**UART串口收发的原理与Verilog实现**

# 软件平台与硬件平台

软件：Vivado 2017

硬件：龙芯FPGA实验箱（不知道啥型号）

# 原理介绍

串口是串行接口（serial port）的简称，也称为串行通信接口或COM接口。串口通信是指采用串行通信协议（serial communication）在一条信号线上将数据一个比特一个比特地逐位进行传输的通信模式。 串口按电气标准及协议来划分，包括RS-232、RS-422、RS485等。其中最常用的就是RS-232接口。

RS-232接口有以下三个特性：

1、用了一个9针的连接器"DB-9"(早期的电脑有用25针的连接器"DB-25")

　　2、允许全双工通信(即通过串口发送数据和接收数据可以同时进行)

3、通信的最大速率大约在10KBytes/s左右

虽然DB-9接头一共有9根线，但是实现串口通信只需要其中的3根线就可以了，分别是：

1、pin-2：RXD(receive data)，接收串行数据

2、pin-3：RXD(transmit data)，发送串行数据

3、pin-5：GND(ground)，地线

在串口通信中，数据在1位宽的单条线路上进行传输，一个字节的数据要分为8次，由低位到高位按顺序一位一位的进行传送，这个过程称为数据的"串行化(serialized)"过程。由于串口通信是一种异步通信协议，并没有时钟信号随着数据一起传输，而且空闲状态(没有数据传输的状态)的时候，串行传输线为高电平1，所以发送方发送一个字节数据之前会先发送一个低电平0，接收方收到这个低电平0以后就知道有数据要来了，准备开始接收数据从而实现一次通信。串口通信的时序如下图所示：

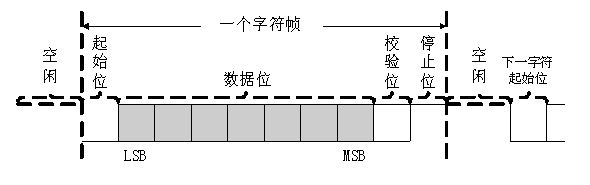


图1

串口通信的规范如下：

1、空闲状态(没有数据传输的状态)下，串行传输线上为高电平1

2、发送方发送低电平0表示数据传输开始，这个低电平表示传输的起始位

3、8-bit的数据位(1 Byte)是从最低位开始发送，最高位最后发送

4、数据位的最高位发送完毕以后的下一位是奇偶校验位，这一位可以省略不要，同时，当不发送奇偶校验位的时候接收方也相应的不接收校验位

5、最后一位是停止位，用高电平1表示停止位

下面以发送字节0x55为例来说明整个的发送过程：

先把0x55转化成二进制为：01010101。显然0x55的最低位bit 0是1，次低位bit 1是0，……..，最高位bit 7是0，由于串口是从最低位开始发送一个字节，所以0x55各个位的发送顺序是1-0-1-0-1-0-1-0,波形如下图所示

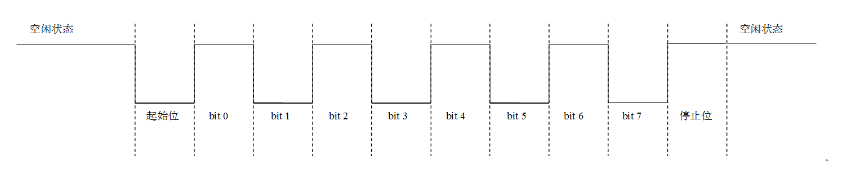


图2

下面在给出一个波形，根据上面的规则也可以很容易判断这是发送字节0x13的波形

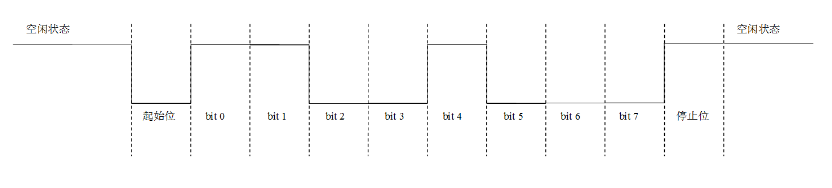


图3

接下来的最后一个问题是：串口传输的速度是多少？

实际上，串口传输的速度用波特率(baudrate)来指定。波特率表示的是每秒发送的比特数，单位是bps(bits-per-seconds)，例如，1000 bauds表示1秒钟发送了1000个比特，或者说每个比特持续的时间是1ms。关于串口发送的波特率是有一组标准的规定的，并不是随便一个数字。常用的波特率标准有：

