International Journal of Art & Humanity Science (IJAHS) e-ISSN: 2349-5235,
www.ijahs.com Volume 2 Issue 4, (July-August 2015), PP. 01-10

IMPLEMENTATION OF UART WITH STATUS REGISTER AND APB

Ku. Neeta Choubey, Mr. H. R. Singh
Pg Student-Vlsi Design,
Principal of Oriental University, Indore, M.P., India
neetacy50@gmail.com, paraag.ou@gmail.com

Abstract -Today in real world the actual application usually needed only a few key features of UART. Specific interface chip will cause of resources and increased cost. In parallel communication the cost as well as complexity of the system increases due to simultaneous transmission of data bits on multiple wires. Serial communication alleviates this drawback of parallel communication and emerges effectively in many applications for long distance communication as it reduces the signal distortion because of its simple structure. This paper focuses on the VHDL implementation of UART with status register which supports asynchronous serial communication. The paper presents the architecture of UART which indicates, during reception of data, parity error, framing error, overrun error and break error using status register. APB is used for solved interfacing problem. With APB other devised are easy to connect with UART. The proposed design of UART satisfies the system requirements of high integration, stabilization, low bit error rate, and low cost. It also supports configurable baud rate generator and variable data length from 5-8 bits per frame. For solving interfacing complexity used an APB.

Keywords- Universal Asynchronous Receiver Transmitter, status register, VHDL implementation, ISE simulator, asynchronous serial communication.

1. INTRODUCATION

UART is a type of "asynchronous receiver/ transmitter", a piece of computer hardware that translates data between parallel and serial forms. A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called USARTs.

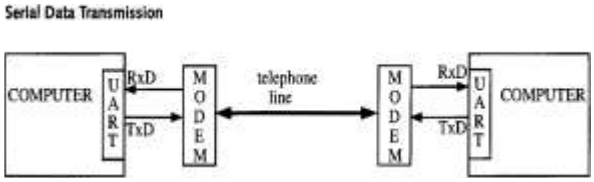


Figure 1 Serial Data Transmission

The above fig is standard format for serial transmission. Since no clock line, data is transmitted asynchronously, one byte at a time.

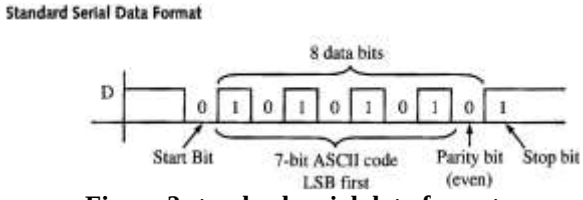


Figure 2 standard serial data format

When transmitting, the UART takes 8 bits of parallel data and converts the data to a serial bit stream that consists of a start bit (logic 0), 8 data bits (least significant bit first), and one or more stop bits (logic 1). UART controller is the key component of the serial communications subsystem of a computer. The UART 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. Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

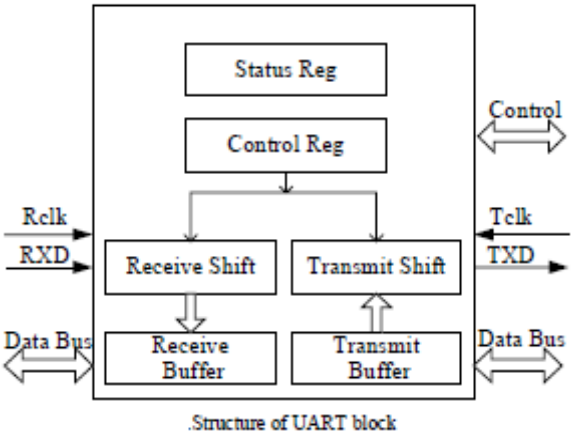


Figure 3. Structure of UART Block

2. VHDL IMPLEMENTATION

VHDL is commonly used as a design-entry language for field-programmable gate arrays and application-specific integrated circuits in electronic design automation of digital circuits. VHDL stands for *very high-speed integrated circuit* hardware description language. Which is one of the programming language used to model a digital system by dataflow, behavioral and structural style of modeling. The detail design of systems at the gate and flip-flop level has become time consuming as the integrated circuit technology has become very complex [7]. In recent years this fact has motivated usage of hardware description language in the design process of digital system. VHDL is used to describe and simulate the operation of variety of digital system which is ranging in complexity from a few gates to an interconnection of many complex integrated circuits. Advantages of VHDL implementation includes minimum cost and time, better design, faster time to market and increased flexibility [7].

