Design of USB-UART Interface Converter and Its FPGA Implementation

Qingming Yi*,Min Shi,Senyuan Li
College of Information Science and Technology, Jinan University
Guangzhou, China
*: tyqm@jnu.edu.cn

***Abstract*—In this paper, we present a UART-USB interface converter design based on FPGA, which implements asynchronous serial communication protocol and USB protocols conversion. Design is implied by Hardware Description Language using modular design, it can be integrated into different SoC system. After Modelsim functional simulation and FPGA simulation with two computers, we verify the feasibility of the design.**

***Keywords*—Serial Communication; Protocol Converter; FPGA**

I. Introduction

Serial communication technology widely used in the field of industrial control, Aerospace and other fields. Such communication protocols appear earlier time, the principle is simple, mature technology. With RS-232, RS-422, RS-485 and so on with the same core controller - UART serial communication interface is the most common. These interfaces are basically the same as the interface drive circuit [1]. In many traditional industries, this kind of interface cannot be replaced.

But with the rapid development of computer technology, computer design in micro era, higher transmission speed, higher performance alternatives -- USB interface replaced the many traditional communication interface, so in the current computer, especially in portable computer is no longer mentioned for the serial port. At the same time, the computer as the main control equipment in industrial control, need to use the serial port as the transmission interface of the instrument or controlled equipment for communication [2].

To solve this problem, many manufacturers have introduced various types of RS232-USB, RS485-USB interface conversion chip. But these chips have low flexibility and can only implement a single interface, and cannot be integrated into the embedded SoC system. Therefore, it is of practical significance to design the UART-USB interface converter based on IP in the form of FPGA core.

II. UART Interface Design

A. *Asychronous serial communication protocol*

In asynchronous serial communication, the data line between the transmitter and the receiver is opposite to the two direction, it does not have a clock signal, the two sides do not use a common reference clock, It takes the basic unit of frame transmission, the transmission rate is also called the baud rate, In different interfaces, baud rate is different, Such as RS-232 75bps, the baud rate is: 50bps, 100bps, 150bps, 300BPS, 600bps, 1200BPS, 2400bps, 4800bps,9600bps,19200bps. The sender and recipient need to use the same transmission baud rate.

Between each bit in the frame (frame start bit, data bits, parity, stop bits) using the same baud rate for transmission, and no fixed time interval between frames [3]. A full data frame format is shown in the following Fig.1.

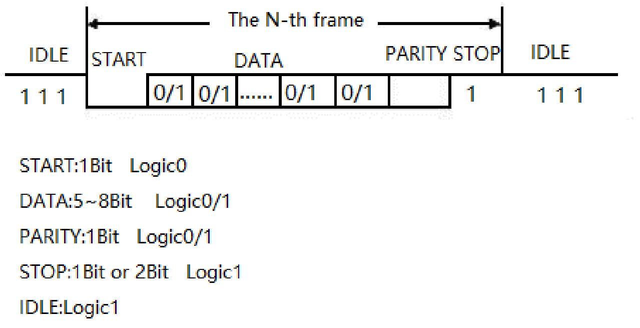


Fig. 1. Asynchronous serial communication data frame format.

B. *FPGA-based design of UART*

In this design, according to the different functions of UART, using TOP-DOWN design ideas, the entire UART is divided into two small modules were designed respectively: The baud rate generator respectively, a transmitting module and a receiving module. As shown in the following Fig.2.

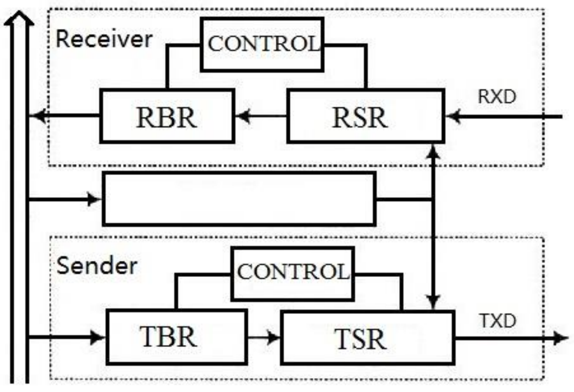


Fig. 2. UART internal modules.

