BEST: International Journal of Management, Information Technology and Engineering (BEST: IJMITE)
ISSN 2348-0513
Vol. 2, Issue 5, May 2014, 57-62

DESIGN AND IMPLEMENTATION OF UART USING VHDL ON FPGA

MAHESH GIRI & P. P. SHINGARE

Department of Electronics and Telecommunication Engineering, College of Engineering, Pune, Maharashtra, India

ABSTRACT

In this paper we propose a technique for software implementation of an UART (Universal Asynchronous Receive-Transmit) with the goal of getting a customizable UART-core which can be used as a module in implementing a bigger system irrespective of one's choice of implementation platform. Here at the implementation of the system in a well efficient manner there is an effective utilization of the core based on the strategy of the UART plays a crucial role in its representative analysis in a well oriented fashion on the effective strategy of the VHDL plays a crucial role in its representation in a well effective manner respectively. Here the above implementation takes place on the tool of the XILINX in a well stipulated fashion with respect to the environment oriented well efficient strategy of the 10.1 ISE plays a crucial role in its representation respectively. There is a test bench has been conducted on the well effective environment based scenario based on the stipulated fashion of its implemented strategy of FPGA related SPARTAN of 3e in a well efficient manner respectively. The simulation results as well as the test results are seen to be satisfactory.

KEYWORDS: Universal Asynchronous Receives and Transmits, Soft Core Implementation, Independent Platform, VHDL Respectively

INTRODUCTION

There is a huge establishment of the system takes place in the environment of the communication related to the aspect of the serial phenomena plays a crucial role in its representative analysis of the transmission of the data in a simultaneous fashion respectively . Here apart from the strategy of the consumption of the power plays a crucial role in its representative analysis point of view where there is an accurate consumption of the power that is programming based on the on board strategy plays an efficient role and the responsibility in its representative analysis point of view respectively. Here there is a communication of the data takes place in the serial manner respectively.

Asynchronous serial communication has advantages of high reliability, less transmission line and long transmission distance, therefore is widely used to exchange data between a computer and external devices. Asynchronous serial communication is implemented by UART. It provides full-duplex communication in serial link; this has been widely used in the data communications. UART includes a transmitter and a receiver. Transmitter controls transmission by taking a data word in parallel format and directing the UART to transmit it in a serially. Likewise, the Receiver must detect transmission, receive the data in serially, and store the data word in a parallel format. The conversion of serial to parallel data is handled by UART. Serial communication reduces the distortion of a signal; therefore data transfer is possible between two systems separated by great distance. The UART serial module is divided into three sub-modules: The baud rate generator, receiver module and transmitter module. The baud rate generator is used to produce a local clock signal. In data transmission through the UART, once the baud-rate has been established, both the transmitter and the receiver's internal clock are set to the same frequency. TXD is the transmit side, i.e. the output of the UART RXD is the receiver, i.e. the input of the UART. 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 data bytes into serial bits according to the frame format and transmits those bits through TXD.UART's basic features are: There are two states in the signal line, using logic high and logic low to distinguish respectively. UART frame format consist of a start bit, data bit, parity bit and stop bit. After the Start Bit the data bits are sent, with the Least Significant Bit (LSB) sent first. The start bit is always low and the stop bit is always high. When the complete data word has been sent, it adds a parity bit this parity bit may be used by the receiver to perform error checking. Then at least one Stop Bit is sent by the transmitter. Because asynchronous data are "self-synchronizing", if there is no data to transmit, the transmission line will be idle.

