Implementation of serial communication using UART with configurable baud rate

Ms.Neha R. Laddha[*]
ME-DE.
Sipna C.O.E.T, Amravati (MS) INDIA
nehaladdha4@gmail.com

Prof.A.P.Thakare
Professor and
Head of Electronics & Telecomm.Deptt.
Sipna C.O.E.T,Amravati (MS) INDIA

Abstract:- Today in real world the actual applications, usually needed only a few key features of UART. 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. Universal Asynchronous Receiver Transmitter (UART) is a kind of serial communication protocol. 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. The UART implemented with VHDL language can be integrated into the FPGA to achieve compact, stable and reliable data transmission. This paper presents implementation of Multi UART with configurable baud rate. Also we can verify the output using LED's on Altera's DE1 board.

***Keywords*:- UART; Asynchronous serial communication; VHDL; Quartus II; simulation, Altera DE1 Cyclone II board.**

*****

I. INTRODUCTION

Universal Asynchronous Receiver Transmitter (UART) is a kind of serial communication protocol; mostly used for short-distance, low speed, low-cost data exchange between computer and peripherals. UARTs are used for asynchronous serial data communication by converting data from parallel to serial at transmitter with some extra overhead bits using shift register and vice versa at receiver. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible. It is generally connected between a processor and a peripheral, to the processor the UART appears as an 8-bit read/write parallel port. The UART implemented with VHDL language can be integrated into the FPGA to achieve compact, stable and reliable data transmission [3]. Various designs are found in literatures for UART as different systems have different requirements and attributes which require data communication between its functional units. In recent years the researchers have proposed various UART designs like automatic baud rate synchronizing capability[2], predictable timing behavior to allow the integration of nodes with imprecise clocks in time-triggered real-time systems[4], recursive running sum filter to remove noisy samples[2], integration of only core functions into a FPGA chip to achieve compact, stable and reliable data transmission to avoid waste of resources and decrease cost[1], programmable logic to enable interfacing between asynchronous communications protocols and DSP having synchronous serial ports.

Basic UART communication needs only two signal lines (RXD, TXD) to complete full-duplex data communication. TXD is the transmit side, the output of UART; RXD is the receiver, the input of UART. UART's basic features are: There are two states in the signal line, using logic 1 (high) and logic 0
(low) to distinguish respectively. For example, when the transmitter is idle, the data line is in the high logic state. Otherwise when a word is given to the UART for asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

After the Start Bit, the individual data bits of the word are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver "looks" at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter. When the receiver has received all of the bits in the data word, 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.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host. If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent. Because asynchronous data are "self synchronizing", if there are no data to transmit, the transmission line can be idle. The UART frame format is shown in Fig. 1.

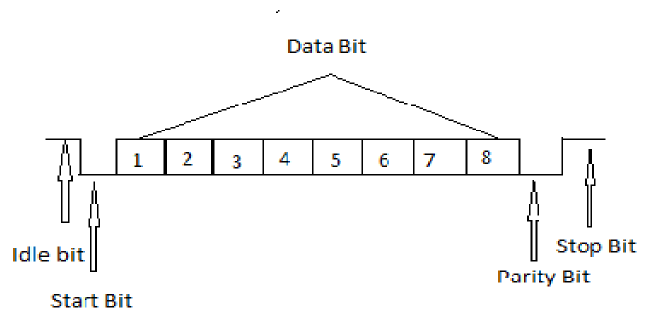


Fig 1. UART Frame Format

In this paper, we present UART which includes three modules which are the baud rate generator, receiver and transmitter. 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 with data length of 8 bits per frame. The configurable baud rate generator has been implemented using two switches of cyclone II FPGA in DE1 Board which specifies the baud rate of input and accordingly input data can verified with start bit and stop bit on LED's of cyclone II FPGA in DE1 Board.

