DOI: 10.21917/ijme.2019.0127

DESIGN AND IMPLEMENTATION OF UART WITH FIFO BUFFER USING VHDL ON FPGA

Sardi Irfansyah
Department of Electrical Engineering, Gunadarma University, Indonesia

Abstract

Universal Asynchronous Receiver Transmitter (UART) is a communication protocol commonly used for serial data communication. This paper presents the design and implementation method of a Universal Asynchronous Receiver Transmitter (UART) using VHSIC Hardware Description Language (VHDL). UART will be implemented to picoblaze processor which can be implemented in large system and have high flexibility in FPGA based design. UART controller has been designed using FIFO (First In First Out) buffer to avoid loss of data. Simulated and synthesized using Xilinx ISE 13.1. The design is successfully downloaded and verified on Spartan-3E FPGA board. The data that is sent will generate output LEDs on Spartan-3E. The total number of slice required is less than 10%. The number of slice Flip Flops are 1%, the total number of 4 input LUTs are 3% and the number of occupied slices are 3%. This design has a small resource.

Keywords:

UART, VHDL, FIFO Buffer, FPGA, Xilinx ISE 13.1, Spartan-3E

1. INTRODUCTION

The development of technology and digital communication at this time is very fast. To communicate with less transmission line and long transmission distance that it needed a serial communication such as Asynchronous serial. Types of serial communications are typically using UART (Universal Asynchronous Receiver-Transmitter) as an interface for serial communication. A UART is a circuit that sends parallel data through serial lines [1]. Universal Asynchronous Receiver Transmitter (UART) is a microchip with programming that controls a computer's interface to its attached serial devices [2]. The UART is basically used in between the slow and the fast peripheral devices for example: computer and printer or in between the controller and LCD [3]. UART allows full-duplex communication in serial link, thus has been widely used in the data communications and control system.

Chip interface will cause the waste of resources and increased cost. By using FPGA (Field Programmable Gate Array) will reduce the waste of resources, because the FPGA is composed thousands of logic circuits which can be configured to perform the desired functions. Therefore, UART module will be implemented into the FPGA. Implementation will be using VHDL to design it. To be able to generate a high flexibility in the implementation of UART needed a processor that can perform the process automatically. The processor soft core is picoblaze (KCPSM3), which can be implemented on Spartan 3E FPGA family. By applying the program using the processor picoblaze can generate high flexibility so that if we modify the program does not need to be changed entirely.

Basic UART communication needs only two signal lines (RXD, TXD) to complete full-duplex data communication [4]. TXD is the transmit side and RXD is the receiver. In the UART, the speed of data transmission (Baud rate) and the phase of the clock on the transmitter and the receiver must be synchronized. It is necessary for the synchronization between transmitter and receiver. It will be done by bits "Start" and a bit "Stop". When the transmitter is idle, the data line is in the high logic "1". When the transmitter wants to transmit data, UART will be set first to logic "0" for one bit. This signal at the receiver will be recognized as a signal "Start" that is used to synchronize the phase of clock with the transmitter clock phase and then the data will be sent serially from the lowest bit "0" to the highest bit.

When the receiver has received all of the data bits, 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, the UART will report a framing error to the host processor when the data 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.

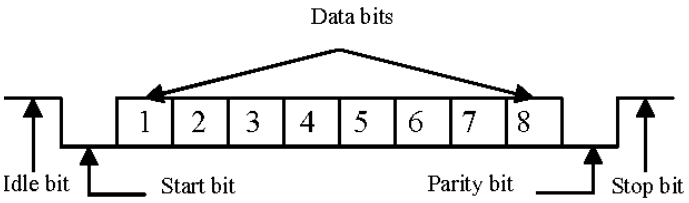

Fig.1. UART Frame Format

If the baud rate used too fast will lead to the received data didn't match the data sent and the error value will be high. To calculate the bit period for one bit of data can be calculated by the formula as shown in Fig.2.

