International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.04,
April-2014,
Pages:0716-0719

www.semargroup.org,
www.ijsetr.com

Design and Implementation of UART Module using FIFO Based on Verilog HDL

SURAJ PRAKASH AHIRWAR[1], D.S.AJNAR[2], P.K.JAIN[3]

[1]PG Scholar, Dept of Electronic & Instrumentation, S.G.S.I.T.S, Indore, India,
Email: surajahirwar111@gmail.com.

[2]Assoc Prof, Dept of Electronic & Instrumentation, S.G.S.I.T.S, Indore, India.

[3]Assoc Prof, Dept of Electronic & Instrumentation, S.G.S.I.T.S, Indore, India.

Abstract: Universal Asynchronous Receiver Transmitter (UART) is a popular and widely used device for data communication in the field of Telecommunication. UART is a kind of serial communication protocol; mostly used for short distance, low speed, low cost data exchange between computer and peripherals. In this paper, we present the design of UART modules based on Verilog HDL. The design consists of the transmitter, receiver, baud rate generator and asynchronous FIFO (First in First Out) buffer. The design is synthesized in Verilog HDL and reliability of the Verilog HDL implementation of UART is verified by simulated waveforms.

Keywords: UART, FIFO, Transmitter, Receiver, Baud Rate Generator.

I. INTRODUCTION

Universal Asynchronous Receiver Transmitter is an integrated circuit, which is used for transmitting and receiving data asynchronously via the serial port on the computer. 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. UART incorporates the transmitter, receiver, baud rate generator and FIFO. Baud rate generator is used to provide clock to transmitter, receiver and FIFO. First-In-First-Out (FIFO) is used at both transmitter and receiver end to store high speed incoming data temporarily to prevent data loss. The RS-232 is applied to serial data communication as a standard to be compiled with. Besides RS-232, RS-422 and RS-485 standards have been commonly applied to UART chip nowadays. These standards offer more reliable communication over much longer distances compared to RS-232. 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).

The advantages of UART systems are the simplicity of interconnection wiring and character transmission formats. UART allows full-duplex communication in serial link, thus has been widely used in data communication 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.

II. UART PROTOCOL

A. UART Data Transmission Protocol

UART transmit and receive data format shown in Figure 1, usually include start bit, data bit, parity bit, stop bit and idle state. Start bit is the beginning of data transmission. When the transmitter sends a character data, a logic "0" signal is firstly send, which is the start bit, and the time width is a baud rate clock cycle. Start bit is followed by data bits, which are usually from 5 to 8 bits. The data bit from the least significant bit (LSB) begin to send. The data bits can be parity bit, which can be odd or even parity, as well as no parity bit. The parity bit or data bit (when no parity bit) is followed by stop bits, which are logic "1" signal containing 1, 1.5 or 2 bits. Stop bits are the end of a data. Idle state is a logic '1'.This data frame format is adopted by the start bit and stop bit to achieve character synchronization. UART has an internal configuration register, in which user can set the data bits, whether there is parity bit, as well as the type of parity and stop bits.

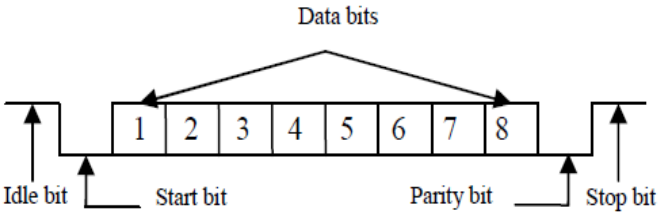

Figure 1: UART Frame Format.

A. Baud Rate Generator

The Baud Rate Generator is used to produce a local clock signal which is much higher than the baud rate to control the UART receiver and transmitter. Baud Rate Generator is actually a kind of frequency divider. The Baud rate, frequency factor can be calculated according to a given system clock frequency and the required baud rate. Then the baud rate, frequency factor, calculated is used as dividing factor. In this UART we will apply the synchronized clock signal to both transmitter and the receiver. The clock signal applied to the receiver is 16 times to that of the transmitter.

B. Receiver

All operations of the UART hardware are controlled by a clock signal which runs at a multiple (say, 16) of the data rate - each data bit is as long as 16 clock pulses. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid and signals the start of a new character. If not, the spurious pulse is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After the required number of bit periods for the character length (5 to 8 bits, typically) have elapsed, the contents of the shift register is made available (in parallel fashion) to the receiving system. The UART will set a flag indicating new data is available, and may also generate a processor interrupt to request that the host processor to transfer the received data. In some common types of UART, a small first-in, first-out FIFO buffer memory is inserted between the receiver shift register and the host system interface. This allows the host processor more time to handle an interrupt from the UART and prevents loss of received data at high rates.

