不能交叉连接(可以把主从机理解为一个整体系统，MOSI为系统主机发送从机接收的数据线，MISO为主机接收从机发送的数据线)。

●SPI数据传输方向
SPI作为全双工的的串行通信协议，数据传输时高位在前，低位在后。主机和从机公用由主机产生的SCK信号，所以在每个时钟周期内主机和从机有1bit的数据交换(因为MOSI和MISO数据线上的数据都是在时钟的边沿处被采样)。
如下图：

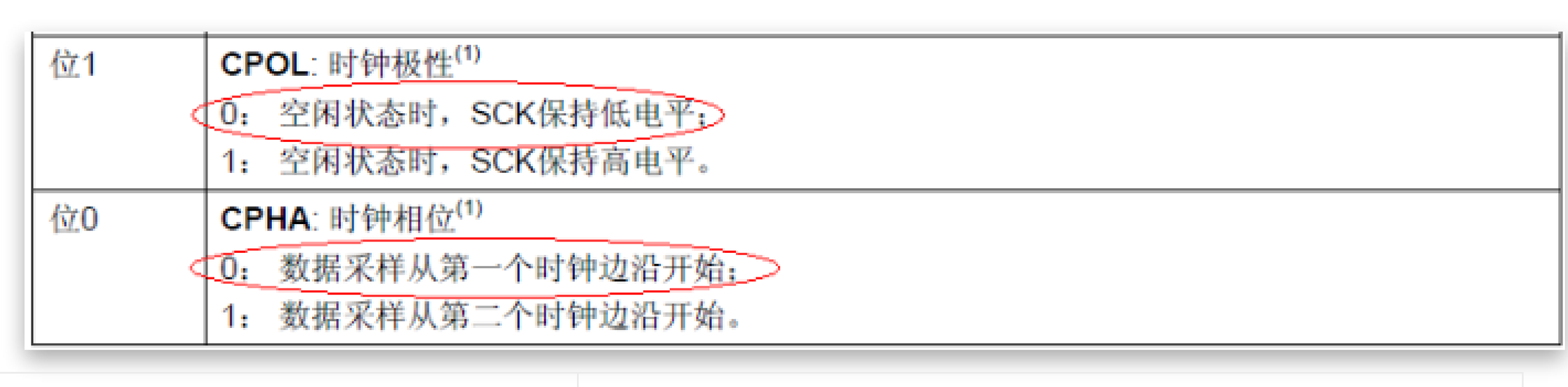


SPI协议规定数据采样是在SCK的上升沿或下降沿时刻(由SPI模式决定，下面会说到)，观察上图，在SCK的边沿处，主机会在MISO处采样(接收来从机的数据)，从机会在MOSI处采样(接收来自主机的数据)，所以每个时钟周期中会有一bit的数据交换。



●SPI传输模式

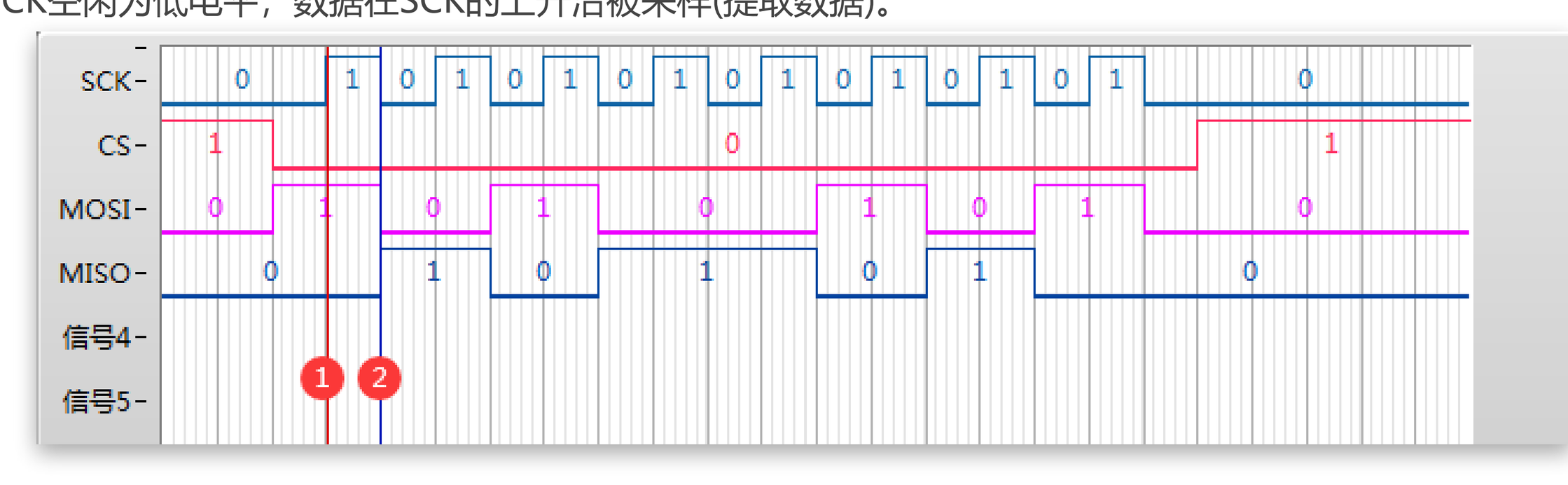
SPI总线传输一共有 4 种模式，这 4 种模式分别由时钟极性(CPOL)和时钟相位(CPHA)来定义。