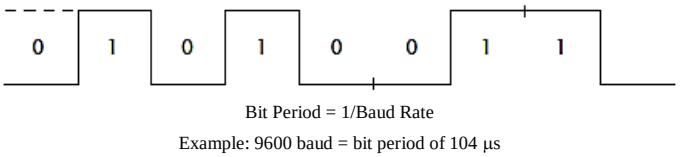

Fig.2. Bit Period

2. LITERATURE REVIEW

2.1 SERIAL COMMUNICATION

There are two kinds of serial communications such as asynchronous serial communication and synchronous serial communication. Synchronous serial communication is requires that the clocks in the transmitting and receiving devices are synchronize and running at the same rate, so the receiver can

sample the signal at the same time intervals used by the transmitter. Asynchronous serial communication is serial communication in which the transmitter and receiver are not continuously synchronized by a common clock signal. In order for the received data same as the data sent then the clock frequency must be the same and there should be synchronization. After the synchronization, the transmitter will transmit the data according to the clock frequency of the transmitter and then the receiver will read the data according to the receiver clock frequency [2].

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 peripheral [3]. Asynchronous serial communication is usually implemented by Universal Asynchronous Receiver- Transmitter (UART).

Universal Asynchronous Receiver-Transmitter (UART) is a computer hardware that translates data between parallel and serial forms. UART is a kind serial communication protocol, mostly used for short-distance and low-speed between computers and peripherals. UART is usually in the form of an integrated circuit used for serial communications on a computer or peripheral device serial port. UART allows full-duplex communication in serial link, thus has been widely used in the data communications and control system.

In serial communication, the data is sent can be either 8-bit data size. From this data format, each data readable can be translated into bits that represent specific data. The baud rate is used to determine the time of the serial communication. Standard baud rates include 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000 and 256000 bits per second.

2.2 FPGA

Field Programmable is a digital integrated circuit that can be designed and programmed according with the wishes and needs of the user or consumer without going through the stage of "burn" in laboratory or "hardwired" by the manufacturer of the device. Gate Array means that FPGA consists of digital gates where each gate interconnection can be configured between each other.

Each FPGA vendors typically provide software to design and program the FPGA. Software used on any FPGA vendor using HDL (hardware description language) or a schematic design. HDL commonly used is VHDL and Verilog.

The HDL will be more useful when working on large structures, because it would be easier to implement numerically rather than having to draw every piece of the circuit by hand. However, schematic entry can allow for easier visualisation of a design. Then, using an electronic design automation tool, a netlist is generated. The netlist can then be fitted to the FPGA architecture using a process called place-and-route, usually performed by the FPGA company proprietary place-and-route software.

The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated is used to reconfigure the FPGA. This file is transferred to the FPGA/CPLD via a serial interface (JTAG) or to an external memory device like an EEPROM. To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called IP (intellectual property) cores, and are available from FPGA vendors and third-party IP suppliers (rarely free and typically released under proprietary licenses).

FPGA contain an array of programmable logic blocks and a hardwired connection. Components programmed into the FPGA gate includes ordinary logic gates (AND, OR, XOR, NOT) as well as the type of mathematical functions and more complex combinations (decoder, adder, subtractor, multiplier, etc.). The blocks in the FPGA component may also contain a memory element (register) ranging from flip-flop to the RAM (Random Access Memory). FPGA programming can used Verilog or VHDL (VHSIC Hardware Description Language). VHSIC is an abbreviation for Very High Speed Integrated Circuit. VHDL intended for synthesis and simulation circuit.

2.3 PICOBLAZE

Picoblaze is a soft-core processor from Xilinx for use in their FPGA and CPLD products. Picoblaze has 8-bit RISC (Reduced Instruction Set Computing) architecture and has a speed of up to 100 MIPS (Million Instructions per Second) on Virtex 4 FPGA family. Picoblaze design originally named KCPSM which stands for "Constant (K) Programmable State Machine Code" [4].

The basic of design for use picoblaze is:

- Picoblaze written using assembler programming language with extensions file .psm
- KCPSM3 assembler can only be run on the file extension .psm and the file extension .vhd will execute the instructions located in Block Memory as its output.

Picoblaze program memory to be implemented on a single block RAM in the FPGA, which will then be configured to function as 1K × 18 instruction PROM. Program to be executed will be initialized in block RAM. Below is a component picoblaze.

Picoblaze development kit provided by Xilinx included a VHDL description of a basic UART. Likewise, Spartan-3 board provides all the facilities to enable serial communication: a DB-9 port and a voltage-level shifter circuit. In order to enable Picoblaze to handle a UART as a peripheral, both the transmitter (Tx) and receiver (Rx) modules have been encapsulated along with an interface module. This interface allows picoblaze to operate UART by access to a set of registers.

