Design and Simulation of UART Serial Communication Module Based on VHDL

FANG Yi-yuan   CHEN Xue-jun

College of Electrical and Electronic Engineering
Shanghai University of Engineering Science
Shanghai, China
evafyy@hotmail.com

Abstract—**UART (Universal Asynchronous Receiver Transmitter) is a kind of serial communication protocol; mostly used for short-distance, low speed, low-cost data exchange between computer and peripherals. During the actual industrial production, sometimes we do not need the full functionality of UART, but simply integrate its core part. UART includes three kernel modules which are the baud rate generator, receiver and transmitter. The UART implemented with VHDL language can be integrated into the FPGA to achieve compact, stable and reliable data transmission. It's significant for the design of SOC. The simulation results with Quartus II are completely consistent with the UART protocol.**

Keywords- UART; asynchronous serial communication; VHDL; Quartus II; simulation

I. INTRODUCTION

Asynchronous serial communication has advantages of less transmission line, high reliability, and long transmission distance, therefore is widely used in data exchange between computer and peripherals. Asynchronous serial communication is usually implemented by Universal Asynchronous Receiver Transmitter (UART) [1]. UART allows full-duplex communication in serial link, thus has been widely used in the data communications and control system. In actual applications, usually only a few key features of UART are needed. Specific interface chip will cause waste of resources and increased cost. Particularly in the field of electronic design, SOC technology is recently becoming increasingly mature. This situation results in the requirement of realizing the whole system function in a single or a very few chips. Designers must integrate the similar function module into FPGA. This paper uses VHDL to implement the UART core functions and integrate them into a FPGA chip to achieve compact, stable and reliable data transmission, which effectively solves the above problem [2] [3].

Basic UART communication needs only two signal lines (RXD, TXD) to complete full-duplex data communication. TXD is the transmit side, the output of UART; RXD is the receiver, the input of UART. UART's basic features are: There are two states in the signal line, using logic 1 (high) and logic 0 (low) to distinguish respectively. For example, when the transmitter is idle, the data line is in the high logic state. Otherwise when a word is given to the UART for asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word [4].

After the Start Bit, the individual data bits of the word are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver "looks" at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent. Because asynchronous data are "self-synchronizing", if there are no data to transmit, the transmission line can be idle. The UART frame format is shown in Fig. 1.
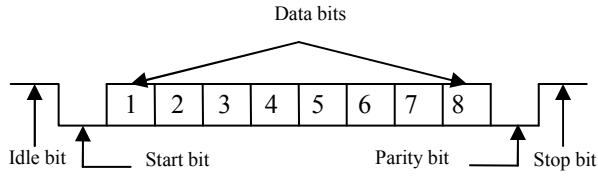
Figure 1.   UART Frame Format

II.   IMPLEMENTATION OF UART

In this paper, the top to bottom (Top to Down) design method is used. The UART serial communication module is divided into three sub-modules: the baud rate generator, receiver module and transmitter module, shown in Fig. 2. Therefore, the implementation of the UART communication module is actually the realization of the three sub-modules. The baud rate generator is used to produce a local clock signal which is much higher than the baud rate to control the UART receive and transmit; The UART receiver module is used to receive the serial signals at RXD, and convert them into parallel data; The UART transmit module converts the bytes into serial bits according to the basic frame format and transmits those bits through TXD.
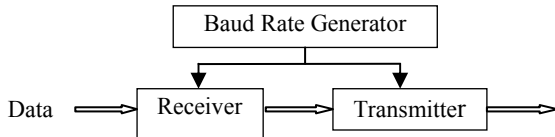

Figure 2.   UART Module

A.   Baud Rate Generator

Baud Rate Generator is actually a frequency divider. The baud rate frequency factor can be calculated according to a given system clock frequency (oscillator clock) and the requested baud rate. The calculated baud rate frequency factor is used as the divider factor. In this design, the frequency clock produced by the baud rate generator is not the baud rate clock, but 16 times the baud rate clock. The purpose is to precisely sample the asynchronous serial data at the receiver. Assume that the system clock is 32MHz, baud rate is 9600bps, and then the output clock frequency of baud rate generator should be 16 * 9600Hz. Therefore the frequency coefficient (M) of the baud rate generator is:

$$M = 32MHz/16*9600Hz = 208$$

When the UART receives serial data, it is very critical to determine where to sample the data information. The ideal time for sampling is at the middle point of each serial data bit. In this design, the receive clock frequency is designed to be 16 times the baud rate, therefore, each data width received by UART is 16 times the receive clock cycle.

