© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

A Review Paper on Design and Simulation of UART for Serial Communication

Vibhu Chinmay, Shubham Sachdeva

Student (B.Tech 5th sem) Department of Electronics and Computers Engineering
Dronacharya College of Engineering, Gurgaon-123506, India

Abstract: **In this paper, an attempt has been made to review the design of UART (universal asynchronous receiver transmitter) based on VHDL. UART is used for serial communication. This paper presents implementation of UART with different methods. Different techniques which were using with UART are discussed for reliable data transmission. The proposed work suggests application Design using UART. In general a UART operated with specific baud rate. As UART is consider as a low speed, low cost data exchange between computer and peripherals.To overcome the problem of low speed data transmission, a 16 bit UART is proposed in this paper. It works on 9600bps baud rate. This will result in increased the speed of UART. Whole design is simulated with Xilinx ISE8.2i software and results are completely consistent with UART protocol.**

.

Keywords: UART, Serial Communication, Baud Rate, Xilinx

I. INTRODUCTION

To meet the modern operation microcontroller and digital signal processor we need desired system performance. But in actual process, it is very difficult to attain desired result, since it depends on various factors. Communication is vital factor which affect the performance of system.

A universal asynchronous receive/transmit (UART) is an integrated circuit which plays the most important role in serial communication. It handles the conversion between serial and parallel data. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible. It contains a parallel-to serial converter for data transmitted from the computer and a serial to parallel converter for data coming in via the serial line. The UART also has a buffer for temporarily storing data from high speed transmissions. In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media and to regulate the flow of data in the event that the remote device is not prepared to accept more data. UART must have a larger internal buffer to store data coming from the modem until the CPU has time to process it. The UART serial communication module is divided into three sub-modules: the baud rate generator, receiver module and transmitter module 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.

A. Serial transmission

Serial transmission is used in transmitting a bit in UART. Serial data transmission is a form of data transmission where bits of characters are sent one at a time along a communication path. Serial data transmission travel over a single wire in one direction.Two types of serial data transmission are there:

1) Synchronous serial communication: Synchronous serial communication is a serial communication protocol where data is sent in a continuous stream at a constant rate.Synchronous Communication requires that the clocks in the transmitting and receiving devices are synchronized, running at the same rate so the receiver can sample the signal at the same time intervals used by transmitter. No start or stop bits are required. In synchronous communication data is not sent in individual bytes, but as frames of large data blocks as shown in Fig. 1.

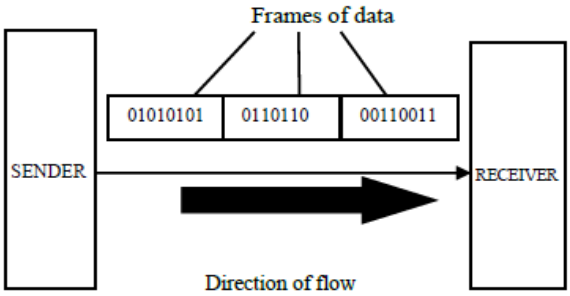


Fig. 1 Synchronous serial transmission

2) Asynchronous serial transmission:Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communications to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters. As shown in Fig. 2, one start bit and stop bit are added to the whole byte.

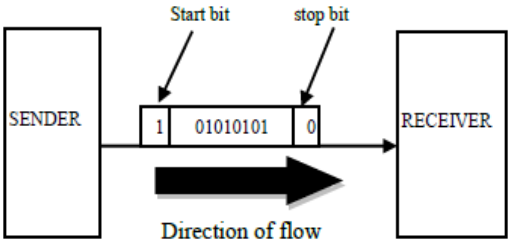


Fig. 2 Asynchronous serial transmission

B. *Use of HDL*

Just as software designers use high level languages (HLL) to express the algorithms in terms of language statements, digital hard-ware designers use hardware description languages (HDL) to describe the system they are designing. Although HDL's were originated as a medium of precise yet concise description of digital hardware, they have found a variety of applications such as generating user manuals, teaching logic design, acting as an input medium for an automatic design system, VHDL and Verilog are widely used HDL's. VHDL is a strongly and richly typed language. Derived from ada programming language, its language requirements make it moreverbose than Verilog whereas Verilog is a weekly and limited typed language. Its heritage can be traced to C programming language and an older HDL called Hilo. We are using VHDL as our programming language.

C. *Technology used*

1) Xilinx ISE:Xilinx ISE (integrated software environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Xilinx ISE 8.2i is a version of ISE, which we are used in our project for simulation.