C. Transmitter

Transmission operation is simpler since it is under the control of the transmitting system. As soon as data is deposited in the shift register after completion of the previous character, the UART hardware generates a start bit, shifts the required number of data bits out to the line, generates and appends the parity bit (if used), and appends the stop bits. Since transmission of a single character may take a long time relative to CPU speeds, the UART will maintain a flag showing busy status so that the host system does not deposit a new character for transmission until the previous one has been completed; this may also be done with an interrupt. Since full-duplex operation requires characters to be sent and received at the same time, practical UARTs use two different shift registers for transmitted characters and received characters.

D. First-In-First-Out Buffer

First-In-First-Out (FIFO) is used at both transmitter and receiver end to store high speed incoming data temporarily to prevent data loss. FIFO is implemented as a queue structure. It has a fixed length. If FIFO is empty or is not filled fully then only data can be written in it. If FIFO is full, it sends a signal "FULL" to the transmitter and receiver at respective end. If it is empty, then it sends signal "Empty" to the transmitter and receiver at respective end. FIFO is used for synchronization purpose between transmitter and CPU at one end and also between receiver and CPU at other end to prevent the loss of data if speed of CPU doesn't match with the transmission between transmitter and receiver.

III. UART MODULE

The UART module that we have designed consists of five parts namely

1. "Receiver", which takes in the serial data (as a frame) coming through the 'Data Frame' line, retrieves the actual data and converts to parallel form (usually as a byte).
2. "Transmitter", which does the opposite function of the "Receiver" module and transmits the frame through the 'Data Frame' line.
3. "Baud Rate Generator", which generates a clock which occurs 16-times (the default over-sampling rate) in one bit-time period.
4. "TX-FIFO", which stores temporarily the bytes (that usually comes from a faster processor) to send, as the sending process takes some time.
5. "RX-FIFO", which is the replica of the "TX-FIFO" module used to store the received bytes temporarily such that the processor may read them at its own pace.

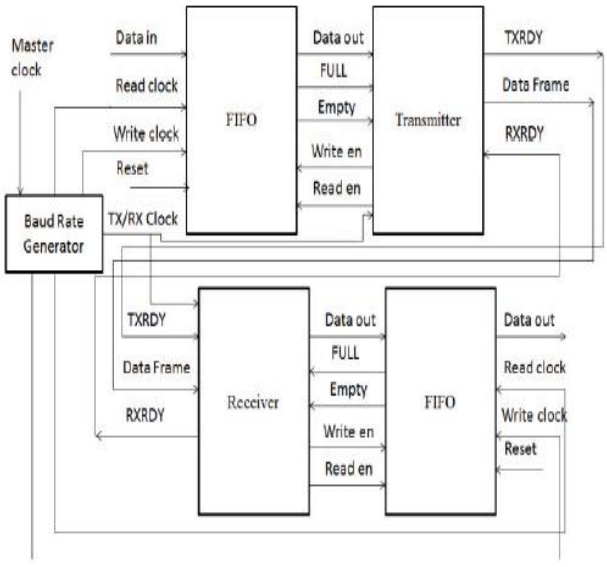

Figure 2: UART Module.

IV. RESULTS

We have simulated each and every part of our module separately in 'Xilinx-ISE simulator'. During simulation the system clock is set to 50 MHz and baud rate is set to

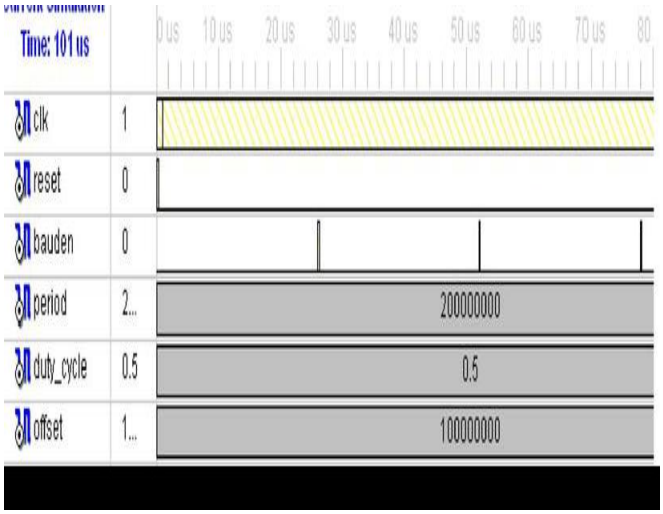

Figure 3: Simulation result for the sub module 'Baud Rate Generator'.

