UART serial communication module driver based on STEP FPGA

In this section, we will use the FPGA to drive the UART interface communication on the backplane .

Hardware description

Universal Asynchronous Receiver/Transmitter, usually called UART , is a universal serial data bus used for asynchronous communication. This bus has two-way communication and can realize full-duplex transmission and reception.
Asynchronous communication uses one character as the transmission unit. The time interval between two characters in the communication is not fixed, but the time interval between two adjacent bits in the same character is fixed. The time interval between two adjacent bits is related to the baud rate of UART communication. The baud rate is used to characterize the data transmission rate in UART communication, that is, the number of binary digits transmitted per second. For example, the data transmission rate is 120 characters per second, and each character is 10 bits (1 start bit, 7 data bits, 1 check bit, 1 end bit), then the baud rate of the transmission is 10 ×120=1200 characters/second=1200 baud.

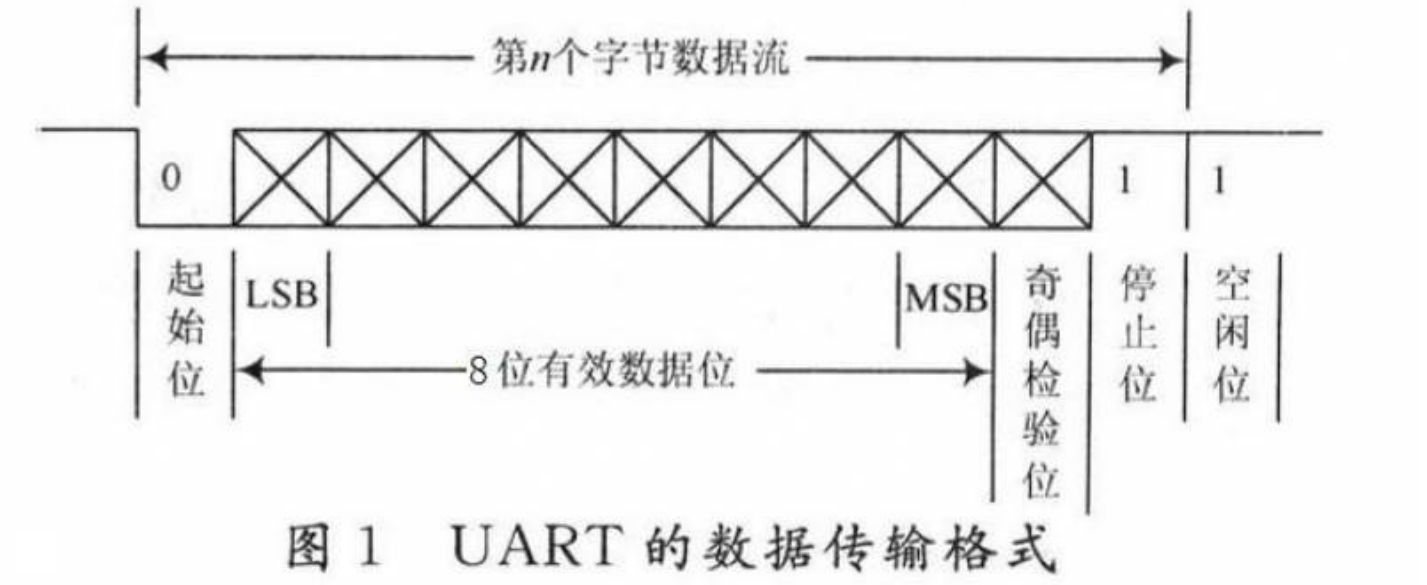
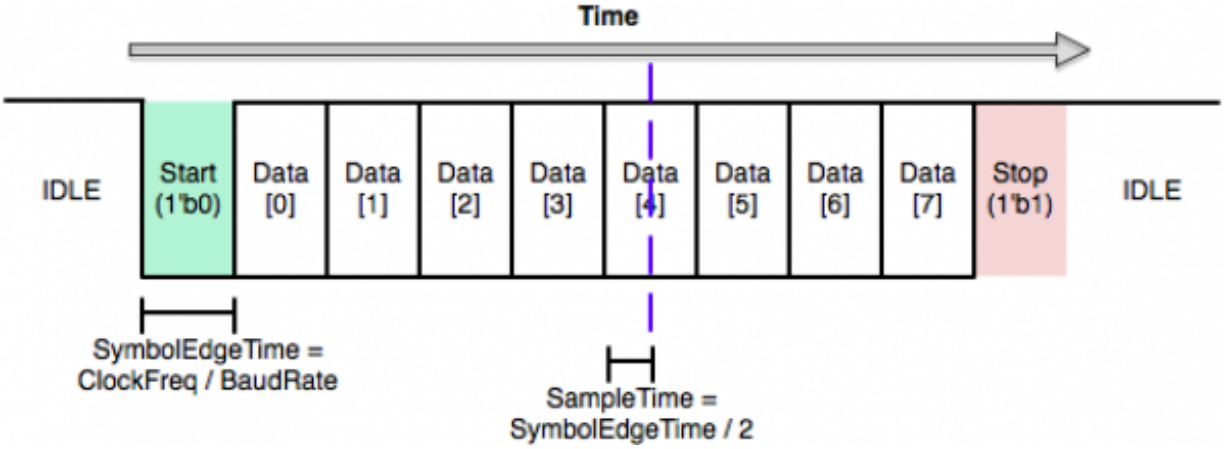