II. OPERATION OF UART

A. *UART block diagram*

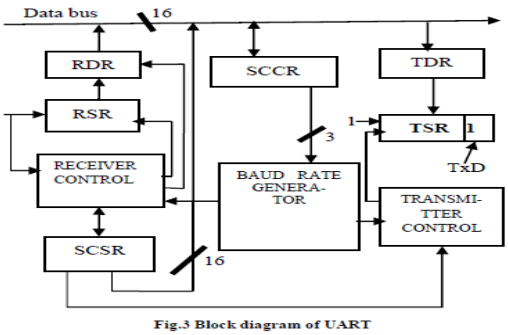


Fig.3 Block diagram of UART

As shown in Fig. 3, UART includes three kernel modules, the Transmitter, Receiver and the baud rate generator. Each and every module is responsible for its own task. Failure in any one of module affects the overall Output of UART.It also consists of six different registers, each used for their specific purpose.

B.UART Registers

In this paper 16 bit registers are used in place of 8 bit registers. Different registers used in UART are discussed below:

***1) RSR*:** Receiver Shift Register, it is a shift register used at the receiver end of UART .It receive the bits sequentially from RxD (receiving data pin) and shift them to right, one at each bit clock. As shown in Fig.4 serial in parallel out shift register is used in receiver. d15 parallel out d0Serial in.

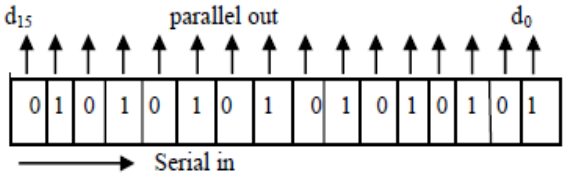


Fig.4 serial in parallel out shift register

***2) RDR*:** Receiver Data Register, it receives data from RSR and whatever data stored in RDR is placed on the data bus.

***3) TDR*:** Transmit Data Register, it receives bytes of data from data bus to be transmitted and transfer it in TSR.

***4) TSR*:** Transmit Shift Register, it is a shift register used at transmitter side it is used to transmit data bitwise by shifting each bit to right. Parallel in serial out shift register is used as shown in Fig.5 d15 parallel in d0Serial out.

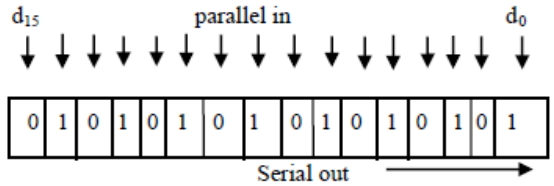


Fig.5. parallel in serial out shift register

5) SCCR:Serial Communications Control Register, this register is used for controlling the UART. The lower 3 bits in this register are used to select baud rate. And the first two bits i.e. TIE (transmit interrupt enable) and RIE (receiver interrupt enable), are interrupts signals and sets in SCCR whenever UART receiver and transmitter needs attention.

6) SCSR:Serial Communications Status Register, it contains the information about the UART's condition or state. It contains flags (TDRE and RDRF) to show the status of different registers. Its last two bits are OE (overrun error) and FE (framing error) which sets if any error occurs in the transmission. An overrun error occurs when the receiver cannot process the character that just came in before the next one arrives. And a framing error occurs when the designated "start" and "stop" bits are not valid.

As the "start bit" is used to identify the begging of an incoming character, it acts as a reference for the remaining bits. If the data line is not in the expected idle state when the "stop" bit is expected, a framing error will occur.

C. UART Flags

Two flags are used in UART. They are:

***1) TDRE Transmit Data Register Empty*:** This flags shows the status of transmit data register. If TDR is empty, then this flag is set in SCCR.

***2) RDRF Receive Data Register Full*:** This flags shows the status of RDR (receive data register).When all the bits from RDR are loaded into RSR, RDRF flag is set.

D. Operation of UART transmitter

Transmitter operation starts only when TDRE flag is set in SCSR register. As soon as data is deposited in shift register after completion of the previous character, the UART hardware generates a start bit, shifts the required number of bits out to the line and appends the stop bit. Since transmission of a single

character may takes a long time relative to CPU speeds, the UART will shows the TDRE flag busy so that the host system does not deposited a new character for transmission until the previous one has been completed. And when the transmission is completed again the TDRE flag is set in SCSR register, this indicated that transmitter is again ready for transmission.

E. Operation of UART Receiver

The receiver tests the state of the incoming signal on each clock pulse looking for the beginning of the start bit.

As soon as the start bit is detected, it reads the remaining bits serially and shifts them into RSR. When all the data

bits and stop bits are received, the RSR is loaded into RDR and RDRF flag is set in SCSR register. And RDR reads the data and cleared the RDRF flag.

F. Operation of Clock Divider

Three bits in the SCCR are used to select any of the eight-baud rates. In this paper we assumed that the system clock is 8 MHz and the required baud rates are 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400. Therefore, the maximum BclkX8 frequency needed is 38400*8 = 307200. To get this frequency, the system clock has to be divided by factor of 8 MHz/307200 = 26.04167. Since division by an integer is only possible, a small amount of error in baud rate is generated and is accepted.

