Design and Implementation of High-Speed Universal Asynchronous Receiver and Transmitter (UART)

Ashok Kumar Gupta, Ashish Raman, Naveen Kumar, and Ravi Ranjan

***Abstract*— In this paper, a universal asynchronous receiver and transmitter (UART) are described, which is basically a serial data transmission protocol used in digital circuit applications. The architecture of the UART transmitter has a baud rate generator, parity generator, transmitter finite state machine (FSM) and parallel in serial out register (PISO). UART receiver has a baud rate generator, negative edge detector, parity checker, receiver finite state machine (FSM) and serial in parallel out (SIPO) register. The baud rate generator of both transmitter and receiver is the same, so the baud rate of transmitter/receiver is the same. Baud rate generator is the same as the frequency divider circuit. The data frame of the UART transmitter is 1 start bit, 8 transmits data bits, 1 parity bit and 1 stop bit. The baud rate of the transmitter and receiver is 4 Mbps using the system clock of 64 MHz's. Implementation, simulation, and synthesis is done Xilinx Vivado 2016.2 version tool. The design is verified using simulated waveform and synthesized on the FPGA Zed board.**

***Index Terms*—UART, Asynchronous Circuit, Baud Rate Generator, High-Speed UART, Negative Edge Detector, Parity Generator, Universal Asynchronous Receiver Transmitter, Serial Communication Protocol.**

I. Introduction

UART stands for a universal asynchronous receiver and transmitter. It is a serial data transmission protocol which is used to communicate device serially. UART is mostly used for a short distance, low speed and the minimum cost is required. UART works in various modes simplex, half-duplex and full-duplex [1]. In simplex only one-way transmission is happening, in half-duplex either of transmitter or receiver only one of them is communicating at a time, if the transmitter transmits data then receiver in idle state wise-versa. In a full-duplex mode, both transmitter and receiver are works at the same time. UART is used as a bridge in slow-speed peripherals and high-speed peripheral devices [2]. UART has basically two parts one is a transmitter and the other is a receiver. The transmitter takes the parallel data from the computer and transmits the data serially. The receiver takes the data serially from the device and sends it to computer parallel fashion. Asynchronous word is used in the UART because both transmitter and receiver do not work at the same clock. To synchronize the received data there is no clock is required. The transmitter and receiver of the UART worked as the same baud rate. Baud rate is the unit data transfer through the channel in the communication protocol. Its unit is bits per second (bps). UART basically works as 1200, 2400, 4800, 9600, and 115200 bps. For successful transmission of data using UART, the baud rate of both transmitter and receiver should be the same [3]-[4]. When transmits the data transmitter make the data frame, which adds the start bits, stop bits and parity bits. The parity bit is optional which depends upon the designer's requirement [5]-[9].

In this work, the architecture of the transmitter and receiver block is implemented, simulated and synthesized using Verilog hardware descriptive language. The architecture and pin description of the UART transmitter and receiver are shown below.

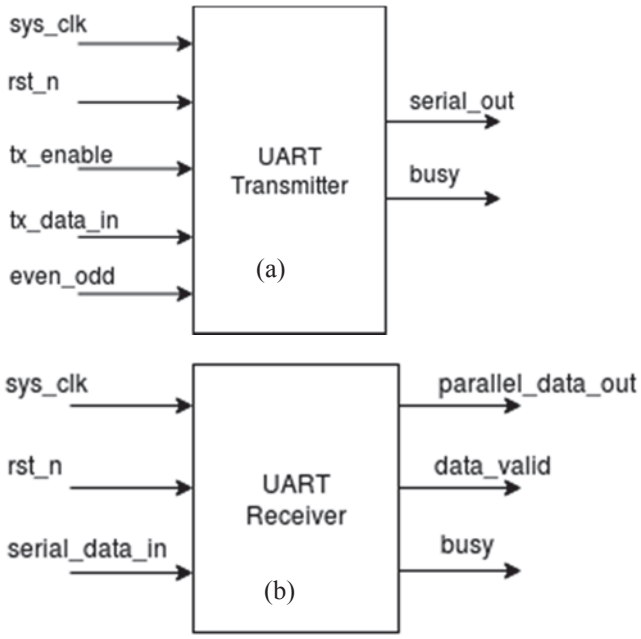

Fig.1 Interface diagram of UART Protocol (a) Transmitter of UART (b) Receiver of UART

Fig. 1 shows the interface diagram of both the transmitter and receiver of the UART protocol. Fig.1 (a) shows the interface diagram of the transmitter which has a system clock, an active-low reset signal, transmits enable, transmits input data signal, even or odd parity select signal as the input signal and busy and serial out data as the output signal. Serial data input is 8 bits data and remaining is 1 bit. Fig. 1(b) shows the interface diagram of the UART receiver which has system clock, active low reset, serial data in as the input and parallel data out, data valid and busy signal as the output. Parallel data output is 8 bits data and remaining signal is 1-bit data.