1、1200 bps

2、9600 bps (常用)

3、38400 bps

4、115200 bps (常用，而且通常情况下是我们能用的最快的波特率)

波特率为115200 bps时，每个比特持续的时间为(1/115200)=8.7us，所以发送8个bit(1 Byte)需要的时间是8\*8.7us=69us。在不考虑奇偶校验位的情况下，发送一个字节还需要发送额外的1个起始位和1个停止位，所以发送1个字节实际所需要的最少时间是10\*8.7us=87us，这意味着1s(1000000us)中能发送的字节数为(1000000/87) = 11494,所以在波特率为115200bps的情况下，串口传输数据的速率约为11.5KB/s。而有些电脑的串口有时候需要一个更长的停止位，比如1.5位或2位的停止位，那么发送一个字节所需要的时间比只有一个比特停止位的情况所耗费的时间更长，在这种情况下，串口的传输速率会低于10.5KB/s。

通过上面一系列的总结以后，可以得出FPGA与PC之间的串口通信主要包括三个模块：波特率产生模块、发射模块和接收模块。

# 目标功能

　　1、编写发送模块的verilog代码，并往PC上连续不断发送0x00~0xff这些数据，PC上用串口调试助手进行接收并以16进制显示出来

　　2、在第一个功能的基础上编写接收模块的verilog代码，接收模块接收到第一个功能中发送模块发送的数据以后，用接收到的并行数据的低四位驱动板上的四个LED灯

3、编写一个顶层模块把发送模块和接收模块均例化进去，然后从PC的串口调试助手上发送数据到FPGA，FPGA接收到数据以后把接收的数据返回给串口调试助手显示。

# 四、设计思路与Verilog代码编写

## 4.1、发送模块波特率时钟的设计与实现

　　本节以波特率为115200bps为例来说明波特率模块设计方法，其余波特率可以以此类推。由于我的开发板上的时钟为50MHz，周期T=20ns，而波特率为115200bps，所以1个bit持续的时间是8.7us，那么每个bit占用的周期数N=(8.7us / 20ns) = 434，所以可以定义一个计数器，每当计数器从0计数到433的时候就把计数器清零，然后在计数值为1(这个计数值最好比433的一半要小，这篇博客的最后一部分分析了原因)的情况下产生一个高脉冲。发射模块只要检测到这个高脉冲的到来就发送一个bit，这样就实现了波特率为115200bps的串口数据发送。

　　而接收模块的波特率时钟产生逻辑与发送的波特率时钟相比稍有不同。不同之处在于当接收模块检测到I\_rs232\_rxd的下降沿以后，表示有数据过来，准备开始接收数据了，由于一个bit持续的时间为434个时钟周期，所以为了保证接收模块接收数据的准确性，我们需要在434/2=217个周期，也就是数据的正中间位置的时候把输入的数据接收并存起来。也就是说接收模块的波特率时钟要比发射模块的波特率时钟滞后数个周期

