Design And Implementation of UART Based on Verilog HDL

Lingxi Kong[1, *], Qirui Niu[2] and Pai Yang[3]

[1] School of Electrical and Information Engineering, Tianjin University, Tianjin, China

[2] School of electronic and information engineering, Lanzhou Jiaotong university, Lanzhou, China

[3] School of Physics and Optoelectronic Engineering, Guangdong University of Technology, Guangzhou, China

* Corresponding Author Email: lingxi_kong@tju.edu.cn

Abstract. As a two-way transmission channel, Universal Asynchronous Receiver/Transmitter (UART) not only greatly improves the efficiency of information transmission between computers and external devices, but also ensures the accuracy and consistency of information by eliminating metastable state, setting baud rate and other means. In this paper, on the basis of fully understanding the definition and function of UART, based on Verilog HDL language to build UART, and through Modelsim simulation, image and data. The experimental results show that the receiving and sending module of this module works well and meets the requirements of full-duplex serial communication equipment. There is no doubt that the design of this paper has made a more detailed explanation of the basic operating principle of UART, which will contribute to its further development.

Keywords: UART, Verilog HDL, full-duplex serial communication, Modelsim.

1. Introduction

Computer is an indispensable part of the development of modern science and technology, and any development of it is of great significance to the progress of the age of science and technology. Universal Asynchronous Receiver/Transmitter is a universal serial data bus, which can meet the needs of information transmission between peripherals and computers, greatly increasing the efficiency of information transmission, and achieving full-duplex serial communication. It was a major milestone in the history of computing.

This paper will fully understand the structure and principle of UART, function and implementation on the basis of the use of Verilog HDL language, by describing its function, to achieve the construction of UART. By writing the code and running the simulation, the results this paper get agree with the expectation, which proves the feasibility of the construction method.

UART can be divided into sending module and receiving module according to functions. It transmits binary codes bit by bit and receives binary codes bit by bit. It is worth noting that in order to take into account the accuracy and efficiency of information transmission, the sending module and the receiving module have different methods of confirming information.

2. Implementation of UART function

2.1. UART operational principle

2.1.1. UART brief introduction

UART is a universal asynchronous receiver/transmitter that converts data to be transmitted between serial communication and parallel communication, enabling full-duplex transmission and receiving. It has a serial data stream that converts the parallel data inside the computer into the output, and the input serial data into the byte transmission bit by bit; Parity check of input and output data streams; Add functions such as deleting start and stop tags to the output data stream.

The frame format of asynchronous communication is usually 1 bit at the start, 5 to 8 bits at the data bit, parity bit (optional), parity or no parity, and stop bit (1, 1.5, or 2 bits). UART data frame format [2]. As can be seen Figure 1.

To speed up the operation process, the parity bit is not set in this operation. The data frame format is 1-bit start bit, 8-bit data bit, no parity bit, and 1-bit stop bit. The communication interface standard is Rs-232 [3] and full-duplex communication mode [4].

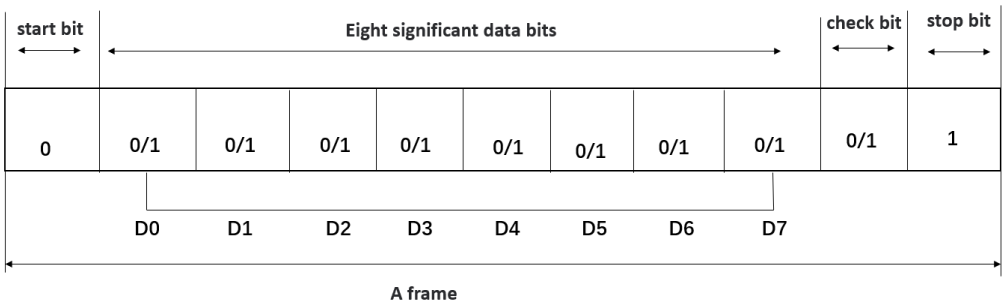

Fig. 1 Data frame format of UART

2.1.2 Division of modules

UART can be divided into two sub-modules: serial port sending module and serial port receiving module. The working diagram is shown in Figure 2 and Figure 3.