978-1-7281-5475-6/20/$31.00 ©2020 IEEE

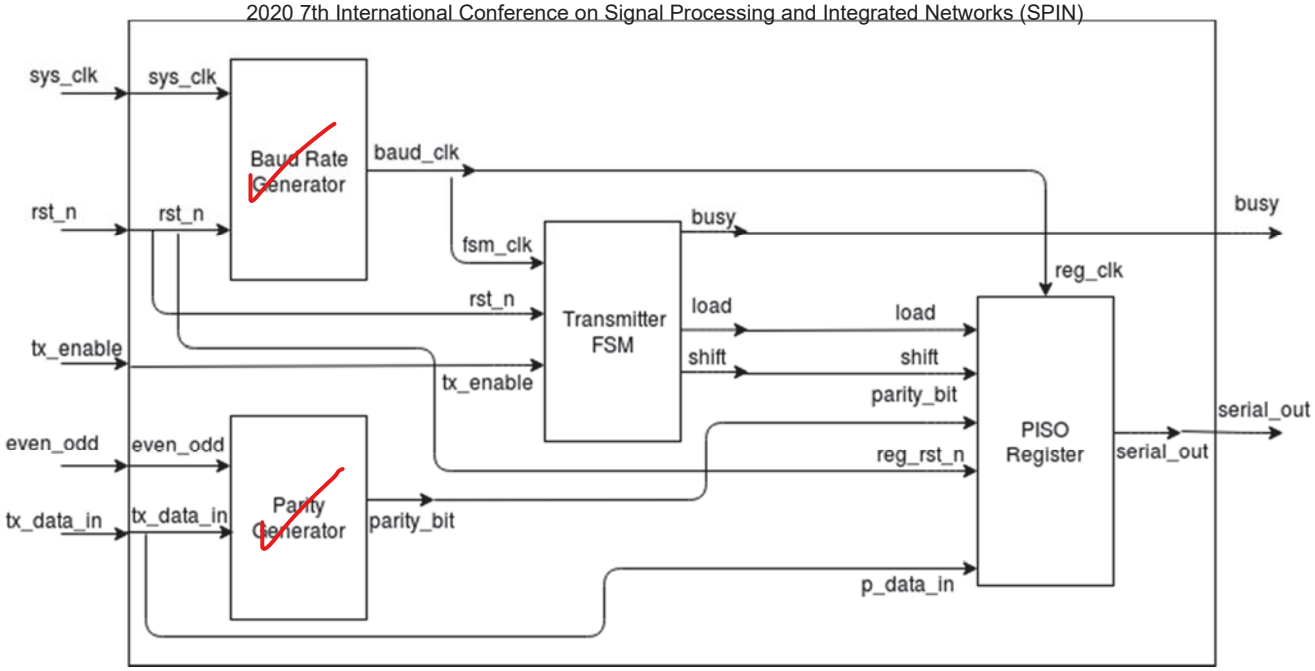

Fig. 2 Proposed Architecture of UART Transmitter

II. UART DATA FRAME AND DATA TRANSMISSION

When data is transmitted through the UART transmitter, the first data frame is created. Transmitter sends the data serially before it adds 1 start bit, 1 parity bit and 1 stop bit. So, the total data frame of 11 bits (including 8-bits transmitting input data bits). In reset conditions, serial data transmission is high

Fig. 3 Data Frame of UART Transmitter

logic i.e. 1. During transmission first start bit is sent i.e. logic low (0) is sent and after that data is transmitted, first LSB is sent and then parity bit and stop bit i.e. logic High (1) send and after that 1 continuously send.

III. (A) PROPOSED ARCHITECTURE OF UART TRANSMITTER

In this section, the architecture of the UART transmitter is described. Fig. 2 describe the architecture of the UART transmitter. UART transmitter architecture is the integration of the following modules.

(A) Baud Rate Generator: Baud rate generator is nothing a frequency divider circuit. This module has a system clock (sys_clk), active low reset (rst_n) as an INPUT and output clock (baud_clk) as an OUTPUT. The main purpose of this module is to generate the baud rate of 4 Mbps with a system clock of 64MHz.

(B) Parity Generator: Parity generator is used to generate the parity bit. Which is useful for to check the output is correct or not at the receiver side. There are two types of parity, even parity, and odd parity. In this paper, odd parity is used to make the data frame during the transmission. To make odd parity, the input signal even_odd make logic high (1), take the xor operation on transmit input data to generate the parity bit like the output.