图 1 UART 的数据传输格式

- Start bit: first send out a logic "0" signal to indicate the beginning of the transmission character.
- Data bit: It can be 5~8 bit logic "0" or "1". Such as ASCII code (7 bits), extended BCD code (8 bits). Little endian transmission
- Check bit: After adding this bit to the data bit, the number of "1" bits should be even (even parity) or odd (odd parity)
- Stop bit: It is the end sign of a character data. It can be 1-bit, 1.5-bit, or 2-bit high level.
- Idle bit: in the logic "1" state, indicating that there is no data transmission on the current line.

The timing here we used to get rid of parity bit timing of this design has four modules, a top module, a baud module, a receiving module and a sending module, you can adjust according to their needs.



Verilog code

```
// ---------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ---------------------------------------------
// Module: Uart_bus
/ /
// Author: Step
//
// Description: The module for uart communication
//
// Web: www.stepfapga.com
//
// ------------------ ---------------------------------------------------
// Code Revision History:
// ---------------------------------------- -------------------------
// Version: |Mod. Date: |Changes Made:
// V1.0 |2016/04/20 |Initial ver
// ---------------------------------------------- --------------------
module Uart_Bus #
(
parameter                               BPS_PARA =  1250  //When using 12MHz clock, the baud r
ate parameter selects 1250 corresponding to 9600 baud rate
)
(
input                                   clk_in ,                        //System clock
input                                   rst_n_in ,              //System reset, low effective
input                                   rs232_rx ,              //UART receiving end in FPGA,
 assigned to the sending end in UART module TXD
output                                  rs232_tx                 //UART sending end in FPGA, a
ssigned to UART module RXD in the receiving end
) ;

//////////////////////////////// UART receiving function module instantiation//////////
 ////////////////////////
wire                                    bps_en_rx , bps_clk_rx ;
wire                    [ 7 : 0 ]       rx_data ;

//UART receive baud rate clock control module instantiation
Baud #
(
.BPS_PARA                                ( BPS_PARA             )
)
Baud_rx
(
.clk_in                                  ( clk_in                        ) ,    //System
clock.rst_n_in                           ( rst_n_in             ) ,    //System reset, low
active.bps_en                                   ( bps_en_rx             ) ,    //Receive cloc
k
enable.bps_clk                           ( bps_clk_rx           )      //Receive clock output
) ;

//UART receive data module instantiation
Uart_Rx Uart_Rx_uut
```

```
(
.clk_in                                 ( clk_in                        ) ,    //System
clock.rst_n_in                          ( rst_n_in              ) ,    //System reset, active
low.bps_en                                      ( bps_en_rx             ) ,    //Receive cloc
k
enable.bps_clk                          ( bps_clk_rx            ) ,    //Receive clock
input.rs232_rx                          ( rs232_rx              ) ,    //UART receives
input.rx_data                           ( rx_data               )      //Received data
) ;


//////////////////////////////// UART sending function module instantiation//////////
 ////////////////////////
wire                                    bps_en_tx , bps_clk_tx ;

//UART send baud rate clock control module instantiation
Baud #
(
.BPS_PARA                               ( BPS_PARA              )
)
Baud_tx
(
.clk_in                                 ( clk_in                        ) ,    //system
clock.rst_n_in                          ( rst_n_in              ) ,    //system reset, low
active.bps_en                                   ( bps_en_tx             ) ,    //send clock
enable.bps_clk                          ( bps_clk_tx            )      //send clock output
) ;

//UART send data module instantiation
Uart_Tx Uart_Tx_uut
(
.Clk_in                                 ( CLK_IN                        ) ,    // the system
 clock
.rst_n_in                               ( rst_n_in              ) ,    // system reset, activ
e low
.bps_en                                 ( bps_en_tx             ) ,    // send clock enable
.bps_clk                                ( bps_clk_tx            ) ,    // send clock input
.rx_bps_en                              ( bps_en_rx             ) ,    //Due to the need for
 self-receiving and self-transmitting, use the receive clock enable to determine: new data nee
ds to be sent.
tx_data                         ( rx_data               ) ,    //data to be