II. FPGA:

FPGA or Field Programmable Gate Arrays can be programmed or configured by the user or designer after manufacturing and during implementation. Hence they are otherwise known as On-Site programmable. Unlike a Programmable Array Logic (PAL) or other programmable device, their structure is similar to that of a gate-array or an ASIC. Thus, they are used to rapidly prototype ASICs, or as a substitute for places where an ASIC will eventually be used. This is done when it is important to get the design to the market first. The programming of the FPGA is done using a logic circuit diagram or a source code using a Hardware Description Language (HDL) to specify how the chip should work. FPGAs have programmable logic components called ‚logic blocks‛, and a hierarchy or reconfigurable interconnects which facilitate the ‚wiring‛ of the blocks together. The programmable logic blocks are called configurable logic blocks and reconfigurable interconnects are called switch boxes. Logic blocks (CLBs) can be programmed to perform complex combinational functions, or simple logic gates like AND and XOR. In most FPGAs the logic blocks also include memory elements, which can be as simple as a flip-flop or as complex as complete blocks of memory. FPGAs that store their configuration internally in nonvolatile flash memory, such as Micro semi's ProAsic 3 or Lattice's XP2 programmable devices, do not expose the bitstream and do not need encryption. In addition, flash memory for LUT provides SEU protection for space applications. Advantages of FPGA is as follows:

- Ability to re-program in the field to fix bug.
- Fast prototyping and turn-around time.
- NRE cost is zero.
- High-Speed.
- Low cost.

III. ALTERA DE1 DEVELOPMENT AND EDUCATIONAL BOARD

Altera DE1 board become one of the most widely development FPGA board which is used to development of FPGA design and implementations. The purpose of the Altera DE1 Development and Education board is to provide the ideal vehicle for learning about digital logic, computer organization, and FPGAs. It uses the state-of the art technology in both hardware and CAD tools to expose researchers and professionals to a wide range of topics. The board offers a rich set of features that make it suitable for research work Altera provides a suite of supporting materials for the DE1 board, including tutorials, "ready-to-teach" laboratory exercises, and illustrative demonstrations [Altera] Figure (2) gives the block diagram of the DE1 board. To provide maximum flexibility for the user, all connections are made through the Cyclone II FPGA device. Thus, the user can configure the FPGA to implement any system design.

The DE1 board features a state-of-the-art Cyclone® II 2C20 FPGA in a 484-pin package. All important components on the board are connected to pins of this chip, allowing the user to control all aspects of the board's operation. For simple experiments, the DE1 board includes a sufficient number of robust switches (of both toggle and push-button type), LEDs, and 7-segment displays. For more advanced experiments, there are SRAM, SDRAM, and Flash memory chips. For experiments that require a processor and simple I/O interfaces, it is easy to instantiate Altera's Nios II processor and use interface standards such as RS-232 and PS/2. For experiments that involve sound or video signals, there are standard connectors for microphone, line-in, line-out (24-bit audio CODEC), SD memory card connector, and VGA; these features can be used to create CD-quality audio applications and video. Finally, it is possible to connect other user defined boards to the DE1 board by means of two expansion headers.

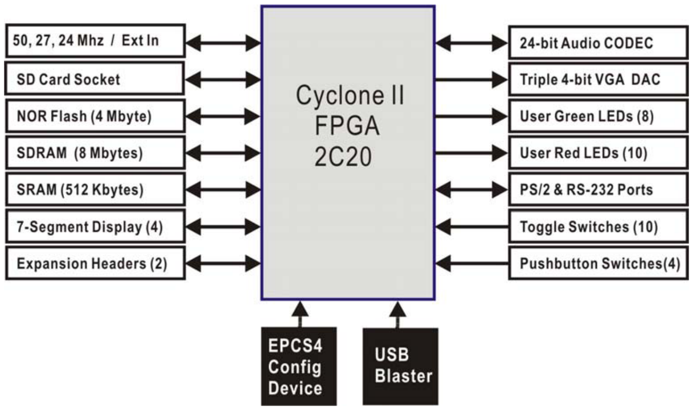


Fig 2. Cyclone II FPGA 2C20

IV. IMPLEMENTATION OF MULTI-UART.

The UART serial communication module is divided into three sub-modules: the baud rate generator, receiver module and transmitter module, shown in Fig. 3. 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.