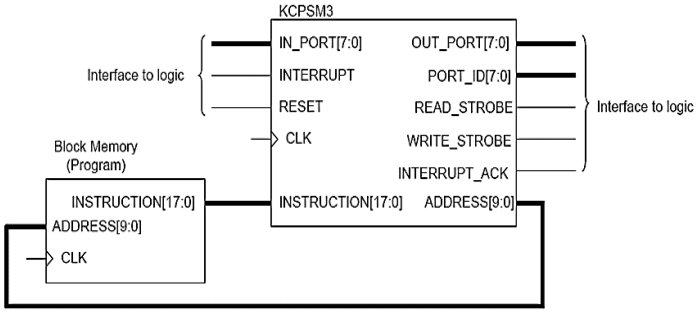

Fig.3. Components of Picoblaze

2.4 VHSI HARDWARE DESCRIPTION LANGUAGE (VHDL)

VHDL (VHSIC Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays (FPGA) and integrated circuits. VHSIC is an abbreviation for Very High Speed Integrated Circuit.

The advantages of using VHDL compared to Verilog is a VHDL does not depend on technology or vendor, so it can be moved (portable) and can be used again (reusable). Another benefit is that VHDL allows the description of a concurrent system. VHDL is a dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, one instruction at a time.

VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design. A test bench is used to simulate of the design.

VHDL is frequently used for two different goals: simulation of electronic designs and Synthesis of such designs. Synthesis is a process where a VHDL is compiled and mapped into an implementation technology such as an FPGA or an ASIC. Simulation is a very important process because it can be applied to any abstract level, both at the level schematic or VHDL code.

Implementation of a logic design with the software usually consists of several steps:

Step 1: Describe a logic circuit of the system designed using a hardware description language (HDL) such as VHDL or Verilog or to describe it using schematic editor.

Step 2: Perform the synthesis process which generates netlists for each source file. Netlist is a description of various logic gates of a design and how logic gates are connected.

Step 3: In the stage of implement design have three steps: translate, map, place and route. Translate is the process of merges multiple files into a single netlist. After translate, the design is mapped to slices and I / O blocks. The design is mapped to slices and I/O blocks. Place and route is process placing the design on the chip and components connected.

Step 4: Once the implementation phase is complete, the application creates a bitstream file that contains the value of a logic '1' and '0' which is a representation of a digital circuit configuration to be implemented into the FPGA to show the open or the closed switch on FPGA.

Step 5: Bitstream file to be uploaded into FPGA or memory are available via the JTAG cable according to type of development board is used.

3. DESIGN METHODOLOGY

Design and implementation will be discussed in this section. UART transmitter logic receive parallel signal and converted into serial data, the UART receiver logic receive serial signal and converted into parallel signal and FIFO are used to avoid the loss of data. FPGAs have programmable logic components called 'logic blocks' and a hierarchy or reconfigurable interconnects which facilitate the wiring of the blocks together [5]. The design that has been created will be implemented to Spartan 3E FPGA.

3.1 DESIGN OF SYSTEM USING SOFTWARE

In the process of design system using VHDL code will be implemented using Xilinx ISE 13.1 software.

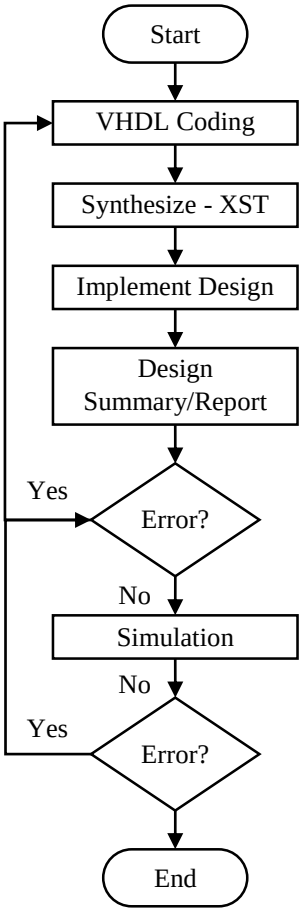

Fig.4. Flowchart Design of System using VHDL

In Fig.4, after making a VHDL code, then the synthesize process using the Xilinx Synthesis Technology (XST) synthesis tool. Synthesize is the process of generating a netlist for each source file. After conducting the synthesis process then implement design. In this section there are three steps: translate, map, place and route. Translate is the process of combining multiple files into a netlist. Map is a process to map a slice and I/O Blocks. Place and route is process placing the design on the chip and components connected. After implement design is completed, it can be seen a summary of the design and report. If there is an error then corrected VDHL code. If no error found, then do simulation. If there is an error in the simulation then fix the VHDL code. If no error found, then do simulation. If there is an error in the simulation then fix the VHDL code.