3. PROPOSED UART ARCHITECTURE

UART supports asynchronous communication in which clock information is not shared between transmitter and receiver; several overhead bits are sent along with data bits for synchronization purpose. This indicates that data bits are transmitted in the form of frame. This frame is received at the receiver input where de-framing is done and only the data bits are available in parallel form at the receiver output.

The proposed design of UART, shown in Fig. 2, has LCR, Baud Rate Generator (BRG), Transmitter and Receiver as its functional units. All these blocks are explained in brief as course of rest of this section.

A. Advanced Peripheral Bus (APB)

UART is very useful and important for data transmission. In simple UART interface with one device so APB is provide an interfacing platform of UART. For this UART interface with many device. High-performance Bus (AHB) or the AMBA Peripheral Bus (APB). This application note focuses on the procedures, example, a size-optimized UART is added to the APB. , AMBA Host and Peripheral Buses. The APB signals interfacing, APB peripheral slaves. 8 Adding Custom Peripherals to the AMBA Host and Peripheral Buses. For communicating between smart cards and host systems using the AMBA Advanced Peripheral Bus (APB). , APB. SCR-APB Top Level Entity The wrapper that connects the SCR core to the AMBA APB. It converts, Reader Core for APB Performs functions needed for complete smart card sessions.

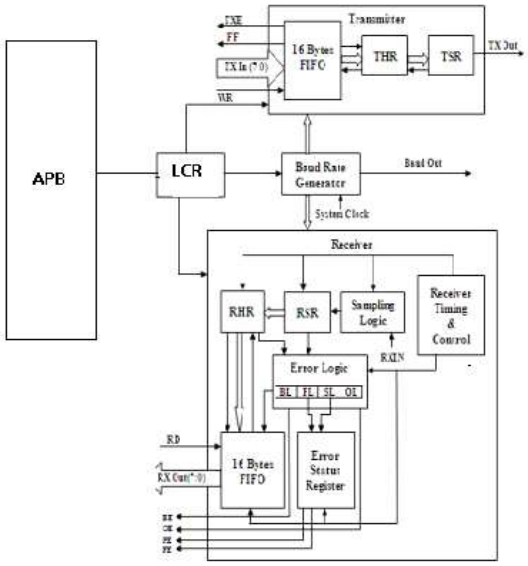


Figure 3. Proposed Uart Architecture

B. Line Control Register (LCR)

The line control register (LCR) is a byte register. It is used for precise specification of frame format and desired baud rate. The parity bits, stop bits, baud rate selection and word length can be changed by writing the appropriate bits in LCR, format of which is shown and indicates exact voltage levels of selection lines for selecting different baud rates, odd/even parity and data word length.

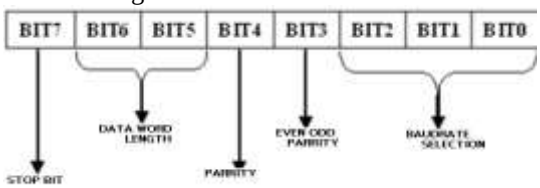


Figure 4 .LCR Format

C. Baud Rate Generator

The baud rate generator is programmable by three control bits Bit 0, Bit 1, Bit 2 in LCR as shown different baud rates can be selected by different combinations of Bit 0, Bit 1 and Bit 2. For the selective baud rate the divisor can be obtained by dividing the system clock by desired baud rate as shown which shows divisors for various baud rates using 10 MHz system clock.