　　波特率产生模块的框图如下图所示

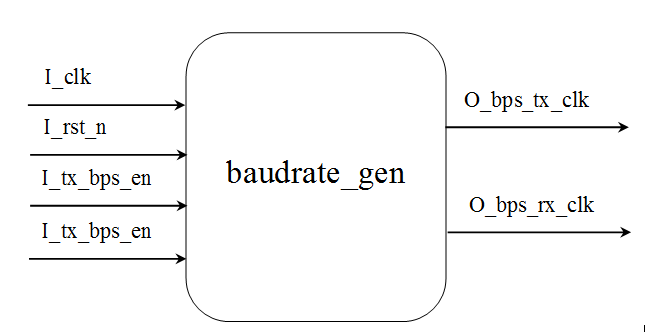


图4

其中：

　　I\_clk是系统时钟；

　　I\_rst\_n是系统复位；

　　I\_tx\_bps\_en是发射模块波特率使能信号，当I\_tx\_bps\_en为1时O\_bps\_tx\_clk才有时钟信号输出；

　　I\_rx\_bps\_en是接收模块波特率使能信号，当I\_rx\_bps\_en为1时O\_bps\_rx\_clk才有时钟信号输出。

波特率模块的完整代码如下：

module baudrate\_gen

(

input I\_clk , // System 50MHz clk

input I\_rst\_n , // Reset

input I\_bps\_tx\_clk\_en , // Transmit Enable

input I\_bps\_rx\_clk\_en , // Receive Enable

output O\_bps\_tx\_clk , // 发送模块波特率产生时钟

output O\_bps\_rx\_clk // 接收模块波特率产生时钟

);

parameter C\_BPS9600 = 5207 ,

C\_BPS19200 = 2603 ,

C\_BPS38400 = 1301 ,

C\_BPS57600 = 867 ,

C\_BPS115200 = 433 ;

parameter C\_BPS\_SELECT = C\_BPS115200 ;

reg [12:0] R\_bps\_tx\_cnt;

reg [12:0] R\_bps\_rx\_cnt;

///////////////////////////////////////////////////////////

// transmit

///////////////////////////////////////////////////////////

always @(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

R\_bps\_tx\_cnt <= 13'd0 ;

else if(I\_bps\_tx\_clk\_en == 1'b1)

begin

if(R\_bps\_tx\_cnt == C\_BPS\_SELECT)

R\_bps\_tx\_cnt <= 13'd0 ;

else

R\_bps\_tx\_cnt <= R\_bps\_tx\_cnt + 1'b1 ;

end

else

R\_bps\_tx\_cnt <= 13'd0 ;

end

assign O\_bps\_tx\_clk = (R\_bps\_tx\_cnt == 13'd1) ? 1'b1 : 1'b0 ;

///////////////////////////////////////////////////////////

// receive

///////////////////////////////////////////////////////////

always @(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

R\_bps\_rx\_cnt <= 13'd0 ;

else if(I\_bps\_rx\_clk\_en == 1'b1)

begin

if(R\_bps\_rx\_cnt == C\_BPS\_SELECT)

R\_bps\_rx\_cnt <= 13'd0 ;

else

R\_bps\_rx\_cnt <= R\_bps\_rx\_cnt + 1'b1 ;

end

else

R\_bps\_rx\_cnt <= 13'd0 ;

end

assign O\_bps\_rx\_clk = (R\_bps\_rx\_cnt == C\_BPS\_SELECT >> 1'b1) ? 1'b1 : 1'b0 ;