3.2 DESCRIPTION OF THE SYSTEM

Spartan 3E FPGA board is used as a system controller and Xilinx 13.1 as a tool to create the program. In Fig.5, input ports are switches, push button, clk_50MHz, and rs232_dce_rxd,

whereas output ports are LEDs and rs232_dce_txd. User will upload the program into the spartan 3E. User will transmit data to rs232_dce_rxd (Spartan 3E) in the form of ASCII characters and then Spartan 3E will transmit the data to the PC. UART_RX and UART_TX will process the data and then it will connect to picoblaze. It will change lowercase character to uppercase when sw(0) ON and the LEDs will light up, then rs232_dce_txd will be sent the data to the PC.

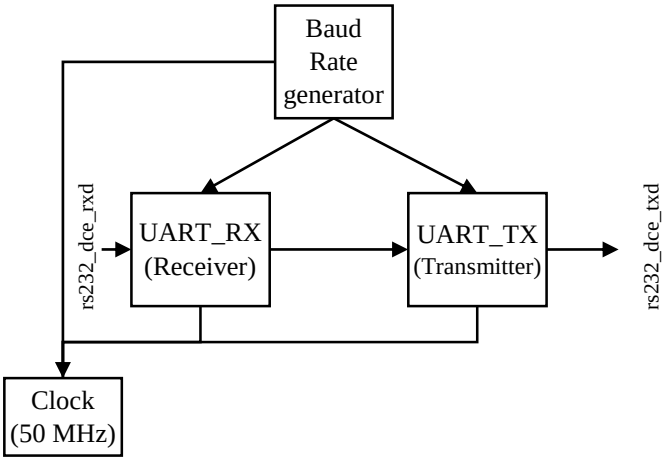

Fig.5. Block Diagram of System

On the program 'Spartan3e.vhd' include UART_RX, UART_TX, Pb1_rom and picoblaze1 (KCPSM3). UART UART TX and RX are modules to process data transmission from the PC to the spartan 3E. The pb1_rom derived from assembly file (skripsi.psm) were generated in block ROM of KCPSM3.

3.3 BLOCK DIAGRAM OF UART

In Fig.6, UART_TX module and UART_RX module are connected with baud rate generator. Baud rate generator is actually a frequency divider.

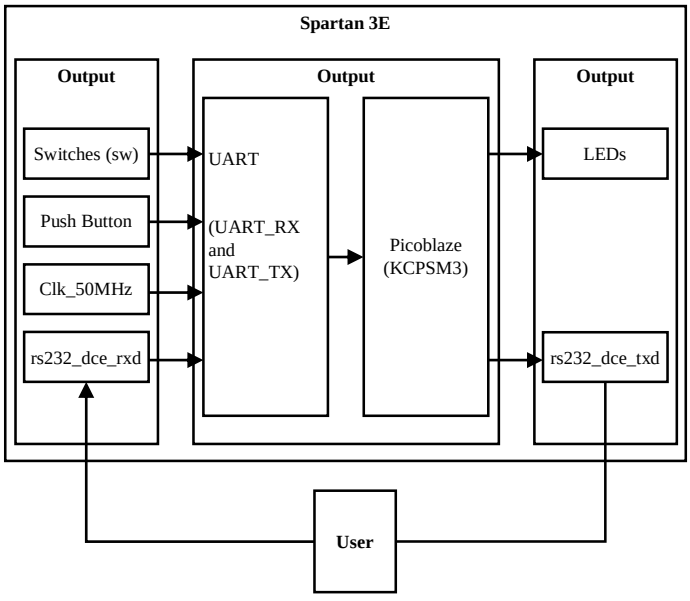

Fig.6. Block Diagram of UART

Consider the baud rate at the transmitter is 9600bps. This means that the time period for one data bit is 1/9600 which comes out to be 104µs. The clock frequency available on the Spartan 3E FPGA board is 50MHz. Data sent from PC to Spartan 3E are ASCII characters and then it will be accepted by rs232_dce_rxd as serial_in (UART_RX module). UART_RX module will convert the data into data parallel. UART TX module will convert bytes into serial bits, then it will send bits of data through a signal tx (rs232_dce_txd) as serial_out.