B.   Receiver Module

During the UART reception, the serial data and the receiving clock are asynchronous, so it is very important to correctly determine the start bit of a frame data. The receiver module receives data from RXD pin. RXD jumps into logic 0 from logic 1 can be regarded as the beginning of a data frame. When the UART receiver module is reset, it has been waiting the RXD level to jump. The start bit is identified by detecting RXD level changes from high to low. In order to avoid the misjudgment of the start bit caused by noise, a start bit error detect function is added in this design, which requires the received low level in RXD at least over 50% of the baud rate to be able to determine the start bit arrives. Since the receive clock frequency is 16 times the baud rate in the design, the RXD low level lasts at least 8 receiving clock cycles is considered start bit arrives. Once the start bit been identified, from the next bit, begin to count the rising edge of the baud clock, and sample RXD when counting. Each sampled value of the logic level is deposited in the register rbuf [7, 0] by order. When the count equals 8, all the data bits are surely received, also the 8 serial bits are converted into a byte parallel data..

The serial receiver module includes receiving, serial and parallel transform, and receive caching, etc. In this paper we use finite state machine to design, shown in Fig. 3.
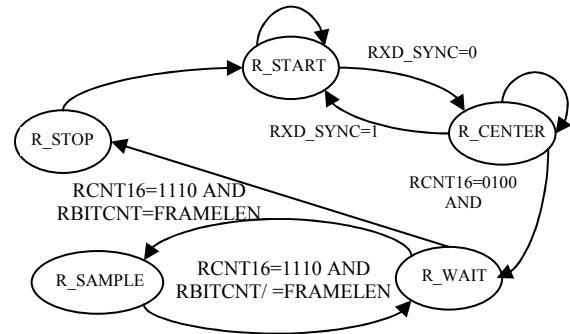

Figure 3.   UART Receiver State Machine

The state machine includes five states: R_START (waiting for the start bit), R_CENTER (find midpoint), R_WAIT (waiting for the sampling), R_SAMPLE (sampling), and R_STOP (receiving stop bit).

R_START Status: When the UART receiver is reset, the receiver state machine will be in this state. In this state, the state machine has been waiting for the RXD level to jump over from logic 1 to logic 0, i.e. the start bit. This alerts the beginning of a new data frame. Once the start bit is identified, the state machine will be transferred to R_CENTER state. In Fig. 3, RXD_SYNC is a synchronization signal of RXD. Because when sampling logic 1 or logic 0, we do not want the detected signal to be unstable. So we do not directly detect RXD signal, but detect the synchronization signal RXD_SYNC.

R_CENTER Status: For asynchronous serial signal, in order to detect the correct signal each time, and minimize the total error in the later data bits detection. Obviously, it is the most ideal to detect at the middle of each bit. In this state, the task is to find the midpoint of each bit through the start bit. The method is by counting the number of bclkr (the receiving clock frequency generated by the baud rate generator) (RCNT16 is

the counter of bclkr). In addition, the start bit detected in the R_START may not be a really start bit, it may be an occasional interference sharp pulse (negative pulse). This interference pulse cycle is very short. Therefore, the signal that maintains logic 0 over 1 / 4 bit time must be a start bit.

R_WAIT Status: When the state machine is in this state, waiting for counting bclkr to 15, then entering into R_SAMPLE to sample the data bits at the 16th bclkr. At the same time determining whether the collected data bit length has reached the data frame length (FRAMELEN). If reaches, it means the stop bits arrives. The FRAMELEN is modifiable in the design (using the Generic). In this design it is 8, which corresponds to the 8-bit data format of UART.

R_SAMPLE Status: Data bit sampling. After sampling the state machine transfers to R_WAIT state, waits for the arrival of the next start bit.