Baud rate	Freq-10MHz	
	Divisor	%Error
600	8333	0.00
1200	4165	0.00
2400	2082	0.00
4800	1040	0.00
9600	520	0.00
19200	259	0.10
38400	129	0.26
57600	86	0.17

D. Transmitter

The transmitter section accepts parallel data, makes the frame of the data and transmits the data in serial form on the Transmitter Output (TXOUT) terminal.

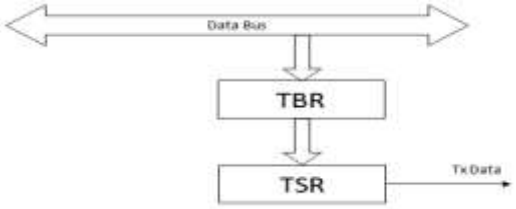


Figure 5 . Transmitter

Data is loaded from the inputs TXIN0-TXIN7 into the Transmitter FIFO by applying logic high on the WR (Write) input. If words less than 8 bits are used, only the least significant bits are transmitted. FIFO is 16-byte register. When FIFO contains some data, it will send the signal to Transmitter Hold Register (THR), which is an 8-bit register. At a same time, if THR is empty it will send the signal to FIFO, which indicates that THR is ready to receive data from FIFO. If Transmitter Shift Register (TSR) is empty it will send the signal to THR and it indicates that TSR is ready to receive data from THR. TSR is a 12-bit register in which framing process occurs. In frame start bit, stop bit and parity bit will be added. Now data is transmitted from TSR to TXOUT serially.

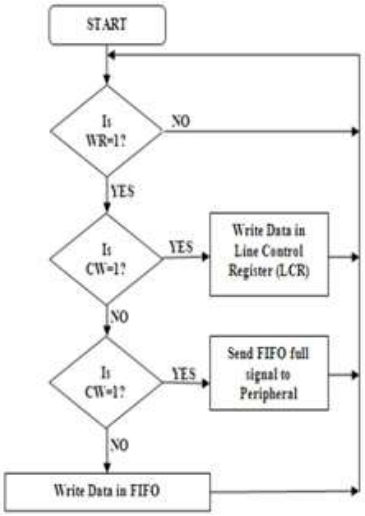


Figure 6 Transmitter flowchart (Input to FIFO)

Figure 6 shows the flowchart of transmitter with FIFO input.

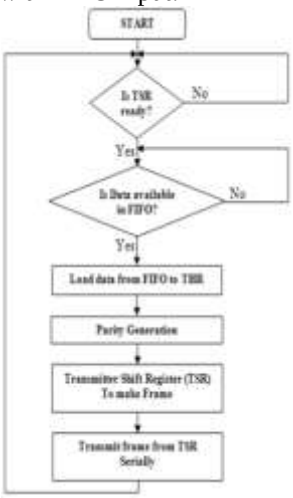


Figure 7. Transmitter flowchart (FIFO to Output)

E. Receiver

The transmitted data from the TXOUT pin is available on the RXIN pin. The received data is applied to the sampling logic block. The receiver timing and control is used for synchronization of clock signal between transmitter and receiver.