3.3.1 UART_RX:

UART_RX is a combination FIFO buffer 'bbfifo_16×8' and the constant (k) compact UART reciever 'kcuart_rx' modules. The data will be captured by the FIFO buffer. The input side of the FIFO is under the control of the 'kcuart_rx' receiver. As shown in Fig.7, In the UART_RX module has several input ports and output ports.

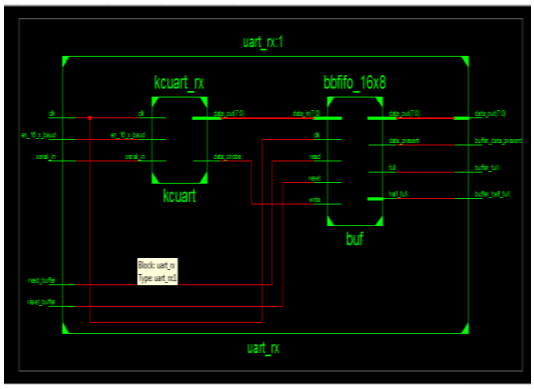

Fig.7. Block of UART_RX

Ports on UART_RX will be connected to ports on the Spartan 3E. Port Serial_in will be connected to the port rs232_dce_rxd through rx signal, so the value of serial_in = rs232_dce_rxd. Port btn_south will connect with reset_buffer.

3.3.2 UART_TX:

UART_TX is a combination FIFO buffer 'bbfifo_16×8' and the constant (k) compact UART transmitter 'kcuart_tx' modules. The FIFO buffer is used to accept byte data for transmission. The output side of the FIFO is under the control of the 'kcuart_tx' transmitter. The buffer is automatically read by the 'kcuart_tx' circuit to pass the data to the serial line. As shown in Fig.8, In the UART_TX module has several input ports and output ports.

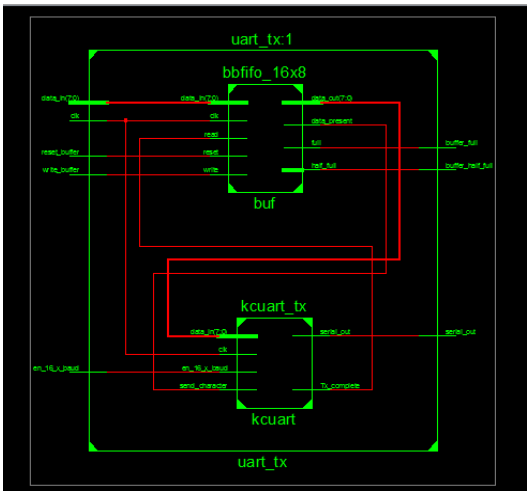

Fig.8. Block of UART_TX

Ports on UART_TX will be connected to ports on the Spartan 3E. Port Serial_out will be connected to the port rs232_dce_txd through tx signal, so the value of serial_out = rs232_dce_txd. Port btn_south will connect with reset_buffer.

3.4 RTL SCHEMATIC OF SYSTEM

The RTL scheme refers to the system logic that will be implemented. The whole system on the spartan 3E block diagram as shown in Fig.3 will produce a schematic as shown in Fig.9.

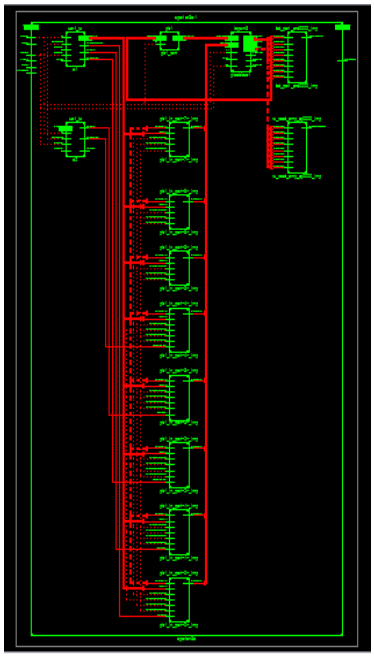

Fig.9. RTL Schematic of System

3.5 DATA TRANSMISSION PROCESS

In this section explains the process of sending data that will be implemented to Spartan 3E FPGA Board (see Fig.10).

When the device is activated, the device will begin to initialize. The initial condition of LEDs is “00001111”. When the PC give input data (ASCII character) to the Spartan 3E, then LEDs = input data and output data = input data. When sw (0) = 1 (active high) then the LEDs = data input that has changed from lowercase letter to uppercase and then output data = input data (lowercase letter to uppercase). When reset = 1 (active high) then the LEDs will return to “00001111”.