1) baud rate generator:

The main function of the baud rate generator is an external clock crystal input divider to generate a clock frequency of the UART communication. Under normal circumstances, the size of the clock frequency is 16 times the baud rate. Functionally, the module is actually a divider for dividing the system clock input.

2) send module:

The transmission module contains the shift register, data register and the corresponding control logic. Implementation process: When the system is reset, TSR and TBR are empty, while the TSR empty flag (TSRE) is set to 1, TBR empty flag (TBRE) is set to 1, the data bus can be written into the TBR. When the data bus is written TBR, TBR is set to 0, indicating that there is data in the cache, At the same time, the data is assembled at this time, the control logic add a start bit, parity bit and stop bits according to the data frame format and the data, then assembled data frame input to the TSR. When the data frame is input to the shift register, TBR empty flag (TBRE) is set to 1, TSR empty flag (TSRE) is set to 0. Meanwhile, according to the baud rate (1/16 clock frequency), TSR will frame a bit output data a bit to TXD, when the transfer is complete, the TXD is set to 1, set TSRE 1, a data transmission completion of a data transfer wait for the next start.

3) receiving module:

Receiver module also includes a shift register, a data register and the corresponding control logic, Implementation process: When the system is reset, the receiver module constantly sampled on RXD when detected high level to low level transition after waiting seven baud generator output clock cycle, and then two consecutive samples (sample point just in case the level of the middle of two o'clock), two consecutive samples at this time to confirm receipt of a low level is not the burrs. After confirmation of receipt of the start bit and receive shift register (RSR) every 16 baud rate generator output clock sampling time and the receive shift register (RSR) shift a, followed by sampling to data bits, parity bit (if any), stop bits, and check data bits according to the parity bit,detect stop position whether to conform to the frame format, with the correct data frame, the received data register (RBR) shift register (RSR) data input and notify the bus to read data, complete a data receiving, waiting for the arrival of the next frame data.

III. USB DATA TRANSCEIVED DESIGN

A. USB protocol overview

According to the USB protocol, USB transfer will have a certain format several fields make up the packet type. The packet type is the basic unit of the host and the device to transmit data, such as SETUP transaction, IN transaction, OUT transaction, etc. These transactions form the four basic transfer types: Control Transaction, Interrupt Transaction, bulk transfer and synchronous Isochronous Transaction. In this design, according to the actual use of the device, no need to achieve synchronous transmission function.

USB package type consists of five parts, including: Synchronization field (SYNC), packet identification field (PID), data field (DATA), cyclic redundancy check field (CRC) and the end of the packet fields (EOP). Synchronization field is the starting field of most packages, then the packet identifier field, followed by a data field, along with a check of the correctness of the data field cyclic redundancy check field, the final field is end of the package [4] ,as shown in Fig.3.

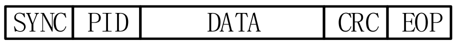


Fig. 3. USB field format.

The most important field is the package identifier, the package identifier field is used to distinguish between different types of packages, main types in the following Table 1.

TABLE I. MAIN TYPES OF PID

Category	*PID*	*Description*
Token	OUT	host-to-function transaction
	SOF	Start-of-Frame and frame number
	IN	function-to-host transaction
	SETUP	Host-to-function transaction for SETUP to a control pipe
Data	DATA0	Data packet PID even
	DATA1	Data packet PID odd
Hand-shake	ACK	Accepts error-free data packet
	NAK	Cannot accept data or cannot send data
	STALL	Endpoint is halted or a control pipe request is not supported

In USB data transfer, transaction processing the basic unit of communication bus, a transaction processing is composed of three categories of packet type (token packets, packet, handshake packet), corresponding to the three stages: phase token, data and state phase [5]. The following will be detailed analysis of three more important transaction processing:

In transaction is usually used by the host to read the data from device, shown as Fig.4.

a) When a host needs to read data from a device, a IN token packet is sent to the device address endpoint;

b) after receiving IN token packet from host, if the data can be read the device will send the data in the form of data packets, if the data is not ready, returns NAK handshake, if the endpoint is disabled returns STALL handshake ;

c) After receiving data from the device, if the data is correct, host will return ACK, otherwise returns no handshake.