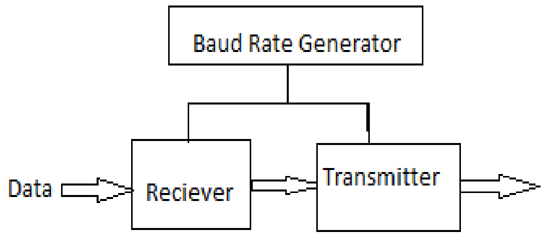


Fig 3. UART Module

A. Baud Rate Generator

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. The calculated baud rate frequency factor is used as the divider factor.

Assume that the system clock is 50MHz, baud rate is 9600bps, and then the output clock frequency of baud rate generator should be 1* 9600Hz.

Therefore the frequency coefficient (M) i.e. counts value of the baud rate generator is:

M =50MHz/1*9600Hz=5208

When the UART receives serial data, it is very critical to determine where to sample the data information. The ideal time for sampling is at the middle point of each serial data bit.

In this design, we designed configurable baud rate generator which can selected by using two switch of cyclone II DE1 board. The selection of baud rate is as shown.

S0	**S1**	**Baud Rate(BPS)**
0	0	2400
0	1	4800
1	0	9600
1	1	19200
Default		9600

Hence, different baud rate has different frequency coefficient (M) i.e. count value of the baud rate generator.
For 2400Hz, M =50MHz/1*2400Hz=20833
For 4800Hz, M =50MHz/1*2400Hz=10416
For 9600Hz, M =50MHz/1*2400Hz=5208
For 19200Hz, M =50MHz/1*2400Hz=2604

B. Receiver Module

During the UART reception, the serial data and the receiving clock are asynchronous, so it is very important to correctly determine the start bit of a frame data. The receiver module receives data from RXD pin. RXD jumps into logic 0 from logic 1 can be regarded as the beginning of a data frame. When the UART receiver module is reset, it has been waiting the RXD level to jump. As we know, the ideal time for sampling is at the middle point of each serial data bit. Hence, RXD low level lasts at least half of receiving clock cycles is considered start bit arrives. Once the start bit been identified, from the next bit, begin to count the rising edge of the baud clock, and sample RXD when counting. Each sampled value of the logic level is deposited in the register parallel_data_signal [7, 0] by order. When the count equals 10, all the data bits are surely received, also the 10 serial bits are converted into a byte parallel data and deposited in the resister parallel_data.

The state machine includes five states: R_Initial (waiting for the start bit), R_Center (find midpoint), R_Delay (Waiting for the sampling), R_Shift register (sampling), and R_Stop (receiving stop bit).

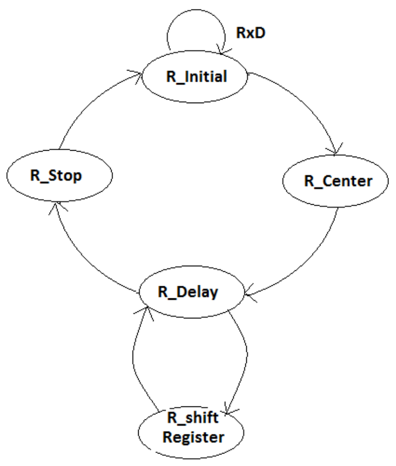


Fig 4. Reciever FSM

R_Initial Status: Initially, the UART receiver is reset; the receiver state machine will be in this state. In this state, the state machine has been waiting for the RXD level to be at logic 0, i.e. the start bit. Start bit indicates the beginning of a new data frame. Once the start bit is identified, the state machine will be transferred to Center state. Also, In this state, start_flag is set to 1.

R_Center Status: For asynchronous serial signal, in order to detect the correct signal each time, and minimize the total error in the later data bits detection. Obviously, it is the most ideal to detect at the middle of each bit. In this state, the task is to find the midpoint of each bit through the start bit. The method is by counting the number of clk_count.

R_Delay Status: When the state machine is in this state, waiting for counting to reach final count value, then entering into shift register to sample the data bits. At the same time determining the received data in 10 bit with start and stop bit.

R_Shift register Status: In this state, data bits are sampled and stored it into shift register which is of 8 bit. After sampling the state machine transfers to delay state unconditionally, waits for the arrival of the next start bit. Also in this state, the start_flag is reset.