Example: Data input is the letter ‘u’ or 75 in hexadecimal, then the LEDs will light according to the data input, LEDs = “01110101”. If if the switch on the letter ‘u’ will be changed to ‘U’, LEDs = “01010101”.

4. TESTING AND ANALYSIS

Testing and analysis of data transmission will be discussed in this section. The data will be sent as ASCII characters and then converted into Binary. It will be implemented into Spartan 3E FPGA Board.

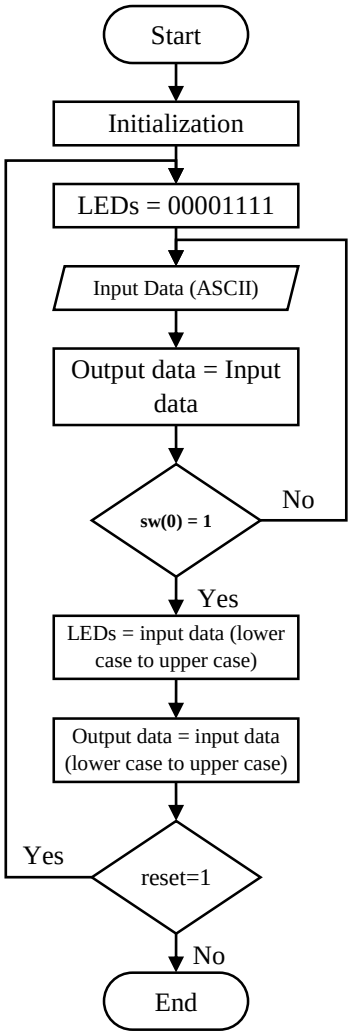

Fig.10. Flowchart of Data Transmission Process

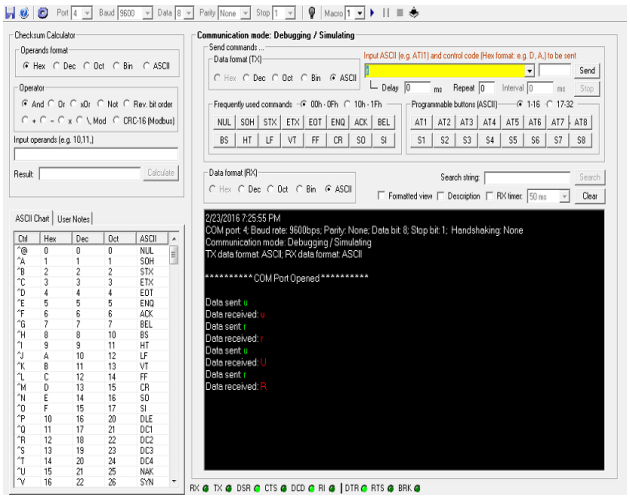

Fig.11. 232 Analyzer Window with Baud Rate of 9.600 bps

4.1 SYSTEM TESTING AND ANALYSIS

The process of sending data from the PC to the Spartan 3E and then retransmits data from Spartan 3E to PC will be displayed on 232Analyzer. 232Analyzer is an advanced serial port protocol analyzer software that allows programmers, engineers, and others

to control, monitor and analyze serial port (RS232/RS485/RS422/TTL, etc.) activities. As shown in Fig.11, 232Analyzer will be set with baud rate of 9.600bps, 8 data bits, 1 stop bit and none parity.

By using this software data will be sent from PC to Spartan 3E and then generate output on the LEDs (see Fig.12).

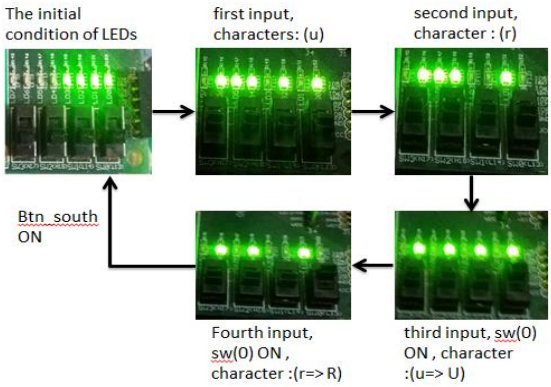

Fig.12. LEDs output on spartan 3E FPGA Board

The initial conditions of LEDs are “00001111”. Example, data is sent in the form of the character ‘u’ (in ASCII), then the LEDs will light up according to the binary of the received data “01110101” (conversion from ASCII to binary). When btn_south ON then the LEDs will return to initial condition is “00001111” (this port functions as a reset). If sw (0) ON then received data will change from lowercase letters to uppercase.