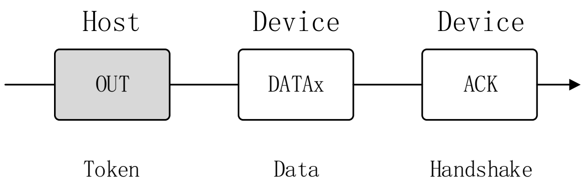


Fig. 4. A successful IN transaction.

OUT transaction is usually used by the host to write the data to device: (1)When a host needs to write data to a device, a OUT token packet is sent to the device address endpoint; (2)The host sends the data to the device in the form of a data packet; (3)After receiving the OUT token packet and packet from the host, if the received data is correct, it returns the ACK handshake packet. If the data is unable to receive temporarily it returns the NAK handshake packet, if the endpoint is stopped, then the STALL handshake packet is returned, shown as Fig.5.

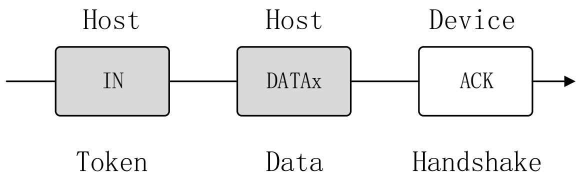


Fig. 5. A successful OUT transaction.

SETUP transaction is a special transmission with transmission direction from host to device, which the data format is special defined. We only use the setup transaction in the establish stage of control transfer: (1)In SETUP transactions, the host sends a SETUP token packet to the communication port of the device; (2)Host sends data packets with USB special definition format; (3)After receive SETUP token packet sent from the host, device must receive the next data packet, and may not return NAK handshake or STALL handshake, if correct, ACK handshake is returned, or not return handshake, shown as Fig.6.

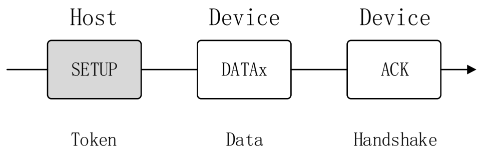


Fig. 6. A successful SETUP transaction.

Bulk transaction and interrupt transaction usually consists of a single IN/OUT transaction, and the control transaction consists of a SETUP transaction, an IN/OUT transaction and a transaction OUT/ IN transaction with different direction.

B. *Design of USB data transceiver*

USB data transceiver implements the protocol processing functions in this design, it can achieve the basic USB data transceiver functions, support for control transaction, interrupt transaction and bulk transaction. In this design, the USB data transceiver module is divided into physical layer module (PHY) and protocol layer module (CONTROL) module, as shown in the following Fig.7.

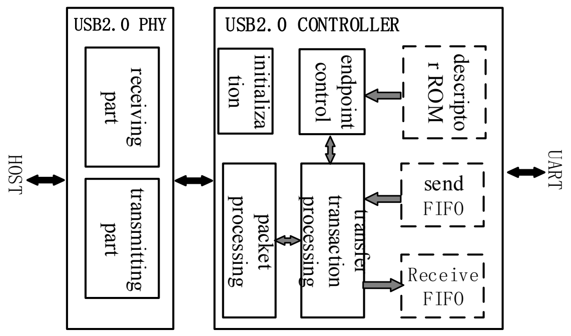


Fig. 7. Composition of USB data transceiver module.