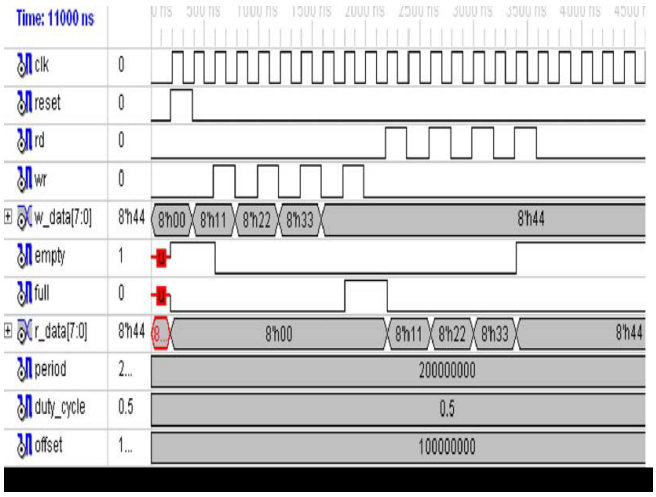

Figure 4: Simulation result for the sub module 'FIFO'.

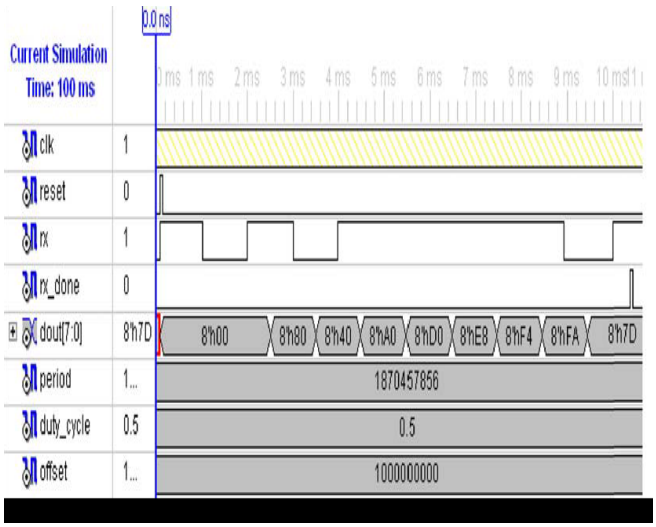

Figure 5: Simulation result for the sub module 'Receiver'.

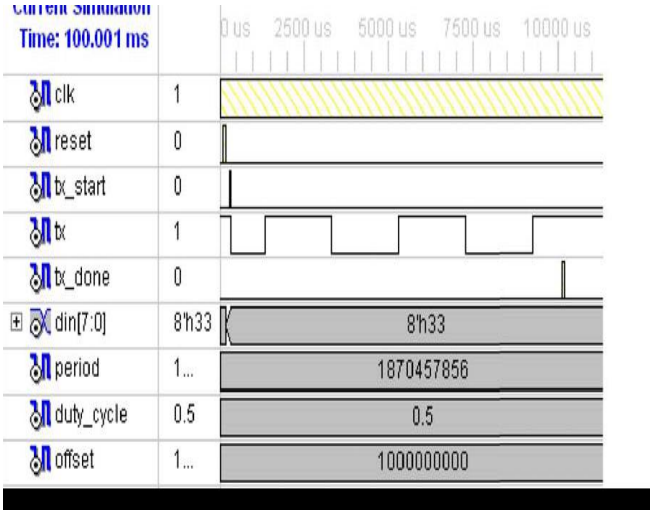

Figure 6: Simulation result for the sub module 'Transmitter.

9600bps and the selected device is Spartan 3E FPGA. The simulation results for the 'Baud Rate Generator', 'FIFO', 'Receiver', and 'Transmitter' sub modules are shown in the figures 3,4,5,6 respectively.