R_STOP Status: Stop bit is either 1 or 1.5, or 2. State machine doesn't detect RXD in R_STOP, but output frame receiving done signal (REC_DONE <= '1 '). After the stop bit, state machine turns back to R_START state, waiting for the next frame start bit.

C. Transmit Module

The function of transmit module is to convert the sending 8-bit parallel data into serial data, adds start bit at the head of the data as well as the parity and stop bits at the end of the data. When the UART transmit module is reset by the reset signal, the transmit module immediately enters the ready state to send. In this state, the 8-bit parallel data is read into the register txdbuf [7: 0]. The transmitter only needs to output 1 bit every 16 bclkt (the transmitting clock frequency generated by the baud rate generator) cycles. The order follows 1 start bit, 8 data bits, 1 parity bit and 1 stop bit. The parity bit is determined according to the number of logic 1 in 8 data bits. Then the parity bit is output. Finally, logic 1 is output as the stop bit. Fig. 4 shows the transmit module state diagram.
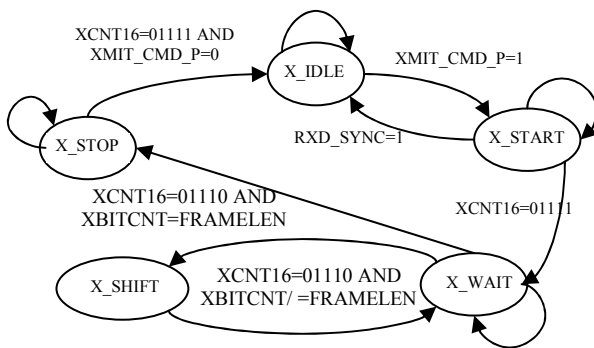

Figure 4. Transmit Module State Diagram

This state machine has 5 states: X_IDLE (free), X_START (start bit), X_WAIT (shift to wait), X_SHIFT (shift), X_STOP (stop bit).

X_IDLE Status: When the UART is reset, the state machine will be in this state. In this state, the UART transmitter has been waiting a data frame sending command

XMIT_CMD. XMIT_CMD_P is a processed signal of XMIT_CMD, which is a short pulse signal. Since XMIT_CMD is an external signal, outside FPGA, its pulse width is unable to be limited. If XMIT_CMD is valid, it is still valid after sending one UART data frame. Then the UART transmitter will think by mistake that a new data transmit command has arrived, and once again start the frame transmit. Obviously the frame transmit is wrong. Here we limit the pulse width of XMIT_CMD. XMIT_CMD_P is its processed signal. When XMIT_CMD_P = '1 ', the state machine transferred to X_START, get ready to send a start bit.

X_START Status: In this state, sends a logic 0 signal to the TXD for one bit time width, the start bit. Then the state machine transferred to X_WAIT state. XCNT16 is the counter of bclkt.

X_WAIT Status: Similar with the R_WAIT of UART receive state machine.

X_SHIFT Status: In this state, the state machine realizes the parallel to serial conversion of outgoing data. Then immediately return to X_WAIT state.

X_STOP Status: Stop bit transmit state. When the data frame transmit is completed, the state machine transferred to this state, and sends 16 bclkt cycle logic 1 signal, that is, 1 stop bit. The state machine turns back to X_IDLE state after sending the stop bit, and waits for another data frame transmit command.

III. SIMULATION OF MODULES

The simulation software is Quartus II. And the selected device is Altera's Cyclone II FPGA: EP2C5F256C6.

A. Baud Rate Generator Simulation

During simulation, the system clock frequency is set to 32MHz, and baud rate is set to 9600bps. Therefore the receiving sampling clock frequency generated by the baud rate generator is 153600Hz, which is 16 times of the baud rate. Thus the frequency coefficient of baud rate generator can be calculated, which equals 208. Fig. 5 shows the simulation result of baud rate generator. The simulation report shows that this module uses 42 logic elements（<1%）, 33 registers (2%), and meets timing requirement.
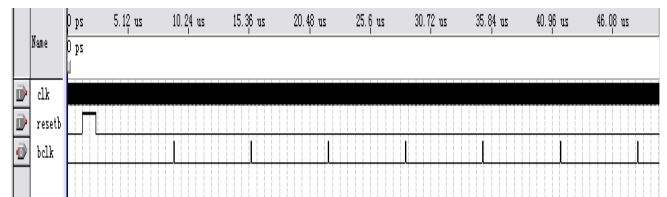