sent.rs232_tx                           ( rs232_tx              )      //UART transmission ou
tput
) ;

endmodule
```

```
// ---------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ----------------------------------------------
// Module: Baud
/ /
// Author: Step
//
// Description: Beat for uart transfer and receive baud rate
//
// Web: www.stepfapga.com
//
// --------------- ---------------------------------------------------- ---
// Code Revision History:
// ------------------------------------ -----------------------------
// Version: |Mod. Date:|Changes Made:
// V1.0 |2016/04/20 |Initial ver
// ---------------------------------------------- --------------------
module Baud #
(
parameter                               BPS_PARA =  1250  //When using 12MHz clock, the baud r
ate parameter selects 1250 corresponding to 9600 baud rate
)
(
input                                   clk_in ,                //system clock
input                                   rst_n_in ,      //system reset, low active
input                                   bps_en ,                //receive or transmit clock en
able
output  reg                             bps_clk          //receive or transmit clock output
) ;

reg                             [ 12 : 0 ]      cnt ;
//Counter count meets the baud rate clock requirement
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in )
                cnt <=  1'b0 ;
        else  if ( ( cnt >= BPS_PARA - 1 ) || ( ! bps_en ) )  //When the clock signal is not e
nabled (bps_en is low), the counter is cleared and stops counting
                cnt <=  1'b0 ;                                                  //When the clock signa
l is enabled, the counter counts the system clock, the period is BPS_PARA system clock cycles
        else
                cnt <= cnt +  1'b1 ;
end

//Generate the clock beat of the corresponding baud rate, and the receiving module will receiv
e UART data at this beat
always  @  ( posedge clk_in or  negedge rst_n_in )
        begin
                if ( ! Rst_n_in )
                        bps_clk <=  1'b0 ;
                else  if ( cnt ==  ( BPS_PARA >> 1 ) )          //BPS_PARA shifted by one bit
 to the right is equal to dividing by 2. Because the final value of the counter BPS_PARA is th
e data replacement time point, the middle value of the counter is the most stable time point o
```

```
f data
                        bps_clk <=  1'b1 ;
                else
                        bps_clk <=  1'b0 ;
        end

endmodule
```

```
// ---------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ---------------------------------------------
// Module: Uart_Rx
/ /
// Author: Step
//
// Description: The receive module of uart interface
//
// Web: www.stepfapga.com
//
// ----------------- ---------------------------------------------------- -
// Code Revision History:
// ---------------------------------------- --------------------------
// Version: |Mod. Date: |Changes Made:
// V1.0 |2016/04/20 |Initial ver
// ---------------------------------------------- --------------------
module Uart_Rx
 (
input                                   clk_in ,                        //system clock
input                                   rst_n_in ,              //system reset, low active

output  reg                             bps_en ,                        //Receive clock enable
input                                   bps_clk ,               //Receive clock input

input                                   rs232_rx ,              //UART receive input
output  reg             [ 7 : 0 ]       rx_data                  // received data
) ;

reg     rs232_rx0 , rs232_rx1 , rs232_rx2 ;
//Multi-level delay latch to remove metastable state
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in )  begin
                rs232_rx0 <=  1'b0 ;
                rs232_rx1 <=  1'b0 ;
                rs232_rx1 <= <=  1'b0 ;
        end  else  begin
                rs232_rx0 <= rs232_rx ;
                rs232_rx1 <= rs232_rx0 ;
                rs232_rx2 <= rs232_rx1 ;
        end
end

//Detect the falling edge of the UART receiving input signal
wire    neg_rs232_rx = rs232_rx2 & rs232_rx1 &  ( ~ rs232_rx0 )  &  ( ~ rs232_rx ) ;

reg                             [ 3 : 0 ]       num ;
//Control of receiving clock enable signal
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in )
                bps_en <=  1'b0 ;
        else  if ( neg_rs232_rx &&  ( ! bps_en ) )      / /When the idle state (bps_en is lo
```

```
w), the falling edge of the UART receiving signal is detected and enters the working state (bp
s_en is high), and the control clock module generates the receiving clock
                bps_en <=  1'b1 ;
        else  if ( num ==4'd9 )                                 //After completing a UART rece
iving operation, exit the working state and restore the idle state