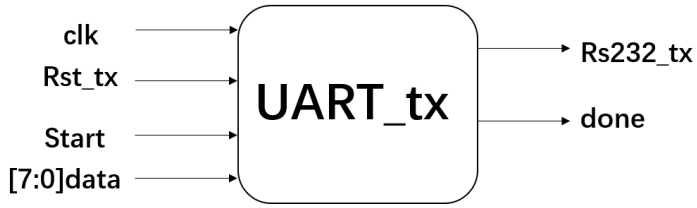

Fig. 2 UART sending module

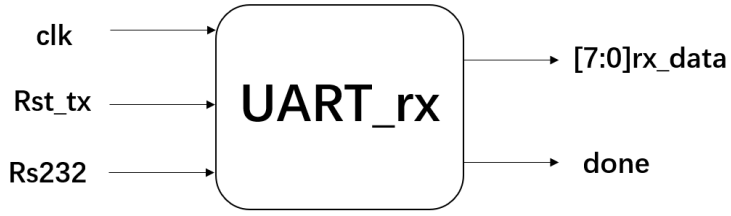

Fig. 3 UART receiving module

Clk stands for master clock, rst_n stands for reset signal, start stands for start signal, [7:0]data stands for data bits (8 bits); Rs232_tx indicates data transmission and done indicates termination signal.

2.2. Transmission module

The starting bit is sent first, followed by 8 valid data bits, the low bit is sent first, the high bit is sent last, and bits 1-7 of the byte are sent successively [5].

A high level is generated by the Start signal, marking the start of data transmission, and data begins to send data. r_data will cache the data data once to prevent the two data fusion, data, error, state logic 0/1 indicates idle state/working state, baud_cnt is the baud rate setting. Set as the clock frequency signal required by the UART module, count value 5208, send data once every count 5208 [6], bit_flag logic 1 send one bit of data (including the start bit, 8-bit data bit and stop bit), bit_cnt is the bit counter, count from zero, every ten is a cycle. A done signal is generated whenever the bit_cnt count is 10 and the bit_flag is at high level, and the state signal returns to low level at the same time. As can be seen Figure 4.

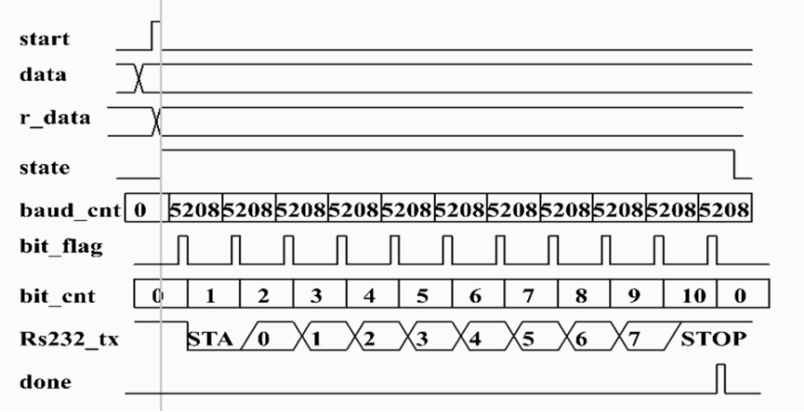

Fig. 4 Timing diagram of sending module

2.3. Receiving module

2.3.1. The elimination of metastable state

When the input of the first-stage trigger does not meet its establishment holding time, the output metastable data will stabilize before the next pulse edge arrives, and the stable data will meet the establishment time of the second-stage trigger. If the requirements are met, then the second stage trigger will not be metastable when the next pulse edge arrives [7]

2.3.2 Receiver timing design

Every time the level changes from "0" to "1", it can be regarded as the arrival of the starting bit of a frame data. However, the noise on the communication line is also likely to make logic "1" jump to logic "0", in order to ensure the accuracy of the information exchanged between the communication parties. In this paper, falling edge detection is used to filter the interference of communication line noise. baud_cnt is the baud rate setting, while bit_cnt and bit_flag jointly control whether rx_data receives each bit of data transmitted by Rs232. Finally, the receiving stops, the state signal is pulled down, and the done signal is pulled up. When the sampling counter counts, all data bits have been entered [8]. As can be seen Figure 5.