Figure 5. Simulation Result of Baud Rate Generator

B. Receiver Simulation

During receiver simulation, the receiving sampling clock frequency generated by the baud rate generator is set to 153600 Hz, UART receiving baud rate is set to 9600bps. The input sequence is: 00110110001, including the start bit 0, parity bit 0 and 1 stop bit. The received data is stored into the register rbuf.

Fig. 6 shows the receiver module simulation diagram. The figure shows that the data in rbuf from high to low is 00110110, which is just the part of data bits of UART frame.
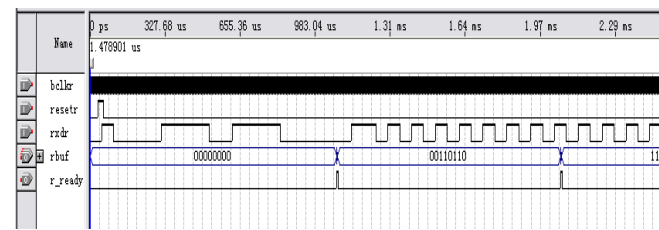

Figure 6.   Receiver Simulation Diagram

C. Transmitter Simulation

During transmitter simulation, the sending clock frequency generated by the baud rate generator is set to 153600 Hz, and UART transmitting baud rate is set to 9600bps. Fig. 7 shows the transmitter module simulation diagram. The simulation report shows that this module uses 78 logic elements（<1%, 13 pins (4%), and meets timing requirement.
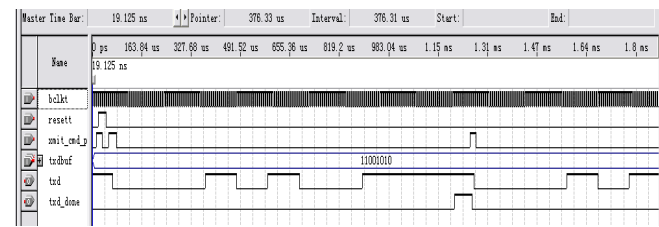

Figure 7   Transmitter Simulation Diagram

D. RTL of Top File

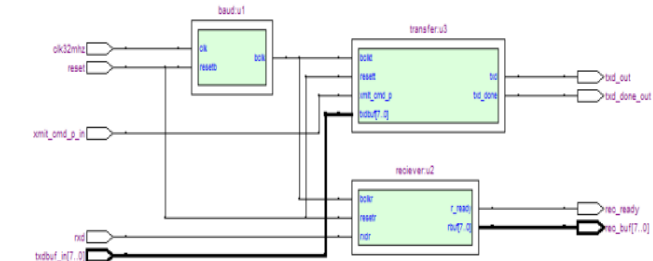Fig. 8 shows the RTL of UART Top File. It includes the baud rate generator, receiver, and transmitter modules.

Figure 8.   RTL of Top File

IV.   CONCLUSION

This design uses VHDL as design language to achieve the modules of UART. Using Quartus II software, Altera's Cyclone series FPGA chip EP2C5F256C6 to complete simulation and test. The results are stable and reliable. The design has great flexibility, high integration, with some reference value. Especially in the field of electronic design, where SOC technology has recently become increasingly mature, this design shows great significance.

REFERENCES

[1]   Zou,Jie Yang,Jianning。 Design and Realization of UART Controller Bas ed on FPGA

[2]   Liakot Ali , Roslina Sidek , Ishak Aris , Alauddin Mohd. Ali , Bambang Sunaryo Suparjo. Design of a micro - UART for SoC application [J].In: Computers and Electrical Engineering 30 (2004) 257– 268.

[3]   HU Hua, BAI Feng-e. Design and Simulation of UART Serial Communication Module Based on Verilog -HDL[J]. J ISUANJ I YU XIANDA IHUA 2008 Vol. 8

[4]   Frank Durda Serial and UART Tutorial. uhclem@FreeBSD.org