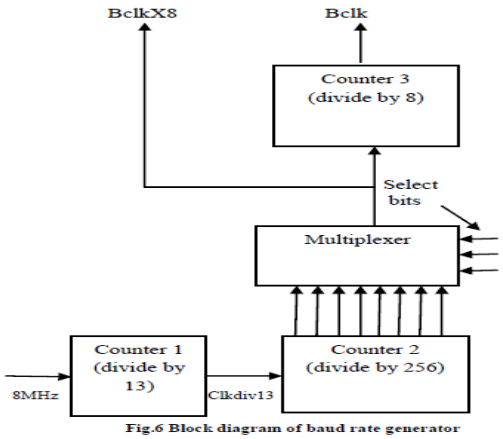


Fig.6 Block diagram of baud rate generator

Fig. 6 shows the block diagram for the baud rate generator. Using a counter, the 8 MHz system clock is first divided by 13. The output of this counter goes to an 8-bit binary counter. The output of the flip-flops in this counter corresponds to divide by 2, divide by 4 and so on up to divide by 256. One of these outputs is selected by the multiplexer. The MUX selects inputs coming from the lower 3 bits of the SCCR. The MUX output corresponds to BclkX8, which is further divide by 8 to give Bclk.

III. SIMULATION OUTPUT

The results obtained after running the simulation process for the UART are presented. Simulations are required to ensure that the design works according to the intended application. In this paper Simulation is performed by Xilinx ISE8.2i software. The results obtained are as follows:

A.UART Transmitter Simulation output

The Test Bench waveform and simulation results for the input "0101010101010101" are as shown in Fig.7.

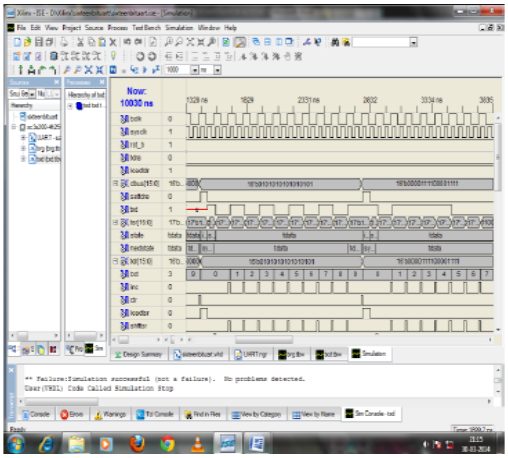
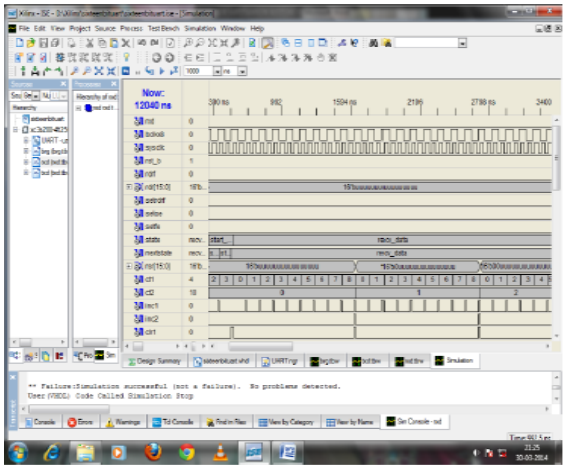


Fig. 7 Simulated output of transmitter

B.UART Receiver Simulation output

The Test Bench waveform and simulation results for receiver inputs of "0101010101010101" are as shown in Fig. 8.



IV. RESULT

As 1 byte contains 8 bit data in general, and here we are sending 16 bits in 1 byte. So if baud rate is 9600bps than by using 16 bit data we are able to send 16,384 bits paper second as compare to 8 bit data where we were only able to send 8,192 bits per second of time. So here we are able to send more number of bits per second of time and this will increased the speed of UART.

V. CONCLUSION

In this paper, we proposed a design of UART. It internally consists of transmitter, receiver and baud rate generator. The design is successfully simulated using Xilinx ISE 8.2i software. The results are stable and reliable and show the correct functionality. Hence, we can improve the speed of UART by sending 16 bits per second of time. But by sending 16 bit ,it become more complex to count number of clock bit per unit time, so this can be overcome by using multichannel UART in future.

REFERENCES

1. FANG Yi-Yuan CHEN Xue-Jun, “Design and Simulation of UART Serial Communication Module Based on VHDL” Shanghai University of Engineering Science, 2011

2. Amanpreet Kaur, Amandeep Kaur, “An approach for designing a universal asynchronous receiver transmitter (UART)”, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 3, May-Jun 2012, pp.2305-2311

3.en.wikipedia.org/wiki/synchronous_serial_communication.

4. Cowley, John (2007) communication and networking, An introduction, Springer, ISBN 9781846286452

5. www.pccompi.com

6. Shiva, S.G University of Alabama in Huntsville, AL, proceedings of the IEEE (volume 67, Issue: 12)

7. Comparison of VHDL, Verilog and System Verilog, by StephenBailey, www.model.com

8. C. H. Roth, “Digital System Design Using VHDL", PWSPublishing Company, 2008