THE UART MODULE

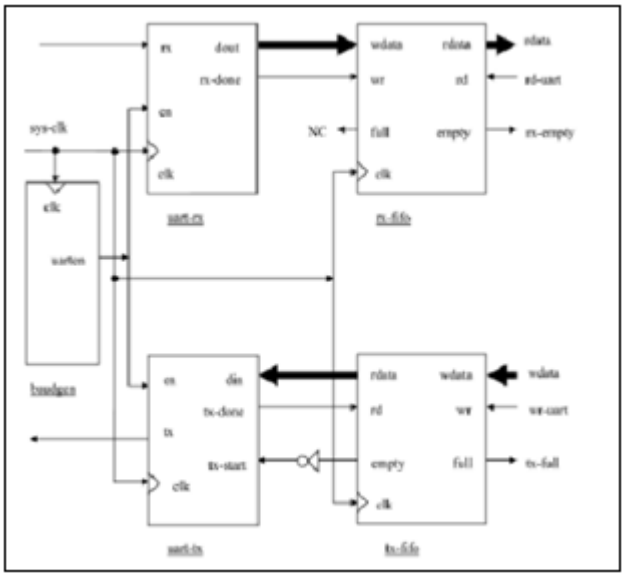

Figure 1: Uart Block Diagram

The UART module that we have designed consists of five parts namely (i) "uart-rx", which takes in the serial data (as a frame) coming through the 'rx' line, retrieves the actual data and converts to parallel form (usually as a byte). (ii) "uart-tx", which does the opposite function of the "uart-rx" module and transmits the frame through the 'tx' line. (iii) "baudgen", which generates a clock which occurs 16-times(the default over-sampling rate)in one bit-time period. (iv) "tx-fifo", which stores temporarily the bytes (that usually comes from a faster processor) to send, as the sending process takes some time. (v) "rx-fifo", which is the replica of the "tx-fifo"vi) No. of buffers needed to cope up with the speed difference between the system using the UART and the rate at which data are coming (default-8).And the core is made available in full VHDL-which makes it platform-independent.

UARTPROTOCOL

The UART protocol is a serial communication protocol that takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signaling levels. External signals may be of many different forms. Examples of standards for voltage signaling are RS-232, RS-422 and RS-485 from the EIA. Typically its a 3-line (transmit, receive, ground) communication. Communication which enables it to be "full duplex" (both send and receive at the same time) or "half duplex" (devices take turns transmitting and receiving).

Transmiter

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.

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.

SIMULATION AND SYNTHESIS RESULTS

We have simulated each and every part of our module separately in modelsim6.4b. The simulation results for the ‘baudgen’, ‘fifo’, ‘uart-rx’, and ‘uart-tx’ sub modules are shown in the figures below respectively. We have synthesis top module in Xilinx10.1.