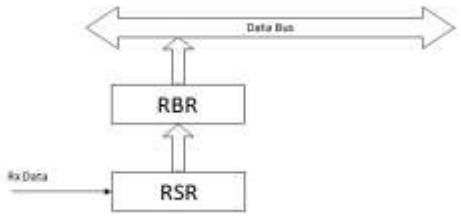


Figure 8. Receiver

Initially the logic line is high whenever it goes low sampling and logic block will take 4 samples of that bit and if all four are same it indicates the start of a frame. After that remaining bits are sampled in the same way and all the bits are send to Receiver Shift Register (RSR) one by one where the entire frame is stored. RSR is a 11 bit shift register. Now if the Receiver Hold Register (RHR) is empty it sends signal to RSR so that only the data bits from RSR goes to RHR which is an 8 bit register. The remaining bits in the RSR are used by the error logic block. Now if receiver FIFO is empty it send the signal to RHR so that the data bits goes to FIFO. When RD signal is asserted the data is available in parallel form on the RXOUT0-RXOUT7 pins. The error logic block handles 4 types of errors: Parity error, Frame error, Overrun error, break error. If the received parity does not match with the parity generated from data bits PL bit will be set which indicates that parity error occurred. If receiver fails to detect correct stop bit or when 4 samples do not match frame error occurs and SL bit is set. If the receiver FIFO is full and other data arrives at the RHR overrun error occurs and OL bit is set. If the RXIN pin is held low for long time than the frame time then there is a break in received data and break error occurs and BL bit is set.

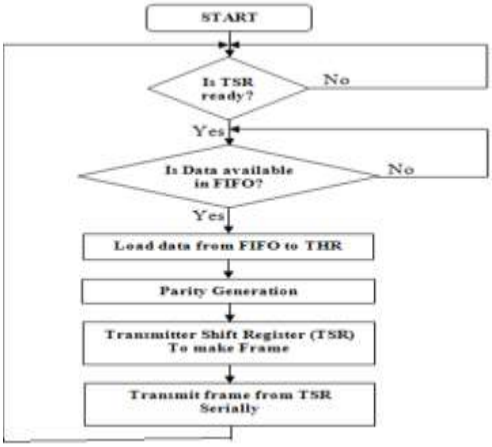


Figure 9: Receiver flowchart (Input to FIFO)

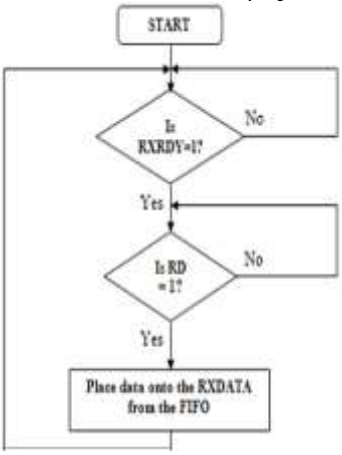


Figure 10: Receiver flowchart (FIFO to Output)

F. Status Register

The status register is a hardware register which contains information about the state of the processor. Individual bits are implicitly or explicitly read and/or written by the machine code instructions executing on the processor. The status register in a traditional processor design includes at least three central flags: Zero, Carry, and Overflow, which

are set or cleared automatically as side effects of arithmetic operations. They may then be tested via conditional branch or jump instructions. A status register may often have other fields as well, such as more specialized flags, interrupt enable bits, and similar types of information. During an interrupt, the status of the thread currently executing can be preserved (and later recalled) by storing the current value of the status register along with the program counter and other active registers into the machine stack or a reserved area of memory.

The Status Register takes the inputs from the error logic block and gives out four outputs that specify the Break error, overflow error, framing error and parity error.

1) Character framing:

The idle, no data state is high-voltage, or powered. This is a historic legacy from telegraphy, in which the line is held high to show that the line and transmitter are not damaged. Each character is sent as a logic low start bit(‘0’), a configurable number of data bits (usually 8, but legacy systems can use 5, 6, 7) an optional parity bit, and one or two logic high stop bits(‘1’).

2) Overrun error:

An "overrun error" occurs when the receiver cannot process the character that just came in before the next one arrives. Various devices have different amounts of buffer space to hold received characters. The CPU must service the UART in order to remove characters from the input buffer. If the CPU does not service the UART quickly enough and the buffer becomes full, an Overrun Error will occur, and incoming characters will be lost.

3) Framing error:

A "framing error" occurs when the designated "start" and "stop" bits are not valid. As the "start" bit is used to identify the beginning 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.

4) Parity error:

A "parity error" occurs when the number of "active" bits does not agree with the specified parity configuration of the USART, producing a *Parity Error*. Because the "parity" bit is optional, this error will not occur if parity has been disabled. Parity error is set when the parity of an incoming data character does not match the expected value.

