FPGA串口实验总结之RX

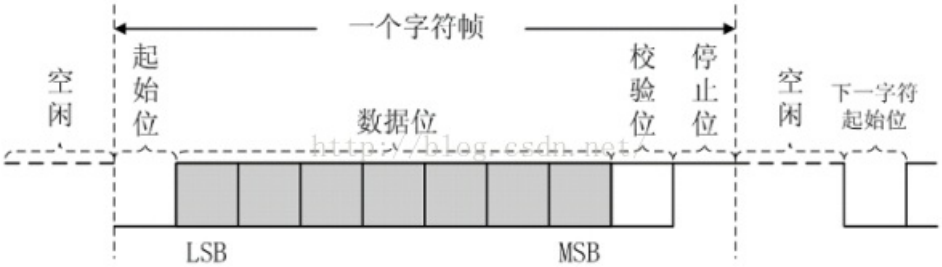
原创

zhitianya 于 2016-01-03 16:54:44 发布 4143 收藏 7
分类专栏： 实验总结 文章标签： fpga 串口时序 RX 串口仿真

本文链接：https://blog.csdn.net/zhitianya/article/details/50451502

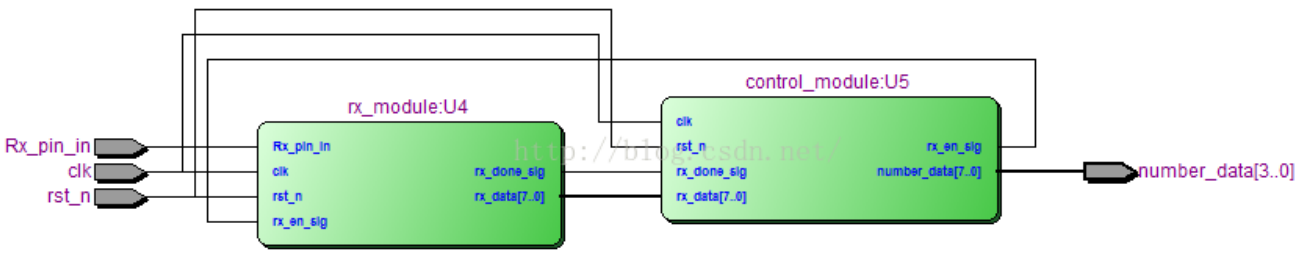
初接触FPGA的同学，必定要做的实验就是FPGA实验，下面我们就进行串口（232）实验RX部分

首先我们需要了解的就是串口在传输数据的时候到底发生了些什么事情：

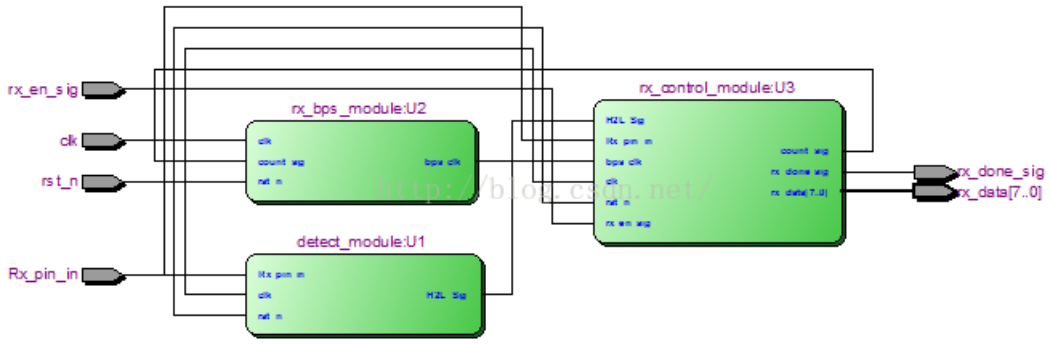


上图就是串口的时序图，我们可以注意到：
1、串口的默认状态为高电平，也就是说在没有数据传送的时候，引脚电平为高电平
2、有数据的时候首先要有一个起始位，起始位为低电平
3、起始位之后是8位的有效数据
4、在有效数据之后会有一位校验位和一位停止位
5、8位数据的传送首先是从地位开始的（知道这一点是非常重要的）
除了时序，波特率也是非常重要的，我们通常用的波特率有9600bps,19200bps,38400bps,115200bps等，bps:（bit per second），例如9600bps就是每秒可以传送数据9600位，理解了这些，我们就可以得到以下计算公式：
一位数据所占用的时间=1/9600；
我们一个数据发送周期为11位的话，则共需要时间11*（1/9600）；
那么一秒就可以传送多少个周期，相信就可以自己计算了吧！
那么我们的数据是怎么采集的呢？