UART Transmitter

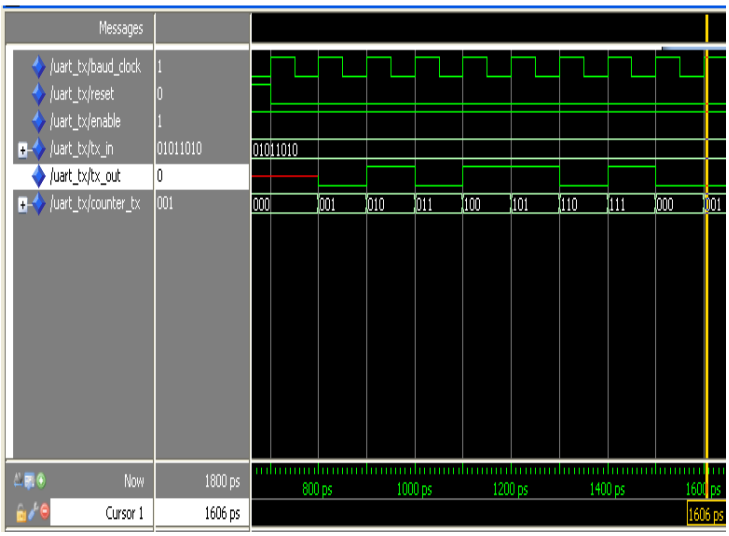

Figure 2

UART Receiver

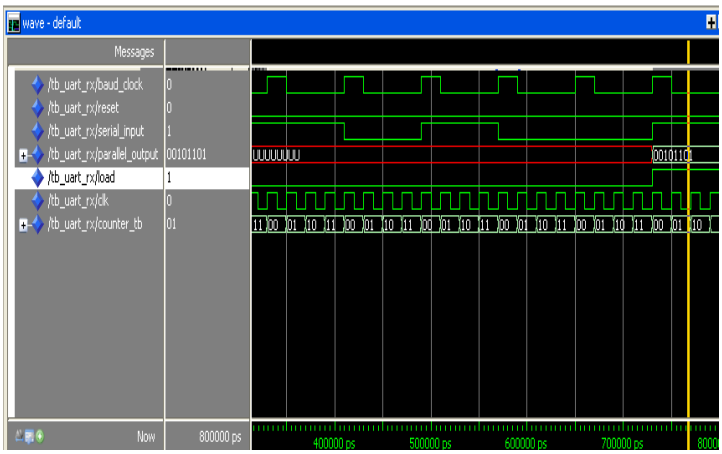

Figure 3

FIFO

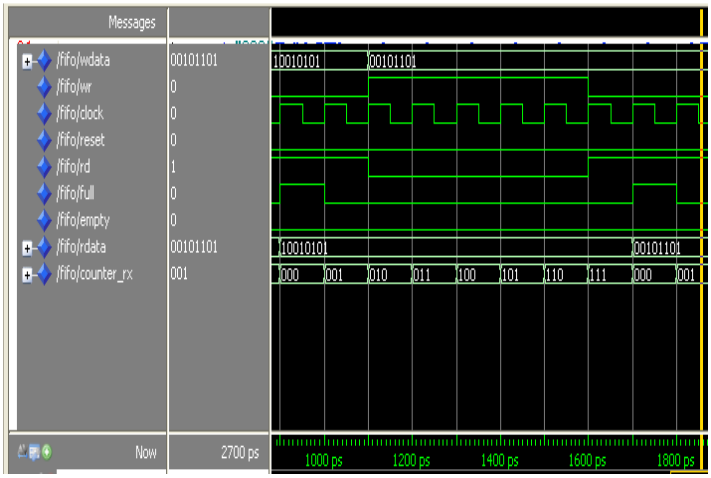

Figure 4

Topmodule

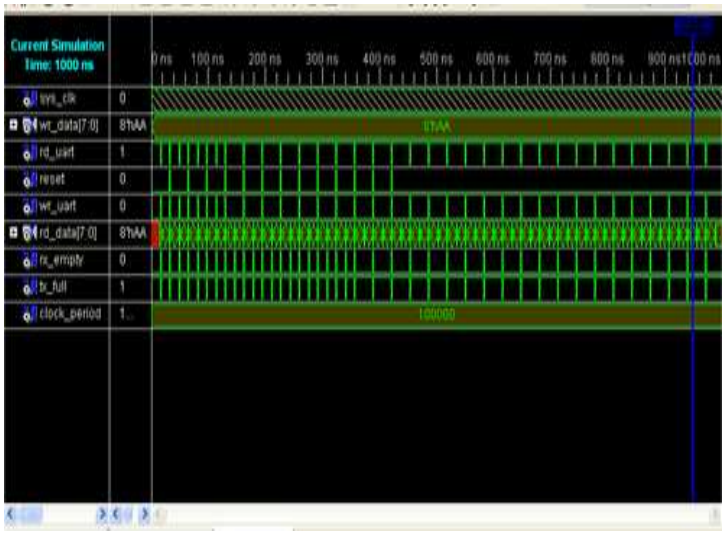

Figure 5

Schematic Results

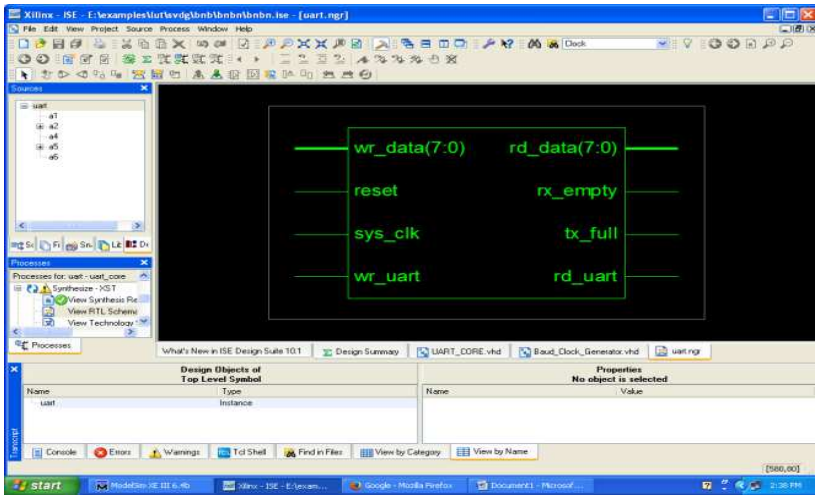

Figure 6

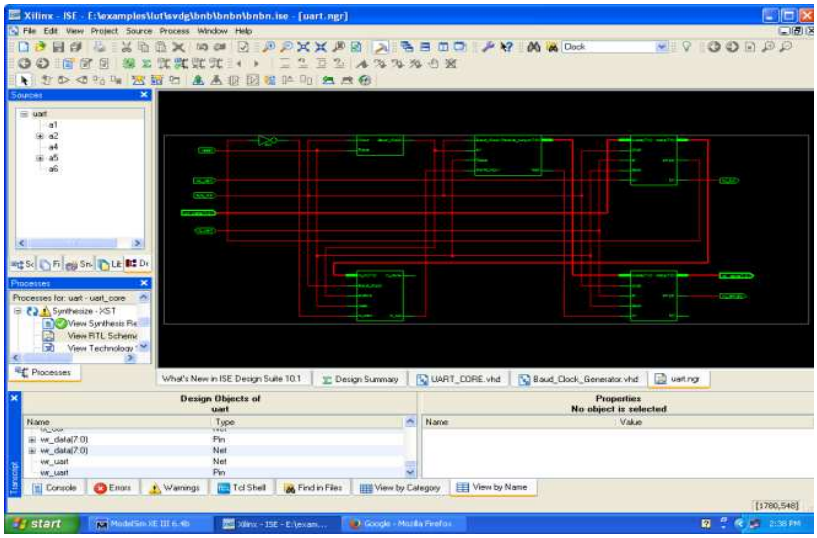

Figure 7

Synthesis Report

sss Project Status (09/20/2013 - 11:46:04)			
Project File:	sss.ise	**Current State:**	Synthesized
Module Name:	uart	**• Errors:**	No Errors
Target Device:	xc3s500e-5fg320	**• Warnings:**	6 Warnings
Product Version:	ISE 10.1 - Foundation Simulator	**• Routing Results:**	
Design Goal:	Balanced	**• Timing Constraints:**	
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	

sss Partition Summary	[-]
No partition information was found.	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	35	4656	0%
Number of Slice Flip Flops	53	9312	0%
Number of 4 input LUTs	43	9312	0%
Number of bonded IOBs	22	232	9%
Number of GCLKs	1	24	4%

Figure 8

Total percentage of the device (spartan 3-an) resources utilized to implement our module has been calculated by Xilinx-XST synthesizer which is found to be nominal.

CONCLUSIONS

We have concluded that the authors had used FIFO and Shift register separately for storing the data and in some paper they have used baud rate generator with single frequency. In this we had used FIFO as well as Shift register