R_Stop Status: when stop bit is 1. State machine doesn't detect RXD in stop. After the stop bit, state machine turns back to R_START state, waiting for the next frame start bit.

C. Transmit Module

The function of transmit module is to convert the sending 8-bit parallel data into serial data, adds start bit at the head of the data as well as stop bits at the end of the data. When the UART transmit module is reset by the reset signal, the transmit module immediately enters the ready state to send. In this state, the 8-bit parallel data is read into the Shift register [7: 0]. The transmitter only needs to output 1 bit every bit_count. The order follows 1 start bit, 8 data bits and 1 stop bit. Fig.5 shows the transmit module state diagram. This state machine has 5 states: X_Initial (free), X_Start (start bit), X_Delay (shift to wait), X_Shift register (shift), X_Stop (stop bit).

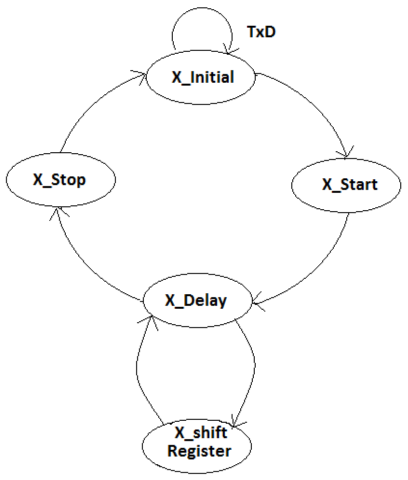


Fig 5. Transmitter FSM

X_Initial Status: When the UART is reset, the state machine will be in this state. In this state, the UART transmitter has been waiting a data frame. When data frame is arrived, the state machine transferred to Start , get ready to send start bit. Start bit indicates the beginning of a new data frame.

X_Start Status: In this state, sends a logic 0 signal to the TXD for one bit time width, the start bit. Then the state machine transferred to delay state.

X_Delay Status: Similar with the R_delay state of UART receive state machine.

X_Shift register Status: In this state, the state machine realizes the parallel to serial conversion of outgoing data. Then immediately return to X_delay state.

X_Stop Status: Stop bit transmit state. When the data frame transmit is completed, the state machine transferred to this state, and sends logic 1 signal, that is, 1 stop bit. The state machine turns back to X_Initial state after sending the stop bit, and waits for another data frame to transmit.

V. OVERALL SIMULATION OF MODULES

The simulation software is ModelSim. Fig. shows simulation of UART as a single entity. From simulation waveform, we select baud rate as 19200 using "sel" pin as "11" , as changes happen on "rxd" lines that serial data is stored in shift register and gives parallel data which contains start bit, data bit(8) and Stop bit.

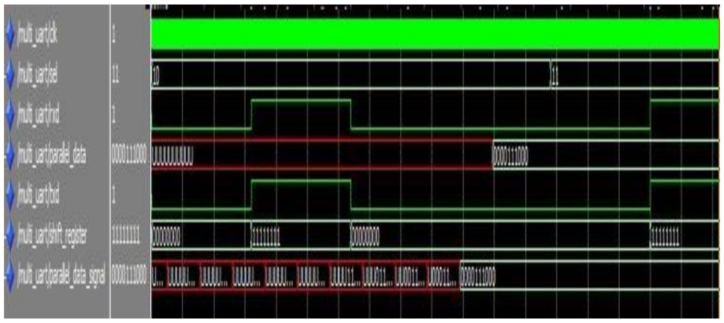
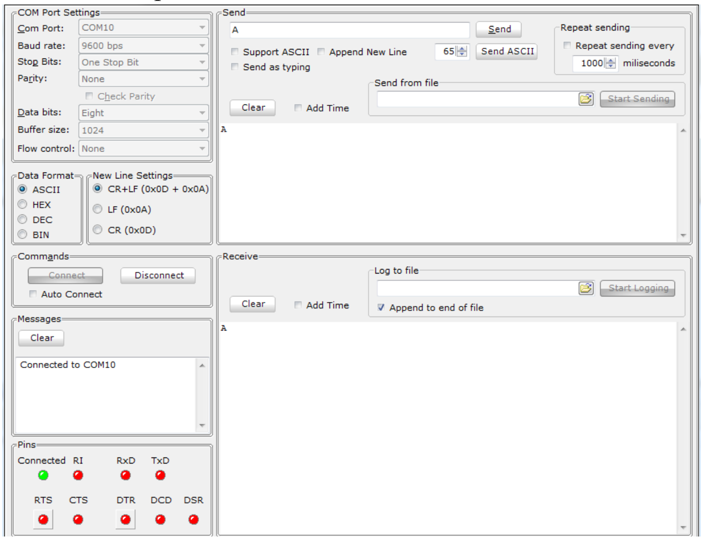