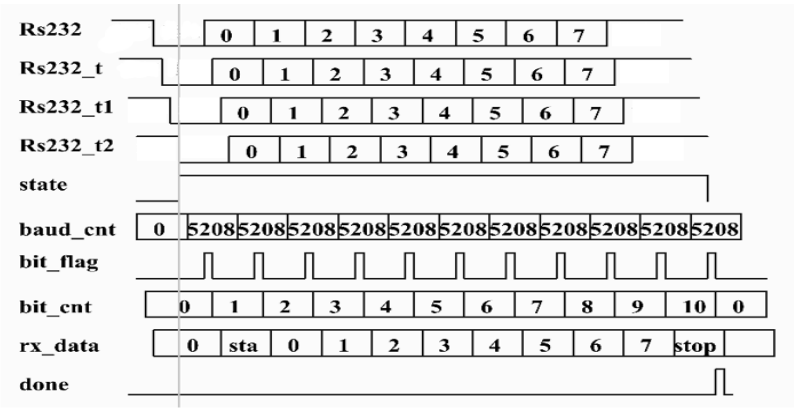

Fig. 5 Timing diagram of receiving module

3. Experiment Content

3.1. Code and Comments

3.1.1 UART sending module

[7:0]data is parallel data ready to be sent. When the input start signal appears high level, data in data signal is stored in r_data, and state is set to high level at this time. According to the baud rate setting, bit_cnt increases by 1 for each high level of bit_flag. The sender rs232_tx sends serial data in

r_data according to the count of bit_cnt. When bit_cnt counts to 10 and bit_flag is high level, a done signal is generated, making the state set to low level to end the transmission [9].

```
module uart_tx(
  input clk,
  input rst_n,
  input start,
  input [7:0]data,

  output reg rs232_tx,
  output reg done
);
```

This code first inputs the clock signal and sets it to 50MHz, uses rst_n to represent the low-level reset signal, and start to represent the start of signal transmission. Finally, the sending end rs232_tx sends the signal successfully, and generates the end signal done.

```
  reg  [7:0] r_data;
  reg state;
  reg [12:0] baud_cnt;
  reg bit_flag;
  reg [3:0] bit_cnt;
```

In order to prevent errors caused by two data fusion, data in data sigal will be stored in r_data. State indicates the working state, state low level indicates idle state, and high level indicates working state. Baud_cnt represents the baud rate counter, bit_flag represents the send flag, and bit_cnt represents the bit counter.

```
/*baud_cnt*/
  always@ (posedge clk or negedge rst_n) begin
      if (! rst_n)
          baud_cnt <= 'd0;
      else if(state)begin
          if (baud_cnt == 'd30)
              baud_cnt <= 'd0;
          else
              baud_cnt <= baud_cnt + 1'b1;
      end
      else
          baud_cnt <= 'd0;
  end
```

The baud_cnt starts counting when the state sets the high-power level. At the clock frequency of 50MHz, if the baud rate is 9600, baud_cnt sends one data bit every counting 50000000/9600=5208 clock cycles [10]. However, in order to facilitate the observation of simulation results, baud_cnt in the program takes every 30 clock cycles to send a data bit.

```
/*rs232_tx*/
      always@ (posedge clk or negedge rst_n) begin
      if (! rst_n)
          rs232_tx <= 1'b1;
      else if(state)begin
          if(bit_flag) begin
              case(bit_cnt)
                  4'd0:rs232_tx <= 1'b0;
                  4'd1:rs232_tx <= r_data[0];
                  4'd2:rs232_tx <= r_data[1];
                  4'd3:rs232_tx <= r_data[2];
```

```
                    4'd4:rs232_tx <= r_data[3];
                4'd5:rs232_tx <= r_data[4];
                4'd6:rs232_tx <= r_data[5];
                4'd7:rs232_tx <= r_data[6];
                4'd8:rs232_tx <= r_data[7];
                4'd9:rs232_tx <= 1'b1;
                default:rs232_tx <= 1'b1;
            endcase
        end
    end
     else
    rs232_tx <= 1'b1;
end
```

When the send flag bit_flag is high, the bit counter bit_cnt starts countingWhen bit_cnt is 0, rs232_tx sends out starting bit (low level). When bit_cnt ranges from 1 to 8, rs232_tx sends eight bits of r_data from low to high. When bit_cnt is 9, rs232_tx sends stop bit (high level).