位1	CPOL: 时钟极性[(1)] 0：空闲状态时，SCK保持低电平； 1：空闲状态时，SCK保持高电平。
位0	CPHA: 时钟相位[(1)] 0：数据采样从第一个时钟边沿开始； 1：数据采样从第二个时钟边沿开始。

CPOL	CPHA
规定了SCK时钟信号空闲状态的电平	规定了数据是在SCK时钟的上升沿还是下降沿被采样
-----------	------------------------------------
模式0：CPOL=0，CPHA =0	SCK空闲为低电平，数据在SCK的上升沿被采样(提取数据)
模式1：CPOL=0，CPHA =1	SCK空闲为低电平，数据在SCK的下降沿被采样(提取数据)
模式2：CPOL=1，CPHA =0	SCK空闲为高电平，数据在SCK的下降沿被采样(提取数据)
模式3：CPOL=1，CPHA =1	SCK空闲为高电平，数据在SCK的上升沿被采样(提取数据)

以模式0为例：
SCK空闲为低电平，数据在SCK的上升沿被采样(提取数据)。



?在时钟的第1个上升沿(游标1处)(采样点)
MOSI上数据为1，则在此边沿从机采样(提取)数据为1，采样点在MOSI数据线的中间(游标1处)。
MISO上数据为0，则在此边沿主机采样(提取)数据为0，采样点在MISO数据线的中间(游标1处)。
?在时钟的第1个下降沿(游标2处)(切换点)
MOSI上数据由1切换为0，数据在时钟下降沿时切换数据。
MISO上数据由0切换为1，数据在时钟下降沿时切换数据。
?在时钟的第2~8个上升沿(采样点)，主机在MISO上采样数据，从机在MOSI上采样数据。
?在时钟的第2~8个下降沿(切换点)，主机在MISO上切换数据，从机在MOSI上切换数据。