Fig 6. Simulation Result

VI. RESULTS:

The synthesis is done with Quartus II. After synthesis, code is burn on Altera's DE1 Cyclone II FPGA: EPC215AF484C7 board. In this design instead of using PC hyper terminal, USART hyper terminal of micro C AVR software is used, for any key board character transmission and reception. Considering first input shown below. Here the selected baud rate is 9600 bps to send data 'A'.

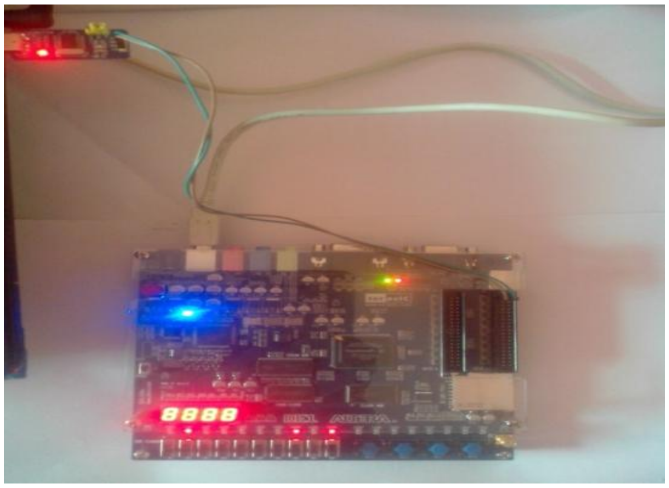

COM Port Settings
Com Port: COM10
Baud rate: 9600 bps
Stop Bits: One Stop Bit
Parity: None
Check Parity
Data bits: Eight
Buffer size: 1024
Flow control: None

Data Format: [✓ ASCII] HEX DEC BIN
New Line Settings: [✓ CR+LF (0x0D + 0x0A)] LF (0x0A) CR (0x0D)

Send: A
Support ASCII Append New Line 65 Send ASCII
Send as typing
Repeat sending: Repeat sending every 1000 miliseconds
Send from file
Clear Add Time
A

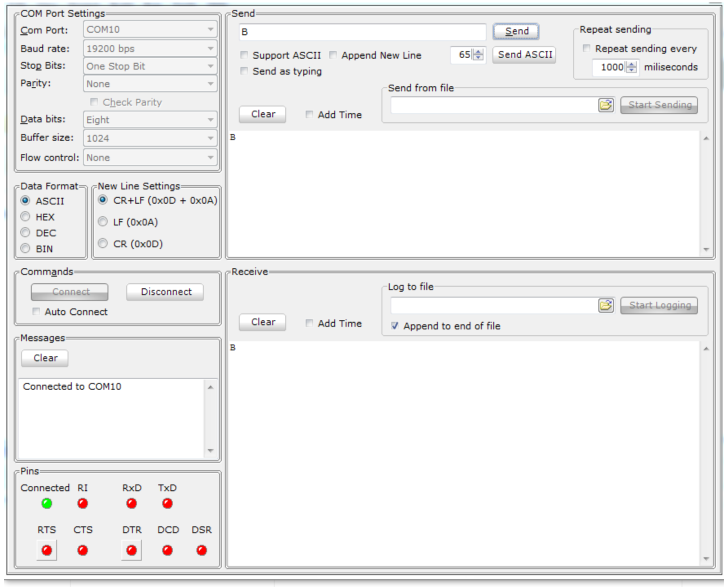
Receive
Log to file
Clear Add Time [✓ Append to end of file]
A

Messages
Connected to COM10



Since, the baud rate is 9600 bps, we have to select switch 's0' as "1" & 's1' as "0" which specified that the selected baud rate on Altera DE1 is 9600 bps which means that both baud rate are synchronized. The input data send by USART hyper terminal of micro C AVR software is 'A' which as a binary value of 8 bit '01000001'. The corresponding output is shown on FPGA board using respected LED's with start bit and stop bit. The output for data is shown in following figure.