V. TESTING

The complete core was also tested by connecting it with the PC by a RS232 cable after implementing a test circuit incorporating our core in Xilinx Spartan-3E starter kit. The test circuit does have an incremented that takes one parallel data stored in the RX-FIFO after getting received by the UART module, increments it by one and then gives it back to the TX-FIFO which is then transmitted serially by the module. The reading from the FIFO, incrementing and writing to the FIFO is done one character at a time when a switch (connected with the test circuit through a debouncer

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	317	11,776	2%	
Number of 4 input LUTs	383	11,776	3%	
Logic Distribution				
Number of occupied Slices	322	5,888	5%	
Number of Slices containing only related logic	322	322	100%	
Number of Slices containing unrelated logic	0	322	0%	
Total Number of 4 input LUTs	507	11,776	4%	
Number used as logic	383			
Number used as a route-thru	124			
Number of bonded IOBs	7	372	1%	
Number of BUFGMUXs	1	24	4%	

Figure 7: Device resource utilized.

otherwise multiple reading and writing of NULL character could produce erroneous result) is pressed. Here we first typed-in 4 characters(displayed locally in the Hyper Terminal window) which are stored in the FIFO, then pressed the assigned switch 4 times which returns the stored characters incremented by one and are displayed again in the HyperTerminal(as shown in figure 7).

VI. CONCLUSION

UART using FIFO is designed in this paper which is comprised of five modules; baud rate generators, FIFO, transmitters and receivers. This design uses VERILOG as design language to achieve the modules of UART. Using XILINX software, Spartan 3E FPGA to complete simulation and test. The results are stable and reliable data transmission with some reference value and great flexibility, high integration as FIFO is used to avoid data loss.

VII. REFERENCES

[1] Fang Yi-yuan, Chen Xue-jun, "Design and Simulation of UART Serial Communication Module Based on VHDL," 3rd Int'l Workshop on Intel. Sys. And App. (ISA 2011), Wuhan, China, May 2011.

[2] Nurul Fatihah Jusoh, Azlina Ibrahim, Muhamad Adib Haron and Fuziah Sulaiman, "An FPGA Implementation of Shift Converter Block Technique on FIFO for UART," IEEE International RF and Microwave Conference (RFM 2011), Seremban, Malaysia, Dec 2011.

[3] J. Norhuzaimin and H.H Maimun, "The Design of High Speed UART," Asia-Pacific Conference on Applied Electromagnetic (APACE 2005), Johor, Malaysia, Dec.2005

[4] Ritesh Kumar Agrawal, Vivek Ranjan Mishra," The Design of High Speed UART," Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).

[5] Y.Wang, and K. Song, "A new approach to realize UART," Int'l Conf. on Elect. and Mech. Eng. and IT (EMEIT 2011), Harbin, Heilongjiang, China, Aug. 2011.

[6] Brian C. O'Neill, Member, IEEE, Steve Clark, and Kar L. Wong," Serial Communication Circuit with Optimized Skew Characteristics," IEEE Communications Letters, Vol. 5, No. 6, June 2001.

[7] New IC Caps Two Decades of UART Development-Application Note – Maxim, March 2000.

[8] Charles H. Roth, Jr, Digital System Design by using VHDL, PWS Publishing Company, 1998.

[9]. Bhaskar, J. A Verilog HDL Primer -Pearson Education. 2001.

[10] Palnitkar, Samir. Verilog HDL - A Guide to Digital Design and Synthesis.

Author's Profile:

Suraj Prakash Ahirwar is currently persuing M.Tech with specialization in Microelectronics and VLSI Design at S.G.S.I.T.S, Indore, India. He received his Bachelor degree in Electronics and Communication Engineering from Gyan Ganga College of Technology, Jabalpur. His field of interest includes Digital VLSI Design, EDA, RTL simulation and synthesis, Verilog HDL.

D.S.Ajnar received the B.E. degree in Electronics and Communication Engineering from D.A.V.V University, India in 1993 and M.E. Degree in Digital Techniques & Instrumentation Engineering from Rajiv Gandhi Technical University Bhopal, India in 2000. He has been teaching and in research profession since 1995. He is now working as Associate Professor in Department of, Electronics & Instrumentation Engineering S.G.S.I.T.S, Indore, India. His interest of research is in Designing of analog filter and Current-Conveyor.

P.K. Jain received the B.E. degree in Electronics and communication Engineering from D.A.V.V. University, India in 1987 and M.E. Degree in Digital Techniques & Instrumentation Engineering from Rajiv Gandhi Technical University Bhopal, India in 1993. He has been teaching and in research profession since 1988. He is now working as Associate Professor in Department of Electronics & Instrumentation Engineering, S.G.S.I.T.S. Indore, India. His interested field of research is analog circuit design.