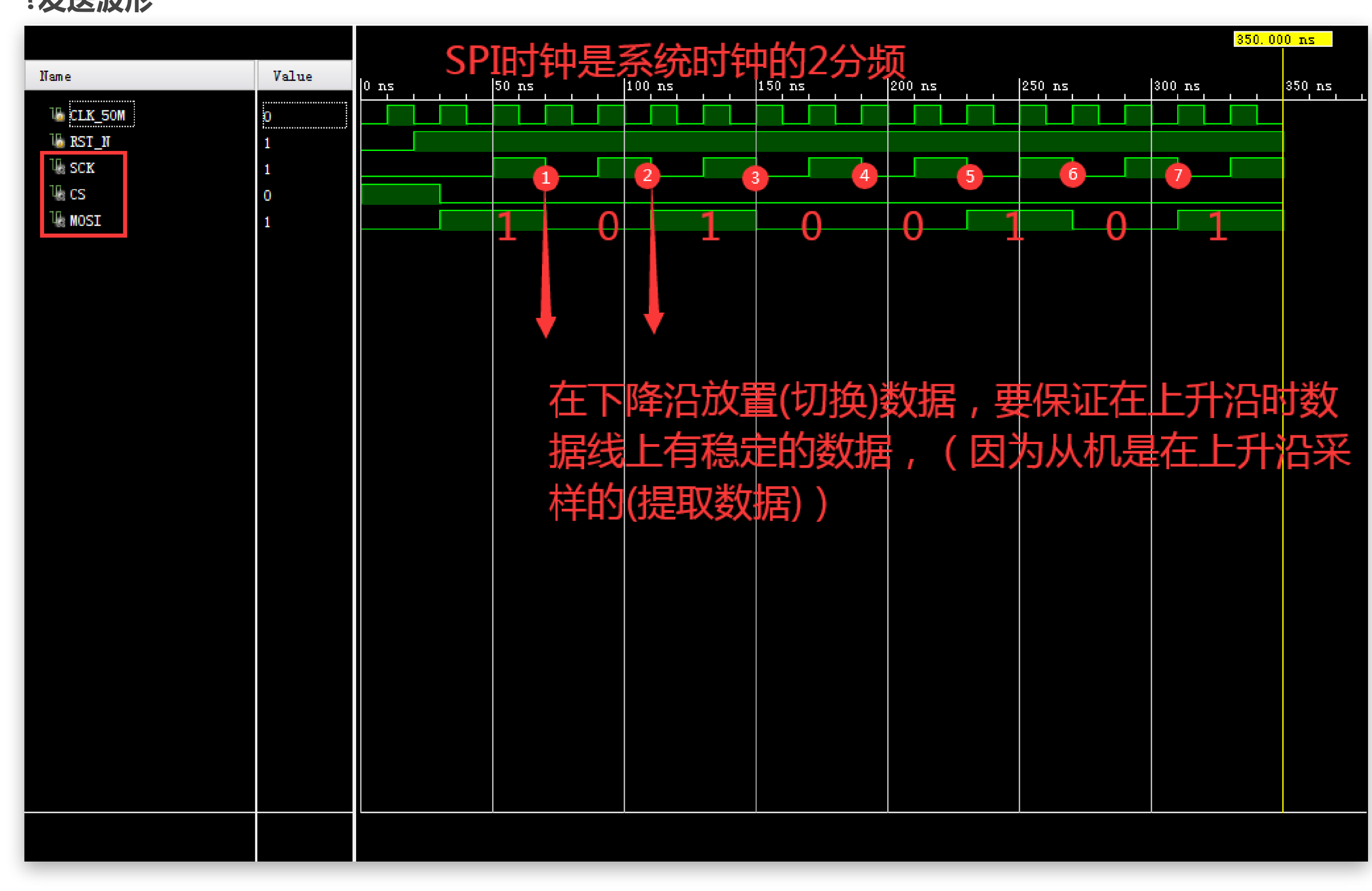
SPI_verilog实现

?主机发送代码

```
//采用SPI模式0：上升沿采样数据，下降沿切换数据
module SPI_MasterToSlave(CLK_50M,RST_N,SCK,CS,MOSI);
    input CLK_50M;
    input RST_N;
    output reg SCK;
    output reg CS;
    output reg MOSI;
reg[7:0] Send_Data = 8'hA5;//所要发送的数据
/*构造状态机*/
reg[3:0] Data_State = 4'd0;
parameter D7_State = 4'd0;//发送最高位数据-状态
parameter D6_State = 4'd2;
parameter D5_State = 4'd4;
parameter D4_State = 4'd6;
parameter D3_State = 4'd8;
parameter D2_State = 4'd10;
parameter D1_State = 4'd12;
parameter D0_State = 4'd14;//发送最低位数据-状态
always@(posedge CLK_50M or negedge RST_N)
begin
    if(RST_N == 0)//复位
    begin
        SCK <= 1'b0;    //SCK初始电平为低
        CS <= 1'b1;     //CS初始电平为高
        MOSI <= 1'b0;   //MOSI初始电平为低
    end
    else//产生SPI时序
    begin
       CS <= 0;//CS拉低准备数据传输
       case(Data_State)
       4'd1,4'd3,4'd5,4'd7,4'd9,4'd11,4'd13,4'd15://每次放置数据完毕后 在此拉高时钟线，便于
       begin
          SCK <= 1'b1;//准备在下降沿放置数据，提前将SCK拉高
          Data_State <= Data_State + 4'd1;//切换为数据放置状态(每发完1bit数据进入此一次，将
       end
        D7_State://第7位数据发送状态
        begin
            MOSI <= Send_Data[7];//D7数据
            SCK <= 1'b0;//在下降沿放置数据
            Data_State <= Data_State + 4'd1;//切换状态
        end
        D6_State://第6位数据发送状态
        begin