(C) Transmitter FSM: Transmitter FSM is used to change the state of the transmitter. It has baud clock (baud_clk), active low reset (rst_n) and transmits enable signal (tx_enable) signal as the input signal and busy, load and shift as the output of the transmitter FSM. The load and shift signal controls the PISO shift registers. Transmitter FSM has four states i.e. IDLE, LOAD, SHIFT and WAIT. WAIT is a dummy state

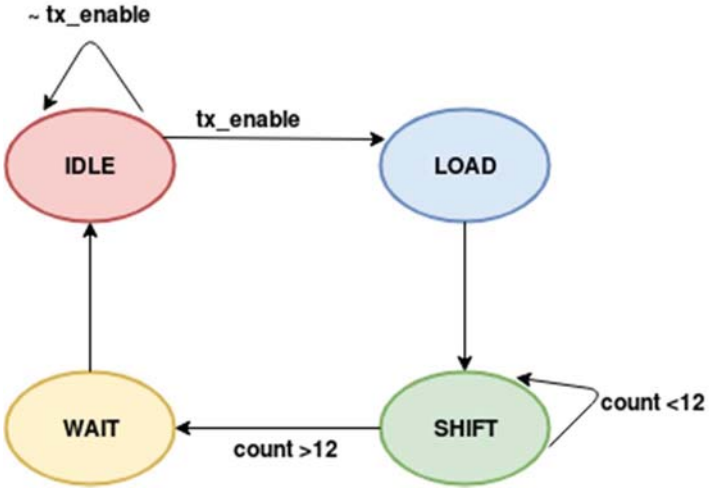

Fig. 4 Finite state machine (FSM) of UART Transmitter

which is used to clean the value. IDLE is the default state when the transmitter stays during a reset condition (rst_n). When transmission start (tx_enable is HIGH) the transmitter moves to LOAD state where data is loaded and the data frame is created. Further next clock (fsm_clk) i.e. baud clock (baud_clk) transmitter is the move to SHIFT state and data are transmitted serially until all the data is transmitted.

(D) PISO Register: Parallel input serial output register used for transmits the data serially. Clock (reg_clk same as baud_clk), active low reset (rst_n), load, shift and input data that is transmitted (tx_data_in) is the input of PISO register and serial out (serial_out) is the output of the PISO register. When load signal is high then the data frame is created means start bit,