and also an automatic baud rate generator which will change its baud rate according to the change in frequency. This design uses VHDL language to achieve the modules of the UART. We have designed our UART module in generic form which is operating fine with no under run error and can be customized to make it free from overrun error with the capability provided and so can be made available as IPcore (Intellectual-Property-Core) by simply coating it with a proper wrapper.

REFERENCES

1. James O. Hamblen, Tyson S. Hall, Michael D. Furman, Rapid prototyping of digital systems, 2nded., springer Publication, 2001.
2. Dauglas L. Perry, VHDL Programming by Example, 4thed., The Tata-McGrawHill Pub, 2002.
3. Volnei A. Pedroni, Circuit Design with VHDL, 3rded., The MIT Press, 2004..
4. XilinxInc, xpsuart lite v1.00, 3rded. DS571, January 14, 2008.
5. ARM Ltd, AMBA 3 APB Protocol Specification, 2nded.
6. Verilog HDL by Samir Palnitkar Publisher: Prentice Hall PTR (January 15, 1996)
7. Paragk.Lala Introduction to Logic Circuit Testing Morgan and Claypool Publishers 2009.
8. Naresh Patel Vatsalkumar Patel, Vikaskumar Patel ”VHDL Implementation of UART with Status Register” in International Conference on Communication Systems and Network Technologies.2012.
9. Dr. GarimaBandhawarkarWakhleIti Aggarwal and ShwetaGaba “Synthesis and Implementation of UART using VHDL Codes” in International Symposium on Computer, Consumer and Control 2012.