endmodule

运行的仿真结果为：

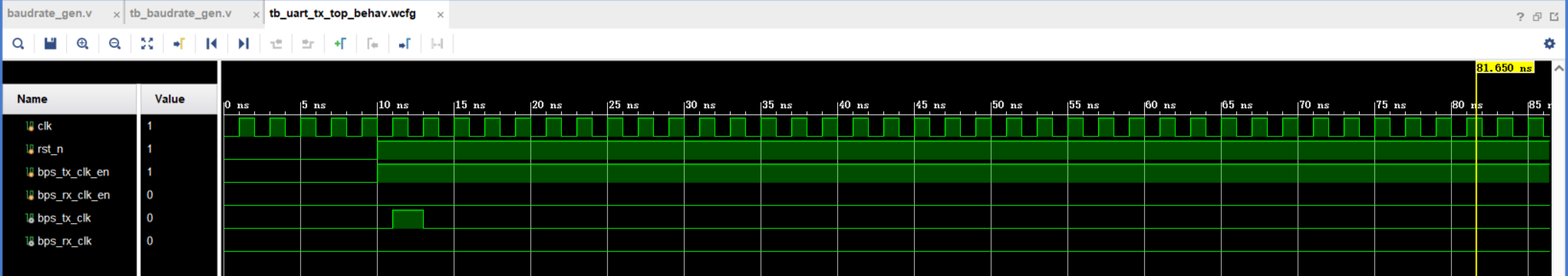


图5

## 4.2、发送模块的设计与实现

　　有了波特率时钟以后，就可以开始编写发送模块的内部逻辑了。发送模块的结构框图如下图所示

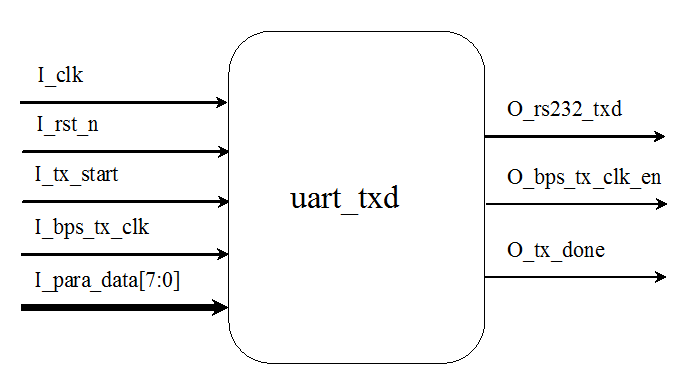


图6

其中：

　　I\_clk是系统时钟；

　　I\_rst\_n是系统复位；

　　I\_tx\_start是开始发送信号，当检测到I\_tx\_start为高电平时，立马把输入I\_para\_data[7:0]的数据串行化成单bit的发出去；