我们需要一个波特率产生模块，此模块用来产生数据采集的心跳，为了准确的采集数据，我们应该在每个数据位的中间采集数据，因为此时数据位应该是最为稳定的，当然我们也可以采用一个数据位采集多次，然后根据一点的判定条件来确定踩到的数据是什么，在这个实验中，我们忽略校验位和停止位。
我的实验的RTL视图如下图所示：



其中rx_module:U4的内部模块如下图：



首先是detect_module：

```
/*
此模块是要用来检测起始位的，即检测到数据输入引脚由高电平变为低电平时，
就会产生一个周期的高电平脉冲，说明一个数据周期已经开始了，应该准备好开始接收数据了。
虽然存在再数据传输的时候也会产生由高电平变为低电平的情况，但是当第一个高脉冲周期产生后，
在今后的一个数据传输周期内后面产生的高脉冲自动视为无效了，这一点在其他模块的处理方法当中可以体现出来
*/

module detect_module(
        clk,rst_n,
        Rx_pin_in,
        H2L_Sig
);
input    clk;
input    rst_n;
input    Rx_pin_in;//数据的输入引脚
output   H2L_Sig;//检测到高电平变为低电平之后输出一个周期高电平

//wire    H2L_Sig;

//----------------------------------------
reg    H2L_F1;
reg    H2L_F2;
//----------------------------------------
always@(posedge clk or negedge rst_n)
 if(!rst_n)
  begin
   H2L_F1 <= 1'b1;
   H2L_F2 <= 1'b1;
  end
 else
  begin
   H2L_F1 <= Rx_pin_in;
   H2L_F2 <= H2L_F1;
  end


assign  H2L_Sig = H2L_F2&~H2L_F1;

endmodule
```

rx_bps_module：
波特率计数的计算(20Mhz，9600bps为例)：
上面讲过，一位数据传送需要的时间=1/9600
因此一位数据的计数为（1/9600）/(1/20M)=2083

此处20Mhz=20*10^6

```
/*此模块是用来产生波特率的，即产生采集标志位，*/

module rx_bps_module(
       clk,rst_n,
       count_sig,
       bps_clk
);

input    clk;
input    rst_n;
input    count_sig;//波特率产生控制位高电平有效
output   bps_clk;
//-----------------------------------
reg [12:0]  count_bps;

always@(posedge clk or negedge rst_n)
 if(!rst_n)
  begin
   count_bps <= 13'b0;
  end
 else
  begin
   if(count_sig)//
    begin
     if(count_bps == 13'd2082)
      count_bps <= 13'd0;
     else
      count_bps <= count_bps + 1'b1;
    end
   else count_bps <= 13'd0;
  end
assign bps_clk = (count_bps == 13'd1041)?1'b1:1'b0;

endmodule
```

rx_control_module：

```
/*此模块的作用是用来根据波特率等信息的配合来采集数据*/
module rx_control_module(
         clk,rst_n,
         H2L_Sig,Rx_pin_in,bps_clk,rx_en_sig,
         count_sig,rx_data,rx_done_sig
//         bps_clk1
);

input    clk;
input    rst_n;

input    H2L_Sig;
input    Rx_pin_in;
input    bps_clk;
input    rx_en_sig;

//output   bps_clk1;
output   count_sig;
output [7:0] rx_data;//忽略起始位与校验位
output   rx_done_sig;
//----------------------------------------------------
```

```
//-------------------------------------------------
reg [3:0] i;
reg [7:0] rdata;
reg   iscount;
reg   isdone;
//-------------------------------------------------

always@(posedge clk or negedge rst_n)
 if(!rst_n)
  begin
   i <= 4'd0;
   rdata <= 8'd0;
   iscount <= 1'd0;
   isdone <= 1'd0;
  end
 else if(rx_en_sig)
  case(i)
   4'd0:
    if(H2L_Sig) begin i <= i + 1'b1;iscount <= 1'b1;end

   4'd1:
    if(bps_clk) begin i<= i + 1'b1;end

   4'd2,4'd3,4'd4,4'd5,4'd6,4'd7,4'd8,4'd9://有效数据
    if(bps_clk) begin i <= i + 1'b1; rdata[i-2] <= Rx_pin_in;end

   4'd10://校验位
    if(bps_clk) begin i <= i+1'b1;  end

   4'd11://结束位
    if(bps_clk) begin i <= i+1'b1; end

   4'd12:
    begin i <= i +1'b1;isdone <= 1'b1;iscount <= 1'b0; end

   4'd13:
    begin i <= 1'b0;isdone <= 1'b0; end
  endcase
//-----------------------------------------------------------
assign count_sig = iscount;
assign rx_done_sig = isdone;//当8位数据接收完毕后，拉高该标志位，（接收完成标志为，高电平有效）
assign rx_data = rdata;
//assign bps_clk1 = bps_clk;

endmodule
```