5) Break condition or Break Error:

A "break condition" occurs when the receiver input is at the "space" level for longer than some duration of time, typically, for more than a character time. This is not necessarily an error, but appears to the receiver as a character of all zero bits with a framing error.

Some equipment will deliberately transmit the "break" level for longer than a character as an out-of-band signal. When signaling rates are mismatched, no meaningful characters can be sent, but a long "break" signal can be a useful way to get the attention of a mismatched receiver to do something (such as resetting itself). Unix-like systems can use the long "break" level as a request to change the signaling rate, to support dial-in access

4. SIMULATION AND SYNTHESIS RESULTS ANALYSIS

Simulation is the process of checking the codes for VLSI for a particular application to implement it in real time. Mainly the codes for VLSI are written using Xilinx software.

A. Baud Rate Simulation

In this simulation is present on 57600 burd rate clock is generate. In normal baud rate is fixed but in this project baud rate is not fixed.

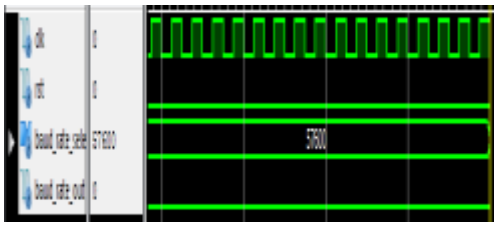


Figure 11. Baud rate simulation

B. Fifo Simulation

In this simulation in this simulation FIFO is use staring different data is high speed and reading odd low speed. When FIFO is full then data is transmit. Write enable is 1 that time wait for 3024 ns.

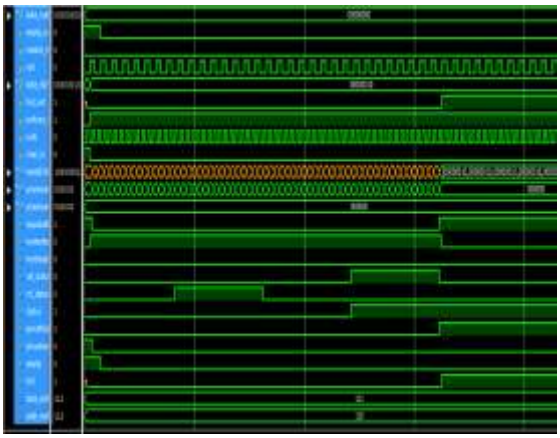


Figure 12 FIFO SIMULATION

C. Lcr Simulation

When input bit will be set to logic 1 when RBR data is valid and will be reset to logic 0 when RBR is empty. When line errors (OE/PE/FE/BI) happen, input bit will also be set to logic 1 and data will be updated to reflect the Data bits portion of the frame

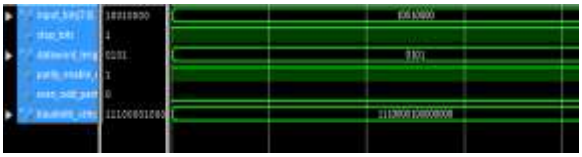


Figure 13 LCR SIMULATION

D. Transmitter Simulation

The simulation shows the transmission of an 8-bit UART frame format with 1 stop bit and without a parity bit. The transmission was set at 115.2kbps using 40MHz clock, which is equal to 25ns (1/40MHz = 25ns) period.

This simulation shows the signal results of the 8-bit data ("00000111B") transmission via DATA[7:0]. The transmitted UART frame format can be observed at TXD (1 low start bit, 8 data bits (LSB to MSB), and 1 high stop bit).

Figure 14 THR

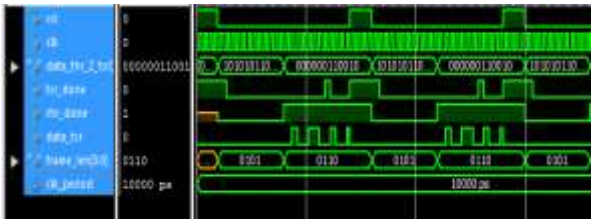


Figure 15 TSR

E. Receiver Simulation

The data consists of 4 high bits and 4 low bits. The data is then converted to parallel RXDATA at 93.01us. S_RXDATA is the internal parallel data received at the receiver. The output of the received parallel data is then routed to DATA[7:0] output's pin. The parallel data will be activated when the data output is enabled (data= '0'). The received data can be observed between the high impedance "DATA [7:0] (if the time/div is widened).