3.1.2 UART receiving module

Rs232 is the sending data of the sending module, and it is also taken as the data to be received in the receiving module, so as to verify whether the design of the receiving module is correct. Similar to the sending module, the receiving end rx_data receives eight-bit data from rs232_t2 from low to high according to the count of bit_cnt.

```
module uart_rx(
  input clk,
  input rst_n,
  input rs232,
  output reg [7:0] rx_data,
  output reg done
);
```

Input the clock signal and set it to 50MHz. rst_n represents the low-level reset signal. The receiving end rx_data receives the signal successfully and generates the end receiving signal done.

```
  always@ (posedge clk or negedge rst_n) begin
      if (! rst_n) begin
          rs232_t <= 1'b1;
          rs232_t1 <= 1'b1;
          rs232_t2 <= 1'b1;
      end
      else begin
          rs232_t <= rs232;
          rs232_t1 <= rs232_t;
          rs232_t2 <= rs232_t1;
      end
  end
```

In order to reduce metastability, multiple triggers are used in the program, and the data to be received is transmitted from rs232 to rs232_t2.

```
/*rx_data*/
  always@ (posedge clk or negedge rst_n) begin
      if (! rst_n)
          rx_data <= 'd0;
      else if(state)begin
          if(bit_flag) begin
          case(bit_cnt)
```

```
                    4'd1: rx_data [0] <= rs232_t2;
                    4'd2: rx_data[1] <= rs232_t2;
                    4'd3: rx_data[2] <= rs232_t2;
                          4'd4: rx_data[3] <= rs232_t2;
                    4'd5: rx_data[4] <= rs232_t2;
                    4'd6: rx_data[5] <= rs232_t2;
                    4'd7: rx_data[6] <= rs232_t2;
                    4'd8: rx_data[7] <= rs232_t2;
                    default:rx_data <= rx_data;
            endcase
            end
        end
        else
            rx_data <= rx_data;
    end
```

When the receiving flag bit_flag is high, the bit counter bit_cnt starts counting. When bit_cnt ranges from 1 to 8, rx_data receives eight-bit data of rs232_t2 from low to high.

3.2. Experimental Results

The design uses simulation software modelsim, clearly simulated UART send and receive data waveform.

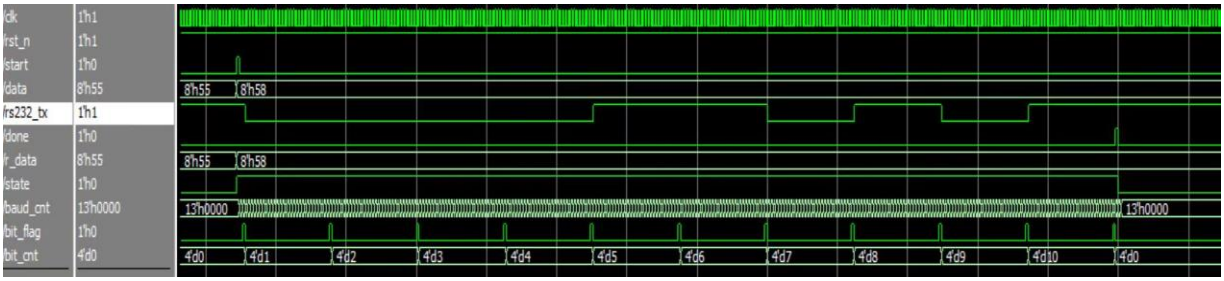

Fig. 6 Waveform of sending module

As can be seen from Figure 6, r_data is the parallel data to be sent and rs232_tx is the serial data that has been sent. r_data data is 58H (converted to binary is 01011000B). When the sending flag bit_flag is set to high level, bit_cnt is increased by 1. The sender rs232_tx according to the data of bit_cnt, in turn, send a start bit (low level), the data bits "0" "0" "0" "1" "1" "0" "1" and "0, ending a (high level). At this time, bit_cnt exactly counts to 10, and when bit_flag is high level, done appears high level, and state signal is set to low level, ending transmission.