             MOSI <= Send_Data[6];//D6数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D5_State://第5位数据发送状态
        begin
             MOSI <= Send_Data[5];//D5数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D4_State://第4位数据发送状态
        begin
             MOSI <= Send_Data[4];//D4数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D3_State://第3位数据发送状态
        begin
             MOSI <= Send_Data[3];//D3数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D2_State://第2位数据发送状态
        begin
             MOSI <= Send_Data[2];//D2数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D1_State://第1位数据发送状态
        begin
             MOSI <= Send_Data[1];//D1数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        D0_State://第0位数据发送状态
        begin
             MOSI <= Send_Data[0];//D0数据
             SCK <= 1'b0;//在下降沿放置数据
             Data_State <= Data_State + 4'd1;//切换状态
        end
        default: Data_State <= D7_State;
         endcase
    end
end
/*链接从机模块*/
SlaveGetMaster u2
(
    .CLK_50M(CLK_50M),
    .RST_N(RST_N),
    .MOSI(MOSI),
    .CS(CS),
    .SCK(SCK)
);
endmodule
```

代码解析见代码注释，整体代码思路即SPI的模式0:SCK的上升沿采样数据，SCK的下降沿切换数据。

?发送波形



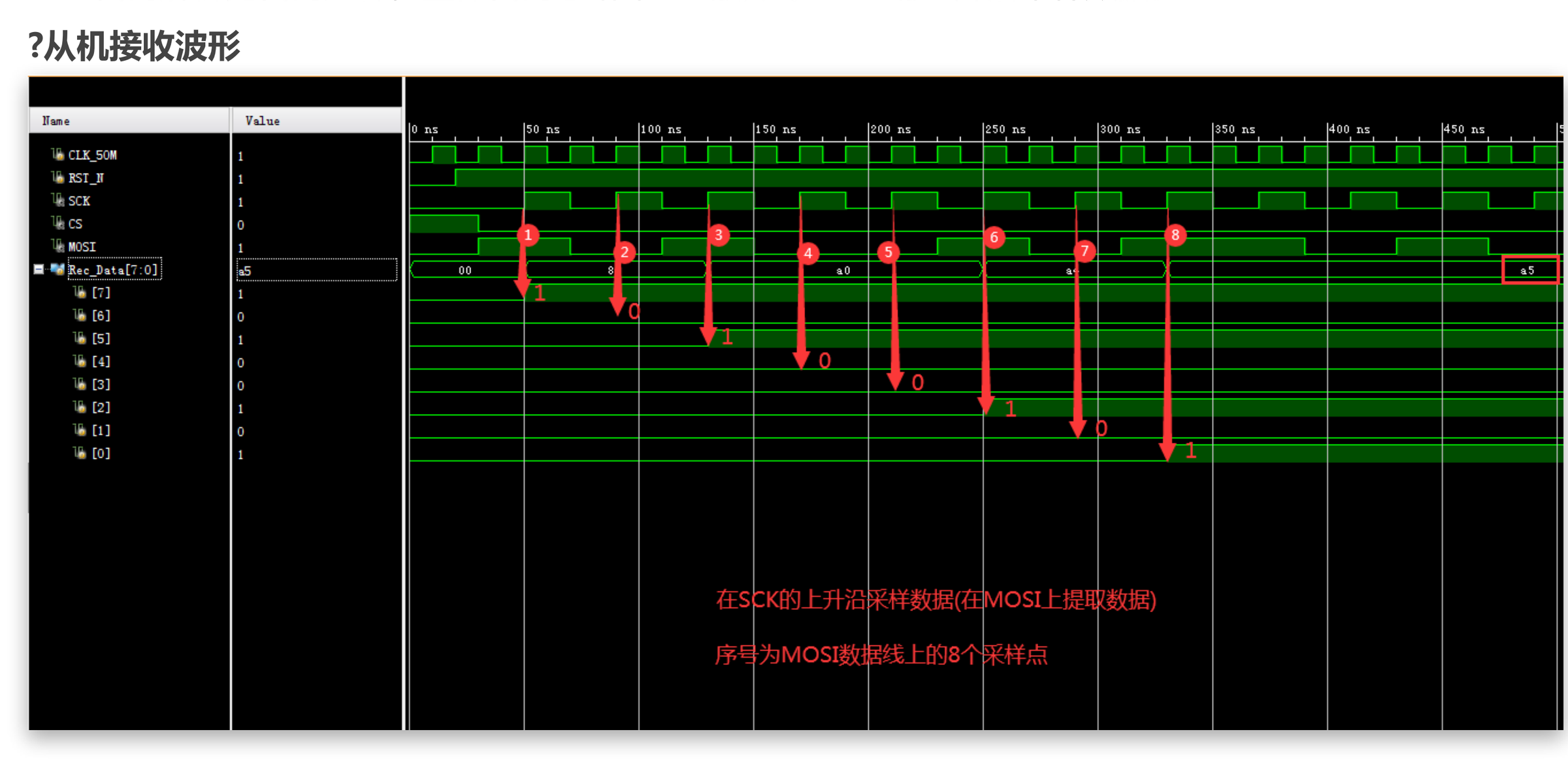
波形解析：主机发送的数据为0xA5，主机所执行的操作为将所要发送的8bit数据从高到低位依次在SCK的下降沿放置在MOSI数据线上，观察波形在图中①-⑧序号点为SCK的下降沿，在此下降沿时MOSI的数据进行了切换(因为从机要在时钟沿的上升沿采集数据，所以上升沿之前数据保持了稳定)。