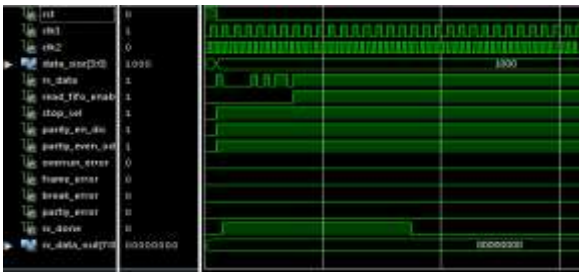
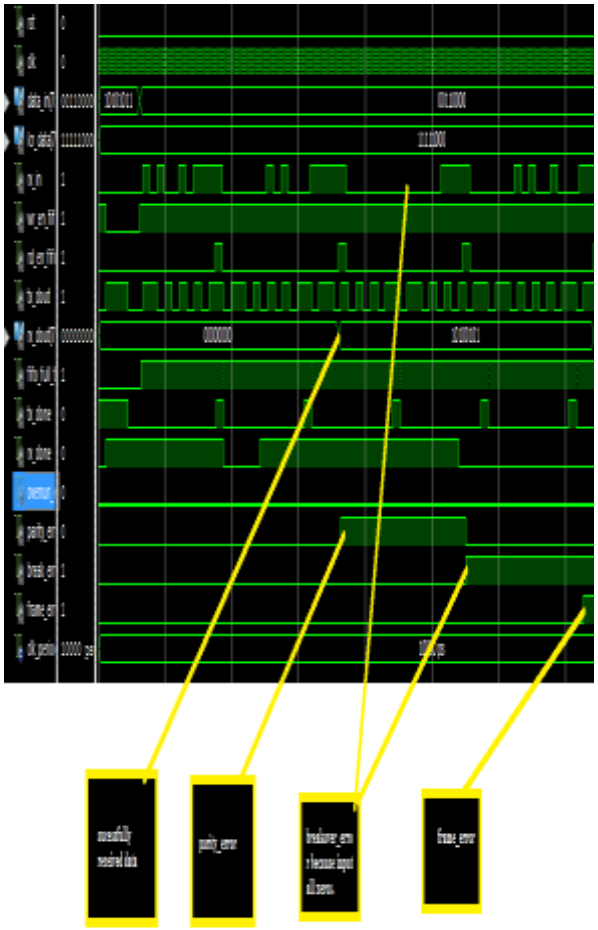
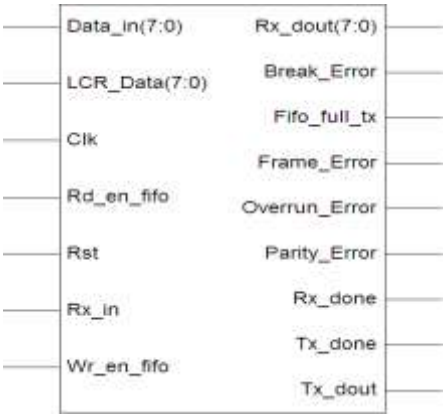