Table.1. Test Result with Baud Rate of 9.600bps

No.	Data sent (PC)	Output LED (Spartan 3E)	Data Received (PC)
1	u (75 ‘Hex’)	01110101 (75 ‘Hex’)	u (75 ‘Hex’)
2	r (72 ‘Hex’)	01110010 (72 ‘Hex’)	r (72 ‘Hex’)
sw(0) ON			
3	u (75 ‘Hex’)	01010101 (55 ‘Hex’)	U (55 ‘Hex’)
4	r (72 ‘Hex’)	01010010 (52 ‘Hex’)	R (52 ‘Hex’)

In Table.1 shows that that the received data is the same as the data sent. When sw (0) ON, the data received on the Spartan 3E will change from lowercase letters to uppercase, then the Spartan 3E will send the data that has been received to the PC, so it will produce output on a PC. The output will be displayed on 232Analyzer.

5. RESULTS

In the Fig.13, functional simulation was performed using ISim. In this simulation using baud rate of 9600bps, the system clock frequency is set to 50MHz.

The value of rs232_dce_rxd = serial_in on UART_RX (1 start bit, 8 data bits, and 1 stop bit), rx_data = data_out (The parallel Byte (8-bit) data which has been received), rx_read = read_buffer (An active HIGH or logic ‘1’ input indicates that the data provided at the ‘data_out’ port has been read). The value of rs232_dce_txd = serial out on UART TX, tx_data = data_in (The parallel Byte (8-bit) data to be transmitted serially). Tx_write = write_buffer (An active HIGH or logic ‘1’ indicates that the data is currently being applied to the ‘data_in’ port to be written to the internal buffer). tx_complete will be active high if ‘data_in’ port has been written to the internal buffer. Signal en_16_x_baud value will change to 1 when 6.51µs as shown in Fig.14.

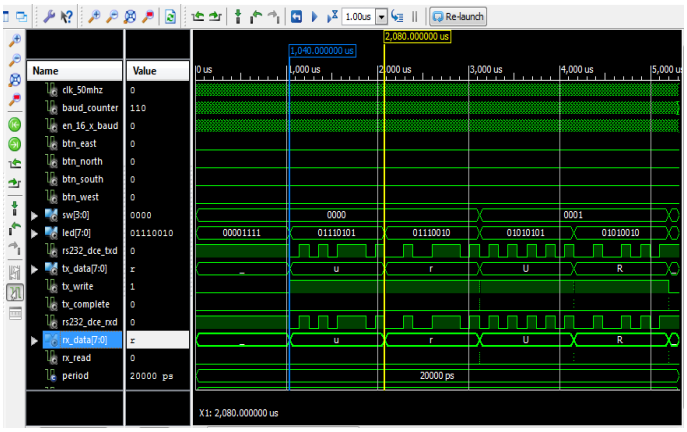

Fig.13. UART Waveforms

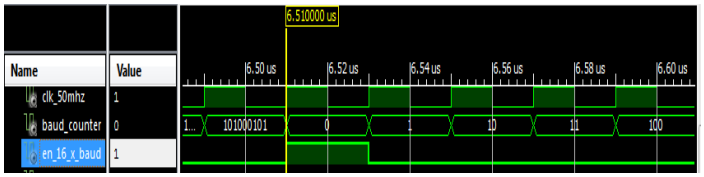

Fig.14: Baud Rate Waveforms

5.1 DESIGN SUMMARY RESULTS

In the Fig.15, it can be seen that the number of slice flip flops are 1%, the total number of 4 input LUTs are 3% and the number of occupied slices are 3%. This design has small resource requirements and also still to be developed.

Current Errors			
No Errors Found			
Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	168	9,312	1%
Number of 4 input LUTs	275	9,312	2%
Number of occupied Slices	171	4,656	3%
Number of Slices containing only related logic	171	171	100%
Number of Slices containing unrelated logic	0	171	0%
Total Number of 4 input LUTs	285	9,312	3%
Number used as logic	171		
Number used as a route-thru	10		
Number used for Dual Port RAMs	16		
Number used for 32x1 RAMs	52		
Number used as Shift registers	36		
Number of bonded IOBs	19	232	8%
Number of RAMB16s	1	20	5%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.57		

Fig.15. Design Summary

6. CONCLUSION

The conclusions of the research can be concluded as follows:

1. Implementation of UART with picoblaze successfully performed on a Xilinx Spartan 3E FPGA board. If the switch (0) ON then the lowercase letters will change to

uppercase. Output LED will change according to the data received on Spartan 3E.