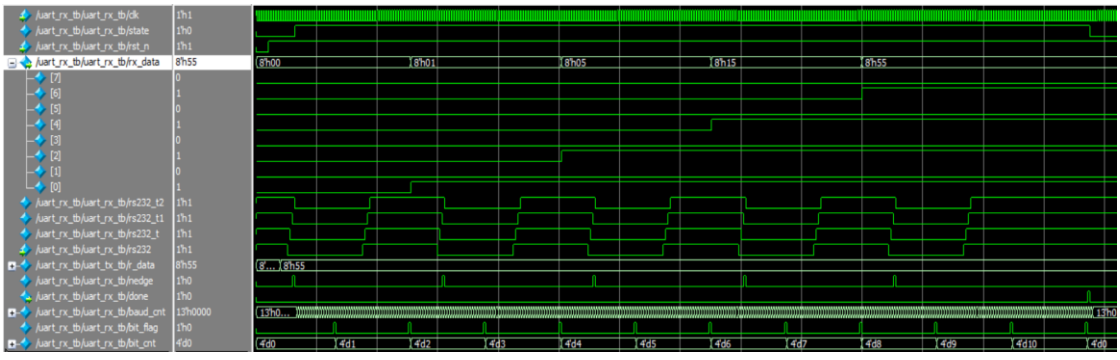

Fig. 7 Waveform of receiving module

As can be seen from Figure 7, rs232_t2 is the serial data to be received (directly calling the data of r_data in the sending module). rx_data is the received parallel data. rx_data on the receiving end does not receive the start and end bits, but receives eight data bits "1", "0", "1", "0", "1", "0", "1", and

"0" in sequence from low to high. The data equivalent of rs323_t2 is 55H (01010101B), which proves that the module design is correct.

4. Summary

This paper introduces the definition, function and module division of UART in detail, and verifies the feasibility of the program by running simulation. From the above experiments, this paper successfully built UART through Verilog HDL language, respectively simulated the sending and receiving modules, took into account the speed and accuracy, realized the high-speed and accurate information exchange between the computer and peripheral, and achieved the ideal result. This design achieves full-duplex transmission and reception, complete the design objectives, so that the data transmission process is efficient, clear, accurate and stable. This paper has certain reference value for the development of UART.

References

[1] R, Y., & V, R. M. Design and Verification of UART using System Verilog. International Journal of Engineering and Advanced Technology, 2020,9(5), 1208–1211.

[2] Fang, Y. Y., & Chen, X. J. Design and Simulation of UART Serial Communication Module Based on VHDL. 2011 3rd International Workshop on Intelligent Systems and Applications, 2011, 5873448.

[3] Han, X., & Kong, X. The Designing of Serial Communication Based on RS232. 2010 First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems, 2010, 80.

[4] Choi, J. I., Jain, M., Srinivasan, K., Levis, P., & Katti, S. Achieving single channel, full duplex wireless communication. Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking - MobiCom '10, 2010, 1859997.

[5] Kashyap, B., & Ravi, V. Universal Verification Methodology Based Verification of UART Protocol. Journal of Physics: Conference Series, 2020, 1716(1), 012040.

[6] Amrit P. Singh, Khushboo Gupta, & Juhee Mala. Determination of UART Receiver Baud Rate Tolerance. International Conference on Computer and Communication Technology, 2015, 265–270, 2818655.

[7] Wu, J., Ma, Y., Zhang, J., Kong, Y., Song, H., & Han, X. Research on metastability based on FPGA. 2009 9th International Conference on Electronic Measurement &Amp; Instruments,2009,5274693.

[8] HU, Z., ZHANG, J., & LUO, X. L. A Novel Design of Efficient Multi-channel UART Controller Based on FPGA. Chinese Journal of Aeronautics, 2007,20(1), 66–74.

[9] Thai, N. P., Ngoc, B. H., Duy, T. D., Truong, P. Q., & Phan, V. C. A novel multichannel UART design with FPGA-based implementation. International Journal of Computer Applications in Technology, 2021, 67(4), 358.

[10] Kamath, A., Mendez, T., Ramya, S., & Nayak, S. G. Design and Implementation of Power-Efficient FSM based UART. Journal of Physics: Conference Series, 2022, 2161(1), 012052.