　　I\_bps\_tx\_clk是发送模块波特率时钟信号，当检测到I\_bps\_tx\_clk为高的时候就发送1个bit；

　　I\_para\_data[7:0]是并行的8-bit数据；

　　O\_rs232\_txd是串行的bit数据流；

　　O\_bps\_clk\_en是发射波特率时钟启动信号，当它为1是波特率产生模块才能产生发射模块的波特率时钟；

　　O\_tx\_done是发送1字节数据完成的标志位，当一个字节发送完毕以后，O\_tx\_done产生一个高脉冲。

　　以发送字节0x55为例，发送模块几个关键信号的时序图如下图所示

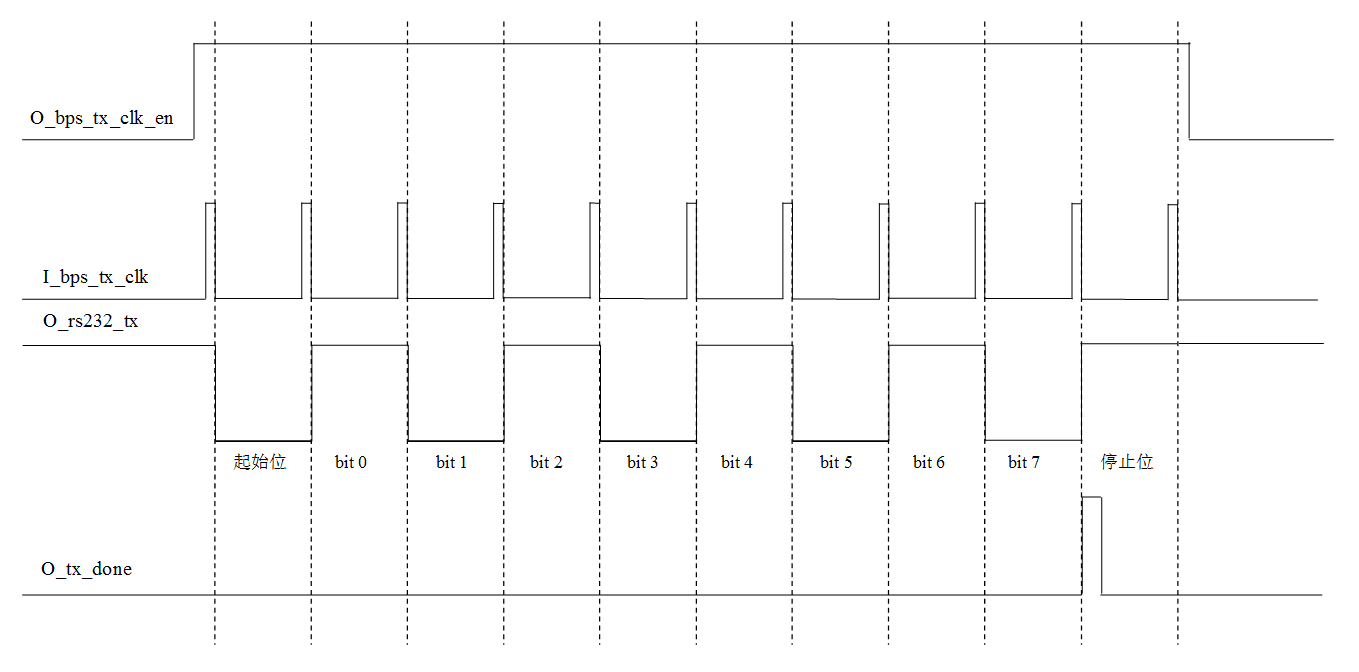


图7

　　发送模块的代码如下：

module uart\_txd

(

input I\_clk , //50MHz

input I\_rst\_n , // 系统全局复位

input I\_tx\_start , // 发送使能信号

input I\_bps\_tx\_clk , // 发送波特率时钟

input [7:0] I\_para\_data , // 要发送的并行数据

output reg O\_rs232\_txd , // 发送的串行数据，在硬件上与串口相连

output reg O\_bps\_tx\_clk\_en , // 波特率时钟使能信号

output reg O\_tx\_done // 发送完成的标志

);

reg [3:0] R\_state ;

reg R\_transmiting ; // 数据正在发送标志

/////////////////////////////////////////////////////////////////////////////

// 产生发送 R\_transmiting 标志位

/////////////////////////////////////////////////////////////////////////////

always @(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

R\_transmiting <= 1'b0 ;

else if(O\_tx\_done)

R\_transmiting <= 1'b0 ;

else if(I\_tx\_start)

R\_transmiting <= 1'b1 ;

end

/////////////////////////////////////////////////////////////////////////////

// 发送数据状态机

/////////////////////////////////////////////////////////////////////////////

always @(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

begin

R\_state <= 4'd0 ;

O\_rs232\_txd <= 1'b1 ;

O\_tx\_done <= 1'b0 ;

O\_bps\_tx\_clk\_en <= 1'b0 ;

end

else if(R\_transmiting)

begin

O\_bps\_tx\_clk\_en <= 1'b1 ;

if(I\_bps\_tx\_clk)

begin

case(R\_state)

4'd0 :

begin

O\_rs232\_txd <= 1'b0 ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd1 : //I\_para\_data[0]

begin

O\_rs232\_txd <= I\_para\_data[0] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd2 : //I\_para\_data[1]

begin

O\_rs232\_txd <= I\_para\_data[1] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd3 : //I\_para\_data[2]

begin

O\_rs232\_txd <= I\_para\_data[2] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd4 : //I\_para\_data[3]

begin

O\_rs232\_txd <= I\_para\_data[3] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd5 : //I\_para\_data[4]

begin

O\_rs232\_txd <= I\_para\_data[4] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd6 : //I\_para\_data[5]

begin

O\_rs232\_txd <= I\_para\_data[5] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd7 : //I\_para\_data[6]

begin

O\_rs232\_txd <= I\_para\_data[6] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd8 : //I\_para\_data[7]

begin

O\_rs232\_txd <= I\_para\_data[7] ;

O\_tx\_done <= 1'b0 ;

R\_state <= R\_state + 1'b1 ;

end

4'd9 : //end

begin

O\_rs232\_txd <= 1'b1 ;

O\_tx\_done <= 1'b1 ;

R\_state <= 4'd0 ;

end

default :R\_state <= 4'd0 ;

endcase

end

end

else

begin

O\_bps\_tx\_clk\_en <= 1'b0 ;

R\_state <= 4'd0 ;

O\_tx\_done <= 1'b0 ;

O\_rs232\_txd <= 1'b1 ;

end

end

endmodule

其中当检测到输入信号I\_tx\_start为高电平以后，发送模块立即把R\_transmiting信号拉高，表示开始要发送数据了，在R\_transmiting为高电平的期间，打开波特率时钟使能信号并且在波特率时钟的控制下通过一个状态机把并行数据发送出去，并产生发送完成信号O\_tx\_done，等O\_tx\_done为高以后再把R\_transmiting拉低表示一次发送结束。