Considering second input shown below. Here the selected baud rate is 19200 bps to send data 'B'.


COM Port Settings
Com Port: COM10
Baud rate: 19200 bps
Stop Bits: One Stop Bit
Parity: None
Check Parity
Data bits: Eight
Buffer size: 1024
Flow control: None

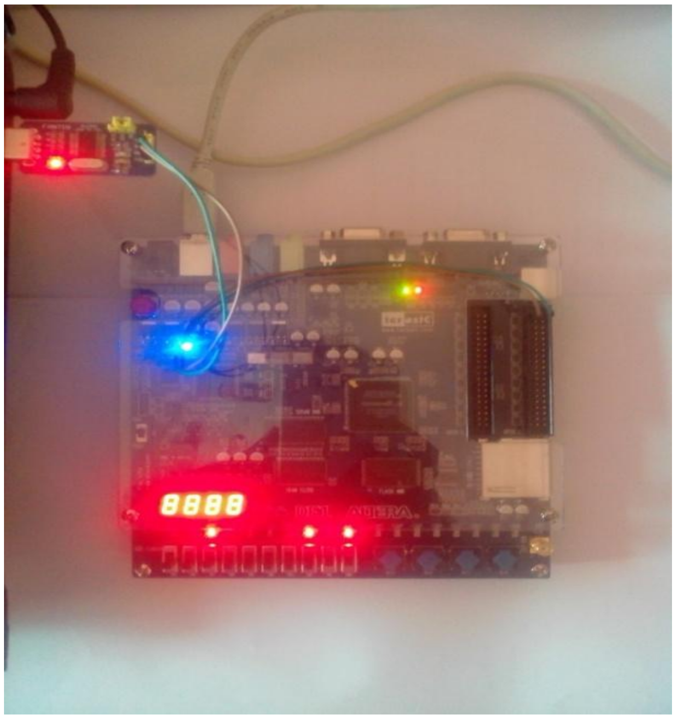
Data Format: [✓ ASCII] HEX DEC BIN
New Line Settings: [✓ CR+LF (0x0D + 0x0A)] LF (0x0A) CR (0x0D)

Send: B
Support ASCII Append New Line 65 Send ASCII
Send as typing
Repeat sending: Repeat sending every 1000 miliseconds
Send from file
Clear Add Time
B

Receive
Log to file
Clear Add Time [✓ Append to end of file]
B

Messages
Connected to COM10

Since, the baud rate is 19200 bps, we have to select switch ‘s0’ as “1” &’ s1’ as “1” which specified that the selected baud rate on Altera DE1 is 19200 bps which means that both baud rate are synchronized. The input data send by USART hyper terminal of micro C AVR software is ‘B’ which as a binary value of 8 bit “01000010”. The corresponding output is shown on FPGA board using respected LED’s with start bit



and stop bit. The output for data is shown in following figur

VII. CONCLUSION:

This paper describes the architecture of Multi UART that supports 8 bit data word length, start bit as well as stop bit and configurable baud rates for serial transmission of data. Working principle of this UART has been tested using Modelsim simulator and quartus II is used for synthesis. Additionally we can verify the output using LED’s on Altera DE1 board.

VIII. ACKNOWLEDGMENT:

We would like to acknowledge the Faculties of Electronics & Telecommunication Department, Sipna College of Engineering & Technology, Amravati for their support. I Neha R. Laddha specially want to thank my guide Prorfessor and H.O.D A.P.Thakare sir for their valuable guidance and constant encouragement towards the work.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] Elmenreich, W.; Delvai, M.; , "Time-triggered communication with UARTs," Factory Communication Systems, 2002. 4th IEEE International Workshop on , vol., no., pp. 97- 104, 2002J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2.Oxford: Clarendon, 1892, pp.68–73.