USB physical layer module implements the conversion of USB signals. The module consists of two parts, namely the receiving part and transmitting part: Send part controlled by the state machine, send the 8-bit bit field data from UTMI interface sequentially and then add synchronization field (SYNC), parallel data into serial data, bit stuffing, NRZI encoding, add the end of packet (EOP), and finally send the data to the transceiver chip for data transmission. Similarly, the receiving part is controlled by the state machine, receive data from transceiver chip, synchronization field (SYNC) detection, NRZI decoding, bit unstuffing, serial data into parallel data, the end of packet (EOP) detection, in addition, due to the asynchronous communication USB no clock synchronization signal, and therefore this module also need to provide a signal receiving module locking and data recovery.

USB protocol layer main function modules: control physical layer module according UTMI interface protocol and send/ receive the original data; interpret the original data according to USB protocol. Analysis the host requirements, and data communication in accordance with the requirements of the host, which is the main function of the controller; to send data and received data storage, provide serial communication interface. this module is divided into initialization module, packet processing module, endpoint control module, transfer transaction processing module, descriptor ROM, and send and receive FIFO.

IV. FUNCTION SIMULATION AND ON BOARD TEST

After the completion of the two modules separately, combined two module and make a complete module. After functional simulation, program was downloaded to the FPGA development board, and tested with serial debugging assistant, shown as Fig.8.

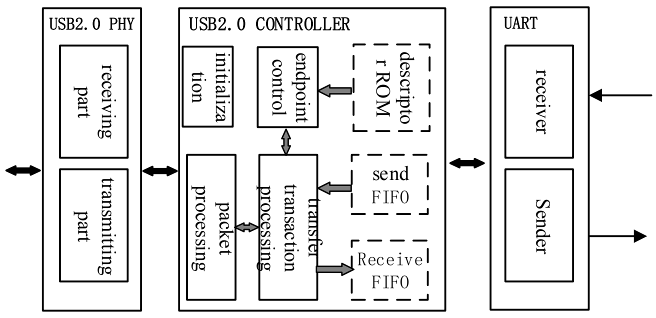


Fig. 8. Overall system components.

A. *Function simulation*

In the functional simulation, we need to build a test environment (testbench). In the test environment, we need to design some modules according to different functions: a control module, a incentive module, a monitoring module and a serial interconnection.

Incentive module: use for imitating the host control function, it can imitate the host behavior in the USB enumeration and USB communication process in sending data and receiving data, send out a variety of packet types of commands to the control module.

Control module: to achieve a specific function for sending a packet, the module convert the command into PID and data segment according to the commands of the incentive module, and add cyclic redundancy check field(CRC), encapsulated into a packet type. Then use the built-in physical layer module for adding synchronization field, NRZI encoding, end of packet field; and finally send data to USB device by two data buses (d POS and D NEG).

Monitoring module: monitor the data on the data bus (d_pos and d_neg), read the waveform signal information and translate it into a text message, and then output the text to the Transcript window in ModelSim SE 10.1a, which make it easy for analysis the bus signal information and find errors.

Serial connection: connect the transmit signal line and the receive signal line of UART.

A complete test environment is shown as Fig.9.

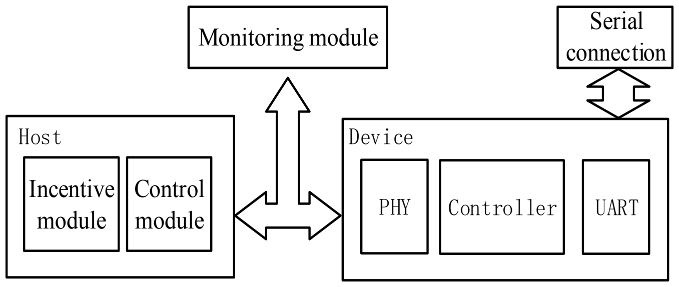
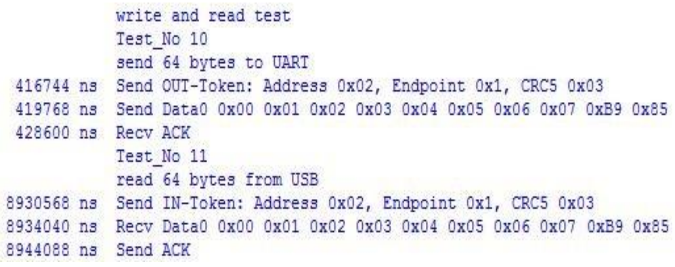