为了实现功能1的效果还需要编写一个顶层模块把波特率模块和发送模块例化进去并产生发送的信号，顶层模块的代码如下：

module uart\_tx\_top

(

input I\_clk , // 系统50MHz时钟

input I\_rst\_n , // 系统全局复位

output O\_rs232\_txd // 发送的串行数据，在硬件上与串口相连

);

wire W\_bps\_tx\_clk;

wire W\_bps\_tx\_clk\_en;

wire W\_tx\_start;

wire W\_tx\_done;

wire [7:0] W\_para\_data;

reg [7:0] R\_data\_reg;

reg [31:0] R\_cnt\_1s;

reg R\_tx\_start\_reg;

assign W\_tx\_start = R\_tx\_start\_reg;

assign W\_para\_data = R\_data\_reg;

/////////////////////////////////////////////////////////////////////

// Generate the data

/////////////////////////////////////////////////////////////////////

always @(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

begin

R\_cnt\_1s <= 31'd0;

R\_data\_reg <= 8'd0;

R\_tx\_start\_reg <= 1'b0;

end

else if(R\_cnt\_1s == 31'd24\_999\_999)

begin

R\_cnt\_1s <= 31'd0;

R\_data\_reg <= R\_data\_reg + 1'b1;

R\_tx\_start\_reg <= 1'b1;

end

else

begin

R\_cnt\_1s <= R\_cnt\_1s + 1'b1;

R\_tx\_start\_reg <= 1'b0;

end

end

uart\_txd U\_uart\_txd

(

.I\_clk (I\_clk),

.I\_rst\_n (I\_rst\_n),

.I\_tx\_start (W\_tx\_start),

.I\_bps\_tx\_clk (W\_bps\_tx\_clk),

.I\_para\_data (W\_para\_data),

.O\_rs232\_txd (O\_rs232\_txd),

.O\_bps\_tx\_clk\_en (W\_bps\_tx\_clk\_en),

.O\_tx\_done (W\_tx\_done)

);

baudrate\_gen U\_baudrate\_gen

(

.I\_clk (I\_clk),

.I\_rst\_n (I\_rst\_n),

.I\_bps\_tx\_clk\_en (W\_bps\_tx\_clk\_en),

.I\_bps\_rx\_clk\_en ( ),

.O\_bps\_tx\_clk (W\_bps\_tx\_clk),

.O\_bps\_rx\_clk ( )

);