?从机接收代码

```
//从机接收MOSI的数据    在上升沿的时候采样数据
module SlaveGetMaster(CLK_50M,RST_N,SCK,CS,MOSI);
    input CLK_50M;
    input RST_N;
    input SCK;
    input CS;
    input MOSI;
reg [7:0] Rec_Data=8'd0;
reg[3:0] Data_State = 4'd0;
parameter D7_State = 4'd0;
parameter D6_State = 4'd1;
parameter D5_State = 4'd2;
parameter D4_State = 4'd3;
parameter D3_State = 4'd4;
parameter D2_State = 4'd5;
parameter D1_State = 4'd6;
parameter D0_State = 4'd7;
always@(posedge SCK)
begin
    if(CS == 1)  //CS为高，从机不响应
        Rec_Data <= 8'b0000_0000;
    else        //CS为低，从机开始接收数据
    begin
          if(SCK == 1)
          begin
            case(Data_State)
            D7_State:begin Rec_Data[7] <= MOSI; Data_State<= D6_State;end
            D6_State:begin Rec_Data[6] <= MOSI; Data_State<= D5_State;end
            D5_State:begin Rec_Data[5] <= MOSI; Data_State<= D4_State;end
            D4_State:begin Rec_Data[4] <= MOSI; Data_State<= D3_State;end
            D3_State:begin Rec_Data[3] <= MOSI; Data_State<= D2_State;end
            D2_State:begin Rec_Data[2] <= MOSI; Data_State<= D1_State;end
            D1_State:begin Rec_Data[1] <= MOSI; Data_State<= D0_State;end
            D0_State:begin Rec_Data[0] <= MOSI; Data_State<= D7_State;end
            default:Data_State<= D7_State;
            endcase
          end
    end
end
endmodule
```

代码解析见代码注释，整体代码思路即SPI的模式0:SCK的上升沿采样数据。

?从机接收波形



波形解析：从机执行的操作为在SCK的上升沿对MOSI数据进行采样，即图中①-⑧序号点为从机对MOSI数据线的采样点。仔细观察采样点在每bit数据的中间点，满足采样定理。8bit数据均接收完毕后所得的数据为0xA5，正是主机发送的数据。

注意：若从机发送主机接收：则从机将数据放置在MISO数据线，主机从MISO数据线上采样数据。
经过以上描述应该明白MISO,MOSI数据线不能交叉连接的原因了*–*。

附：SPI协议_STM32实现

如有错误欢迎指导。