rx_module：

```
module rx_module(
     clk,rst_n,
     Rx_pin_in,rx_en_sig,
     rx_done_sig,rx_data
//     bps_clk1
);
input    clk;
input    rst_n;

input    Rx_pin_in;
input    rx_en_sig;

//output   bps_clk1;

output   rx_done_sig;
output  [7:0] rx_data;
//-------------------------------------
wire H2L_Sig;
wire count_sig;
wire bps_clk;
detect_module U1(
        .clk(clk),
        .rst_n(rst_n),
        .Rx_pin_in(Rx_pin_in),
        .H2L_Sig(H2L_Sig)
);

//-------------------------------------

rx_bps_module U2(
       .clk(clk),
       .rst_n(rst_n),
       .count_sig(count_sig),
       .bps_clk(bps_clk)
);
//-------------------------------------

rx_control_module U3(
         .clk(clk),
         .rst_n(rst_n),
         .H2L_Sig(H2L_Sig),
         .Rx_pin_in(Rx_pin_in),
         .bps_clk(bps_clk),
         .rx_en_sig(rx_en_sig),
         .count_sig(count_sig),
         .rx_data(rx_data),
         .rx_done_sig(rx_done_sig),
//         .bps_clk1(bps_clk1)
);

endmodule
```

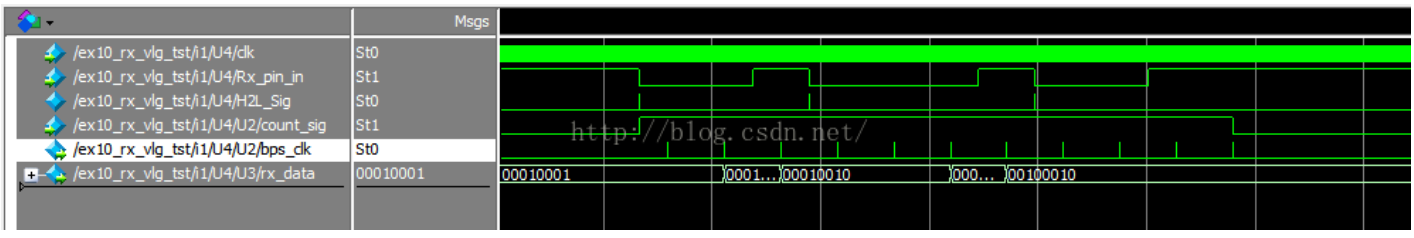
control_module:

```
//串口接收到一个字节的数据后，

module control_module(
        clk,rst_n,
        rx_done_sig,
        rx_data,
        rx_en_sig,
        number_data,
);
input    clk;
input    rst_n;
input    rx_done_sig;
input  [7:0] rx_data;
output   rx_en_sig;
output [7:0] number_data;
//-----------------------------------------------
reg [7:0] rdata;
reg isen;

always@(posedge clk or negedge rst_n)
 if(!rst_n)
  rdata <= 8'd0;
 else if(rx_done_sig)//当接收数据的寄存器值满后，再将值存到目标寄存器rdata
  begin
   rdata <= rx_data;
   isen <=1'b0;
  end
 else isen <= 1'b1;
//-----------------------------------------------
assign number_data = rdata;
assign rx_en_sig = isen;

endmodule
```

top_module:

```
module top_module(
     clk,rst_n,
     Rx_pin_in,
     number_data
//     bps_clk1
);
input    clk;
input    rst_n;

input    Rx_pin_in;
//output   bps_clk1;
output  [3:0] number_data;
//--------------------------------------
wire   rx_done_sig;
wire  [7:0] rx_data;
wire   rx_en;
wire [7:0] output_data;
wire   bps_clk;
rx_module U4(
     .clk(clk),
     .rst_n(rst_n),
     .Rx_pin_in(Rx_pin_in),
     .rx_en_sig(rx_en),
     .rx_done_sig(rx_done_sig),
     .rx_data(rx_data)
//     .bps_clk1(bps_clk1)
);

//--------------------------------------


control_module U5(
        .clk(clk),
        .rst_n(rst_n),
        .rx_done_sig(rx_done_sig),
        .rx_data(rx_data),
        .rx_en_sig(rx_en),
        .number_data(output_data)
);


assign number_data = output_data[7:0];
endmodule
```

然后看仿真时序图：



以上代码经过验证，希望同学亲自验证，注意要根据自己的时钟频率更改波特率产生的计数！！！