Figure 16 Receiver Simulation

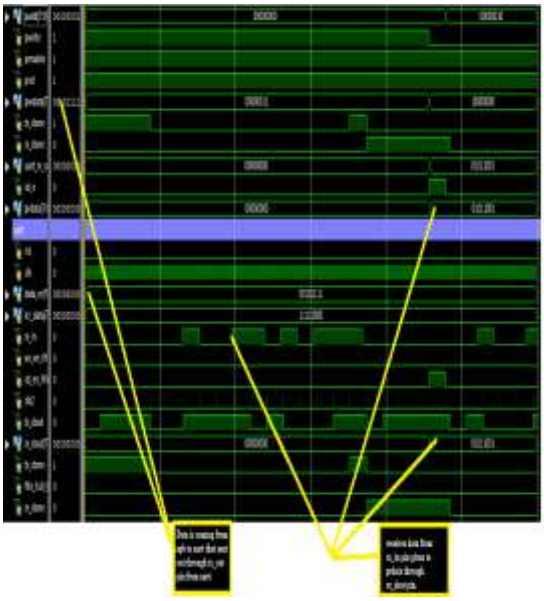
F. With Extension Simulation



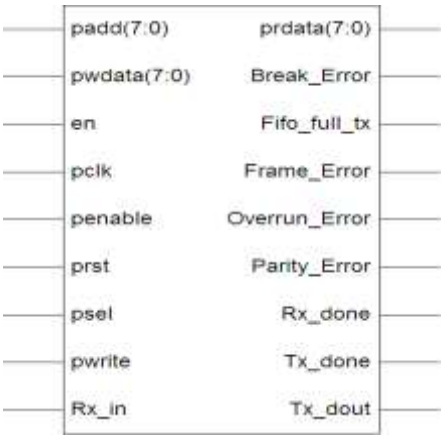
RTL UART module -1



G. With Extation Simulation



RTL UART module -2



5. CONCLUSION

This paper describes the architecture of UART that support various data word length, parity selection and different baud rates for serial transmission of data. Working principle of this UART has been tested using ISE simulator, which can be implemented on FPGA. Additionally we can detect the different types of errors occurred during communication and hence correct them.

- This design uses VHDL as design language to achieve the modules of UART.
- The results are stable and reliable.
- The design has great flexibility, high integration with some reference value.

In this project baud rate is variable. Detecting all error by error logic and use of status register correct all error.

6. ACKNOWLEDGMENT

I am highly grateful to the MR. H.R.SINGH Principal of Oriental University Indore, for providing this opportunity to carry out the Major project at VHDL Implementation of UART with Status Register.

I would like to expresses my gratitude to Mr. Jeewan Raddi HOD, Department of Electronics and communication OU, Mr. Parag Parandkar and other faculty members of Oriental University, Indore for providing academic inputs, guidance & encouragement throughout this period.

The author would like to express a deep sense of gratitude and thank my all friend, and specially my parents, without whose permission, wise counsel and able guidance, it would have not been possible to carry out my project in this manner.

Finally, I express my indebtedness to all who have directly or indirectly contributed to the successful completion of my major project.

REFERENCES

[1]. Elmenreich, W.; Delvai, M., "Time-triggered communication with UARTs," Factory Communication Systems, 2002. 4th IEEE Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[2]. Gallo, R.; Delvai, M.; Elmenreich, W.; Steininger, A., "Revision and verification of an enhanced UART," Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on, vol., no., pp. 315- 318, 22-24 Sept. 2004.
[3]. Norhuzaimin, J.; Maimun, H.H., "The design of high speed UART," Applied Electromagnetics, 2005. APACE 2005. Asia-Pacific Conference on, vol., no., pp.5 pp., 20-21 Dec. 2005.
[4]. Himanshu Patel; Sanjay Trivedi; R. Neelkanthan; V. R. Gujraty; , "A Robust UART Architecture Based on Recursive Running Sum Filter for Better Noise Performance," VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, vol., no., pp.819-823, Jan. 2007.
[5]. Fang Yi-yuan; Chen Xue-jun; , "Design and Simulation of UART Serial Communication Module Based on VHDL," Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on , vol., no., pp.1-4, 28-29 May 2011
[6]. Yongcheng Wang; Kefei Song; , "A new approach to realize UART," Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on , vol.5, no., pp.2749- 2752, 12-14 Aug. 2011.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
[7]. Idris, M.Y.I.; Yaacob, M., "A VHDL implementation of BIST technique in UART design," TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region , vol.4, no., pp. 1450- 1454 Vol.4, 15-17 Oct. 2003.