Fig. 9. Test bench.

When tested we use the host to send data, and then device the receive data, then the device transmit serial data. Because serial transmit is the input of the device, they will be received and stored in the receive FIFO, finally use the host to read the data. By comparing the received data with sent data in the host, we can verify that device is working properly. Functional simulation results are showed as Fig.10.

Use monitoring module for translating the specific content in d_pos and d_neg and get send data and receive data, as shown in Fig.11.



```
             write and read test
             Test_No 10
             send 64 bytes to UART
 416744 ns   Send OUT-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 419768 ns   Send Data0 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0xB9 0x85
 428600 ns   Recv ACK
             Test_No 11
             read 64 bytes from USB
8930568 ns   Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
8934040 ns   Recv Data0 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0xB9 0x85
8944088 ns   Send ACK
```

Fig. 11. Functional simulation results

Simulation map: after the completion of the device enumeration of UART-USB interface converter, host the data 0x00~0x07 sent to the device, and device using serial transmission mode to send out the data, due to send signal lines and receiving signal line Internet. Therefore, the device will received the same serial data, at the same time, the data stored in the device receive FIFO which, finally, the host reads the data equipment. By monitoring the output of the module to get the host to send and receive data, the data can be compared with the data, the transmission of the data and the received data is the same.

B. *Development board test*

After the completion of the system function simulation, build the hardware platform for the entire system for board level simulation, the system hardware test platform and the physical shown as Fig.12.

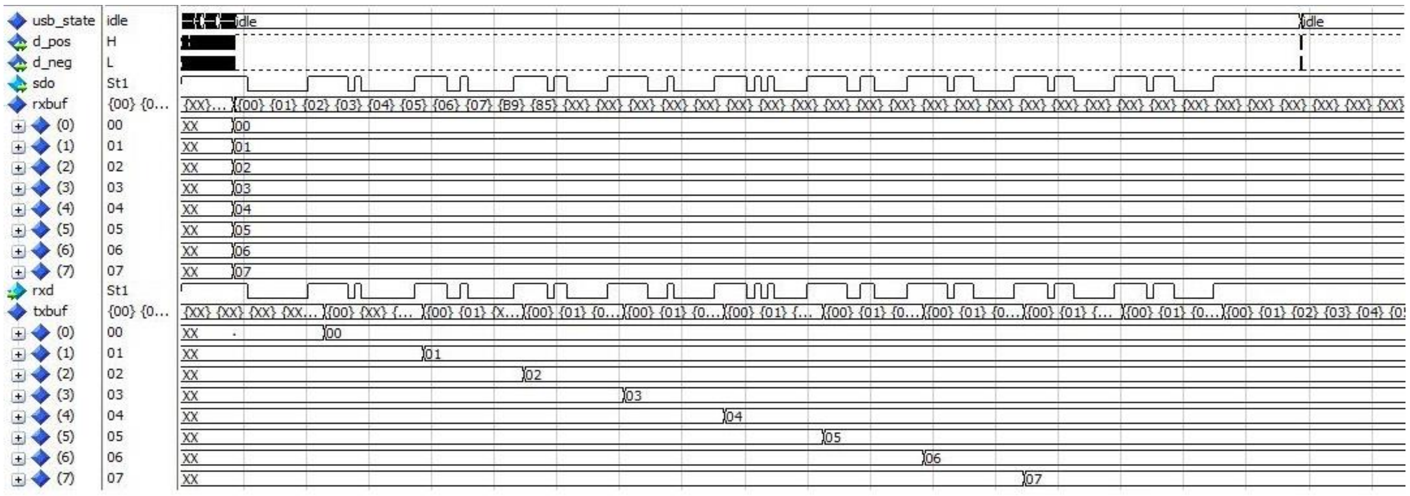


Fig. 10. Functional simulation results.