endmodule

将R\_cnt\_1s修改为4999，即5000个时钟周期传输一个消息

仿真结果为：

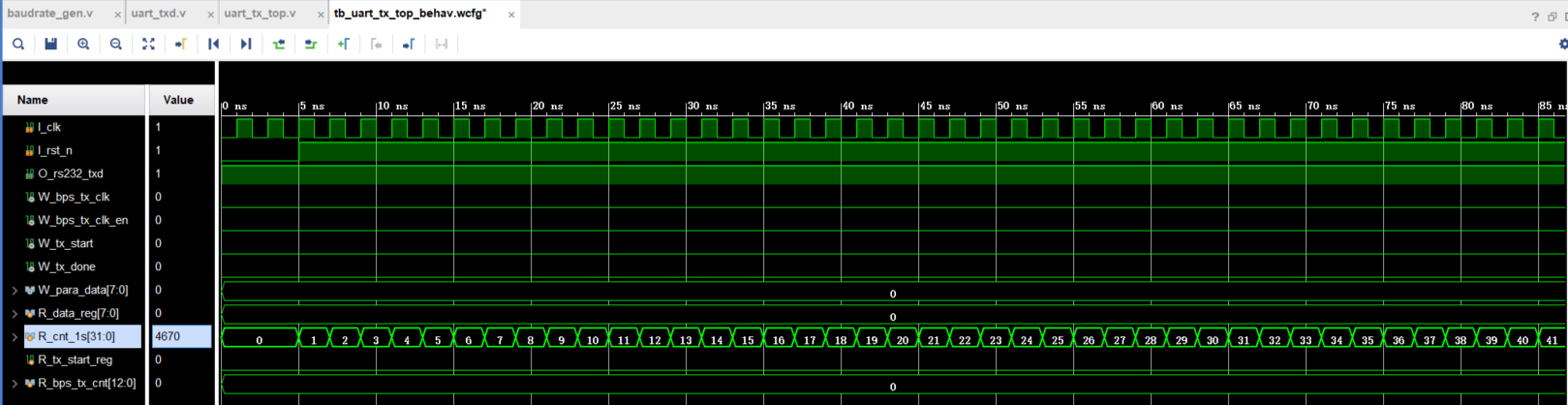


图8

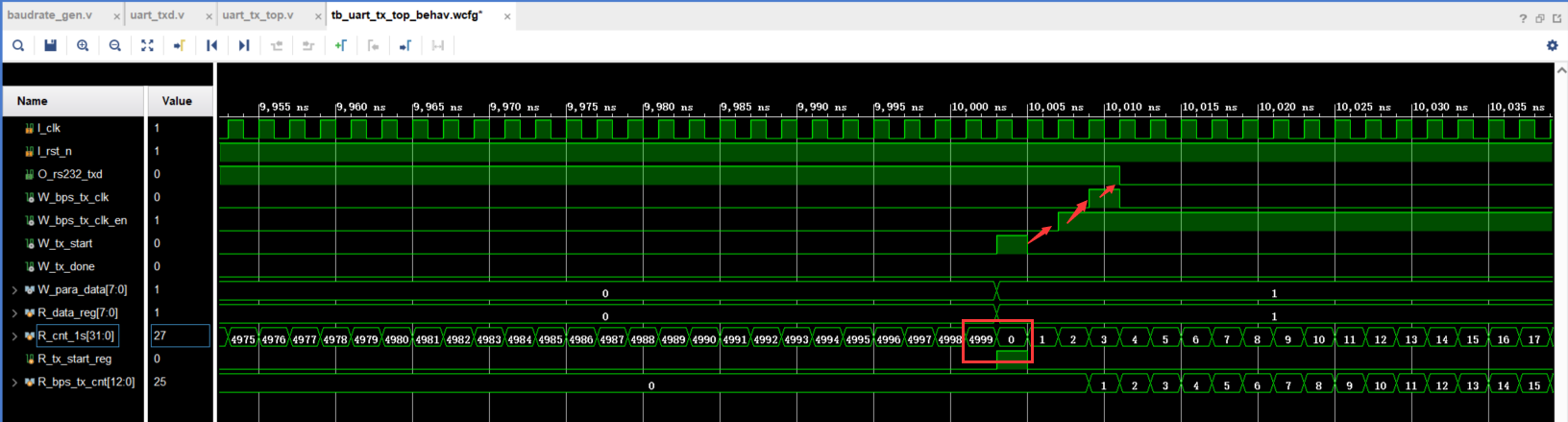


图9

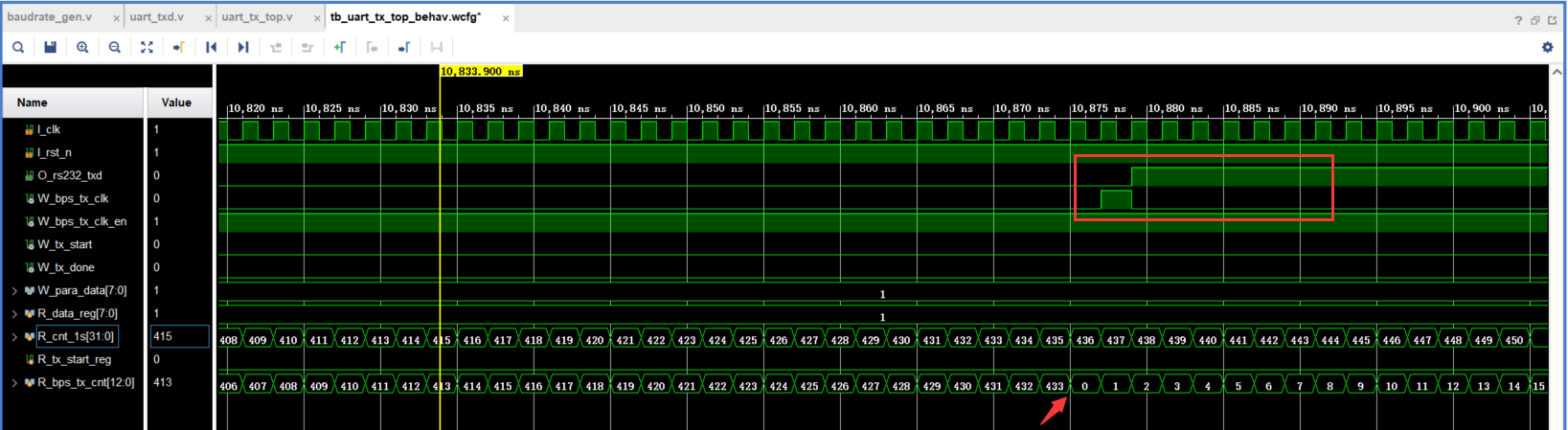


图10

在本地仿真完成后，便要接入板上测试，我们编写约束文件：

set\_property PACKAGE\_PIN H7 [get\_ports {O\_led\_out[3]}]

set\_property PACKAGE\_PIN D5 [get\_ports {O\_led\_out[2]}]

set\_property PACKAGE\_PIN A3 [get\_ports {O\_led\_out[1]}]

set\_property PACKAGE\_PIN A5 [get\_ports {O\_led\_out[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {O\_led\_out[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {O\_led\_out[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {O\_led\_out[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {O\_led\_out[3]}]

set\_property PACKAGE\_PIN AC19 [get\_ports I\_clk]