                bps_en <=  1'b0 ;
end

reg                             [ 7 : 0 ]       rx_data_r ;
//When in working state, get data according to the beat of the receiving clock
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in )  begin
                num <=  4'd0 ;
                rx_data <=  8'd0 ;
                rx_data_r <=  8'd0 ;
        end  else  if ( bps_en )  begin
                if ( bps_clk )  begin
                        num<= num + 1'b1 ;
                        if ( num <= 4'd8 )
                        rx_data_r [ num - 1 ] <= rs232_rx ;     // first accept the low bit an
d then the high bit, 8 bits of valid data
                end  else  if ( num ==  4'd9 )  begin   //After completing a UART receiving op
eration, output the acquired data
                        num <=  4'd0 ;
                        rx_data <= rx_data_r ;
                end
        end
end

endmodule
```

```
// ---------------------------------------------- --------------------
// >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE < <<<<<<<<<<<<<<<<<<<<<<<
// ------------------------ ----------------------------------------------
// Module: Uart_Tx
/ /
// Author: Step
//
// Description: The transfer module of uart interface
//
// Web: www.stepfapga.com
//
// ----------------- ---------------------------------------------------- -
// Code Revision History:
// ---------------------------------------- ---------------------------
// Version: |Mod. Date: |Changes Made:
// V1.0 |2016/04/20 |Initial ver
// ---------------------------------------------- --------------------
module Uart_Tx
 (
input                                   clk_in ,                        //System clock
input                                   rst_n_in ,              //System reset, low effective
output  reg                             bps_en ,                        //Send clock enable
input                                   bps_clk ,               //Sending clock input
input                                   rx_bps_en ,             //Because self-receiving and s
elf-transmitting are required, use the receive clock enable to determine: new data needs to be
sent
input                   [ 7 : 0 ]       tx_data ,               //data to be sent
output  reg                             rs232_tx                 //UART transmission output
) ;

reg                                             rx_bps_en_r ;
//delay latch receive clock enable signal
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in ) rx_bps_en_r <=  1'b0 ;
        else rx_bps_en_r <= rx_bps_en ;
end

//Detect the falling edge of the receive clock enable signal, because the falling edge represe
nts the completion of the received data, which is used as an incentive to send the signal
wire    neg_rx_bps_en = rx_bps_en_r &  ( ~ rx_bps_en ) ;

reg                             [ 3 : 0 ]       num ;
reg                             [ 9 : 0 ]       tx_data_r ;
//According to the completion of receiving data, drive sending data operation
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! rst_n_in )  begin
                bps_en <=  1'b0 ;
                tx_data_r <=  8'd0 ;
        end  else  if ( neg_rx_bps_en ) begin
                bps_en <= 1'b1 ;                                                        //When the fal
ling edge of the receive clock enable signal is detected, it indicates that the reception is c
omplete and data needs to be sent. Enable the transmit clock enable signal
```

```
                tx_data_r <=  { 1'b1 , tx_data , 1'b0 } ;
        end  else  if ( num == 4'd10 )  begin
                bps_en <=  1'b0 ;       //One UART transmission requires 10 clock signals, and
then end
        end
end

//When in working state, send data according to the beat of the sending clock
always  @  ( posedge clk_in or  negedge rst_n_in )  begin
        if ( ! Rst_n_in )  begin
                num <=  1'b0 ;
                rs232_tx <=  1'b1 ;
        end  else  if ( bps_en )  begin
                if ( bps_clk )  begin
                        num <= num +  1'b1 ;
                        rs232_tx <= tx_data_r [ num ];
                end  else  if ( num >= 4'd10 )
                        num <=  4'd0 ;
        end
end

endmodule
```

summary

This section mainly explains the principle of UART communication and software design for everyone. You need to create your own project while mastering it, and generate FPGA configuration file loading test through the entire design process . If you are not familiar with the use of Diamond software, please refer to here: Use of Diamond .

Relevant information

Use STEP-MXO2 second generation of UART communication program: subsequent download connection will be updated
using the STEP-MAX10 the UART communication program: subsequent download link will be updated