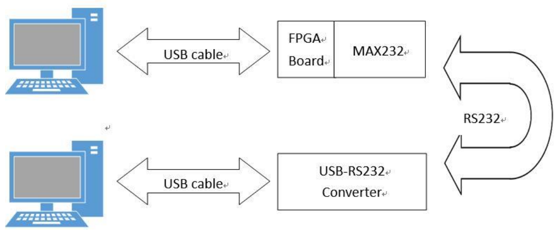


Fig. 12. Hardware testing platform.

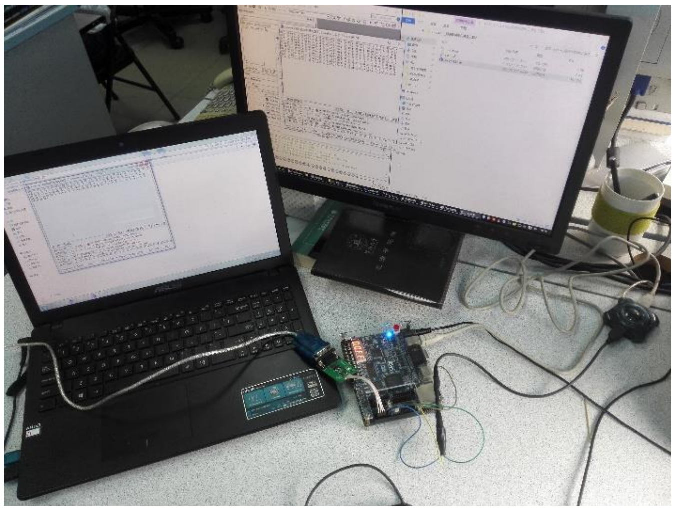


Fig. 13. Hardware.

Design used Terasic DE0 FPGA development board for test, The development board uses Altera EP3C16F484 as the main control chip [6]. Download the program to the development board, Sent serial data through one computer's serial debugging assistant, and then received by another computer, finally compare the data. The results showed that: sent and received data were exactly the same in two transmissions with opposite direction, the design's sending and receiving functions are working correctly, shown as Fig.13.

V. CONCLUTION

Asynchronous serial communication protocol and USB communication protocol is fully study in this research. A UART module and a USB data transceiver module were designed. Through the function simulation and board level test, verify the correctness of the design.

ACKNOWLEDGMENT

This paper is supported by the 2016 Guangzhou Science and Technology Project (201604010007).

REFERENCES

[1] Liu,Wen. Dongfang,Jiang. and Xincheng,Wang. 2010. Directly to The USB Programming of The USB - UART Converter Design. *Observation and Control Technology.* (2010), pp.81-84.

[2] Junxiang,Guo. and Ling Cao. 2007. RS-232C Interface and Its Application. *Electronic Instrumentation Customer.* (2007), pp.53-54.

[3] Xiansheng,Wang. 2014. Design and Application of UART Circuit Based on FPGA. Xi'an Electronic and Science University.

[4] Universal Serial Bus specification (Revision 2.0) [S/OL]. April 27, 2000,DOI= http://www.usb .org/developers/docs/usb20_docs/#usb20spec.

[5] Juxiong,Xiao. Tiecheng,Weng. and Zhongqing,Song. 2003. *USB Technology and Application Design.* Tsinghua university press.

[6] HuiLi. 2010. Logic Design and Implementation of IP USB2.0PHY Authentication System. Xi'an Electronic and Science University.