2. The design has great flexibility and high integration as FIFO buffer is used to avoid data loss.
3. In the design has been created, the total number of slice required is less than 10%. The number of slice Flip Flops are 1%, the total number of 4 input LUTs are 3% and the number of occupied slices are 3%. This design has a small resource.

Here are some suggestions that can be done for future development:

1. ASCII characters to be sent can be changed to other characters to find out the results of that received when using other characters.
2. LED output on Spartan 3E can be changed using the LCD as a sign that the data has been received.
3. The Baud rate can be changed to get high speed data transmission.

REFERENCES

[1] Liu Weifeng, Zhuang Yiqi, Liu Feng and He Wei, "Design of a High Performance Embedded UART", *Chinese Journal of Electron Devices*, Vol. 30, No. 4, pp. 1275-1278, 2007.

[2] S. Saha, M.A. Rahman and A. Thakur, "Design and Implementation of a BIST Embedded High Speed RS- 422 Utilized UART over FPGA," *Proceedings of 4th International Conference on Computing, Communications and Networking Technologies*, pp. 1-5, 2013

[3] Garima Bandhawarkar Wakhle, Iti Aggarwal and Shweta Gaba, "Synthesis and Implementation of UART using VHDL Codes", *Proceedings of International Symposium on Computer, Consumer and Control*, pp. 145-149, 2012.

[4] V. Naresh and V. Patel, "VHDL Implementation of UART with Status Register", *Proceedings of International Conference on Communication Systems and Network Technologies*, pp. 11-14, 2012.

[5] N.R. Laddha and A.P. Thakare, "Implementation of Serial Communication using UART with Configurable Baud Rate", *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol. 1, No. 4, pp. 263-268, 2016.

[6] L. Ruchika, M. Mithilesh and D. Vidya, "Design and Implementation of UART", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, No. 5, pp. 18-24, 2015.

[7] Fang Yi-Yuan and Chen Xue Jun, "Design and Simulation of UART Serial Communication Module Based on VHDL", *Proceedings of 3rd International Workshop on Intelligent Systems and Applications*, pp. 1-4, 2011.

[8] Y. Wang and K. Song, "A New Approach to Realize UART", *Proceedings of International Conference on Electronic and Mechanical Engineering and Information Technology*, Vol. 5, pp. 2749-2752, 2011.

[9] Ken Chapman, "Ultra-Compact UART Macros for Spartan-6, Virtex-6 and 7-Series", Available at: http://ohm.bu.edu/~dean/Xilinx/KCPSM6_Release7_30Sept13/UART_and_PicoTerm/UART6_User_Guide_and_Reference_Designs_29March13.pdf.

[10] D.V.R.K Raju, B. Narsingarao, K. Vikash, D.A. Kumar and S.R. Krishna, "UART Serial Communication Module Design and Simulation Based on VHDL", *Proceedings of 3rd International Workshop on Intelligent Systems and Applications*, pp. 28-32, 2011.

[11] PicoBlaze 8-bit Microcontroller, Available at: https://www.xilinx.com/products/intellectual-property/picoblaze.html.

[12] Summerville Douglas, "Embedded Systems Interfacing for Engineers using the Freescale HCS08 Microcontroller I: Machine Language Programming", Morgan and Claypool, 2009.

[13] G.A. Kumar and M.A. Sufhan, "FPGA Implementation of Picoblaze based Embedded System for Monitoring Applications", *International Journal of Science and Research*, Vol. 2, No. 3, pp. 1-8, 2013.

[14] J. Norhuzaimin and H.H Maimun, "The Design of High Speed UART", *Proceedings of Asia-Pacific Conference on Applied Electromagnetics*, pp. 121-128, 2005.

[15] B. Suresh, P. Teja Reddy, V.S.V. Srihari and S. Sivanantham, "ASIC Implementation of Low Power Universal Asynchronous Receiver Transmitter", *World Applied Sciences Journal*, Vol. 32, No. 3, pp. 472-477, 2014.