set\_property PACKAGE\_PIN Y3 [get\_ports I\_rst\_n]

set\_property IOSTANDARD LVCMOS33 [get\_ports I\_clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports I\_rst\_n]

set\_property PACKAGE\_PIN H19 [get\_ports O\_rs232\_txd]

set\_property IOSTANDARD LVCMOS33 [get\_ports O\_rs232\_txd]

set\_property PACKAGE\_PIN F23 [get\_ports I\_rs232\_rxd]

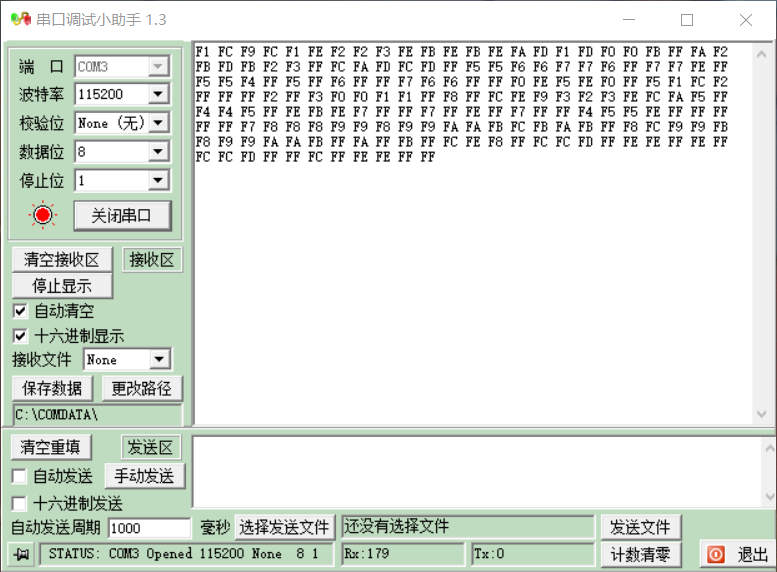
set\_property IOSTANDARD LVCMOS33 [get\_ports I\_rs232\_rxd]

选取芯片：xc7a200tfbg676-1 (active)

注意：约束文件一个字都不能打错，一个空格都不能少

然后点击，生成比特流文件，时间较长，耐心等待。

最后接入FPGA实验箱，安装驱动（都是自动的），指的提醒的是，在比特流文件拷入实验箱之前（包括进行调试的时候，实验箱都不要断电，因为实验箱使用ROM，断电就会丢失文件），最后测试成功的截图：



还有接收模块、串口回显功能等还没有完成，以后有空会继续实现。

参考：[【接口时序】3、UART串口收发的原理与Verilog实现](https://www.cnblogs.com/liujinggang/p/9535366.html)（[jgliu的博客园](https://www.cnblogs.com/liujinggang/)）