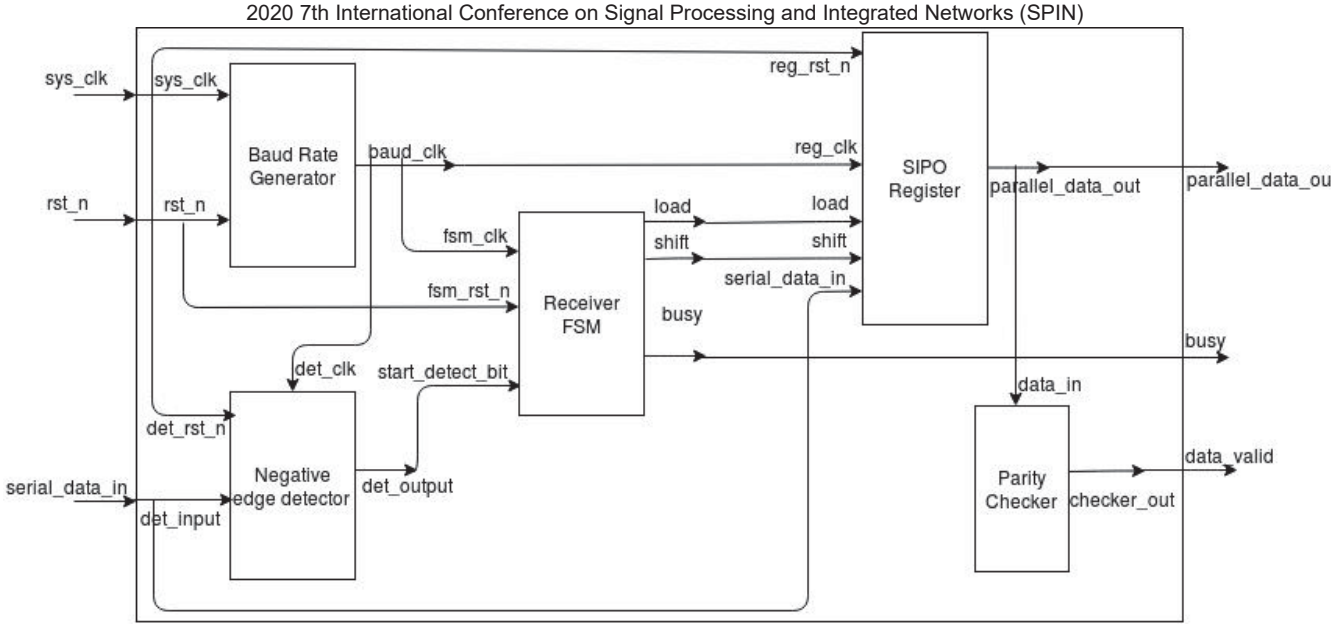

Fig. 5 Proposed Architecture of UART Receiver

parity bit and stop bit is added with transmit data bit (tx_data_in) and when load signal is high data is transmitted serially.

III. (B) PROPOSED ARCHITECTURE OF UART RECEIVER

In this section architecture of the UART, the receiver is described. Fig. 5 describes the architecture UART receiver which includes the burd rate generator, negative edge detector, receiver finite machine (FSM), serial in parallel out (SIPO) register and parity generator. These modules are described below.

(A) Baud Rate Generator: Baud rate generator is nothing a frequency divider circuit. This module has a system clock (sys_clk), active low reset (rst_n) as an INPUT and output clock (baud_clk) as an OUTPUT. The main purpose of this module is to generate the baud rate of 4 Mbps with a system clock of 64MHz. The baud rate of the receiver is the same as the baud rate of the receiver.

(B) Negative Edge Detector: Negative edge detector is used for the detection of the start bit of the transmission data frame. Before the transmission transmits the data default signal is logic high. UART receiver receives the serial bits when the start bit comes then the signal goes from logic high to logic low. The negative edge detector is used to detect the start bit. After detecting the start bit state of receiver changes and store the transmission data. The negative edge detector is designed using a combination of D flip-flop and AND gate.

(C) Receiver FSM: Receiver FSM is used to change the state of the receiver. It has baud clock (baud_clk), active low reset (rst_n) and start to detect bit (start_detect_bit) signal as the input signal and busy, load and shift as the output of the receiver FSM. The load and shift signal controls the SIPO shift registers. Receiver FSM has four states i.e. IDLE, LOAD, SHIFT and WAIT. WAIT is a dummy state which is used to clean the value. IDLE is the default state when the receiver is staying during the reset condition (rst_n). The receiver starts when it detects the start bit (start_detect_bit) which is generated from the negative edge detector module. The transmitter moves to SHIFT state where shifting operation takes place until all the bits are not received. Further next clock (fsm_clk) i.e. baud clock (baud_clk) receiver movies from SHIFT state to LOAD state. Data is loaded after

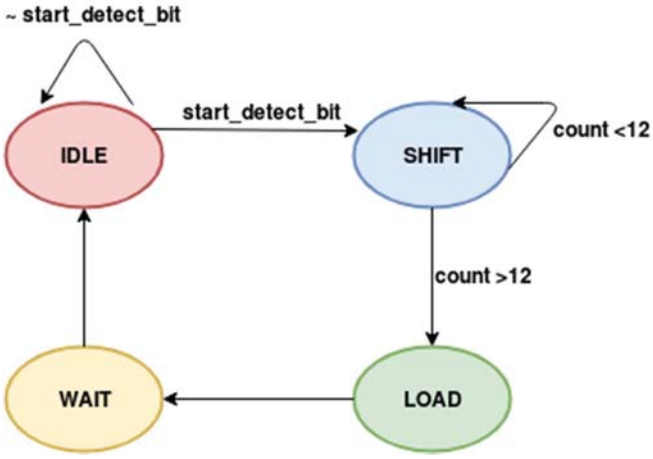

Fig. 6 Finite state machine (FSM) of UART Receiver

removing a start bit, parity bit and stop bit to the final output of the receiver (parallel_data_out). This all operation happens LOAD state. Further next clock (fsm_clk) i.e. baud clock (baud_clk) receiver moves to LOAD state to the WAIT state.

(D) SIPO Register: Serial input parallels the output shift register used for receiving the data serially. Clock (reg_clk same as baud_clk), active low reset (rst_n), load, shift and input data that is received (serial_data_in) is the input of SIPO register and parallel-out (parallel_data_out) is the output of the SIPO register. When shift signal is high then 1-bit shift at the positive edge of baud clock (baud_clk). When load signal is high 8 bits data is the load to the output of the receiver (parallel_data_out) after removing start bit, parity bit and stop bit.

(E) Parity Checker: Parity checker is used to checking the received data is correct or not. It has one input (data_in) and one output (checker_out). A parity checker is used to check the received data (data_valid) is valid or not. In this paper odd parity is used to check the received data is valid or not.

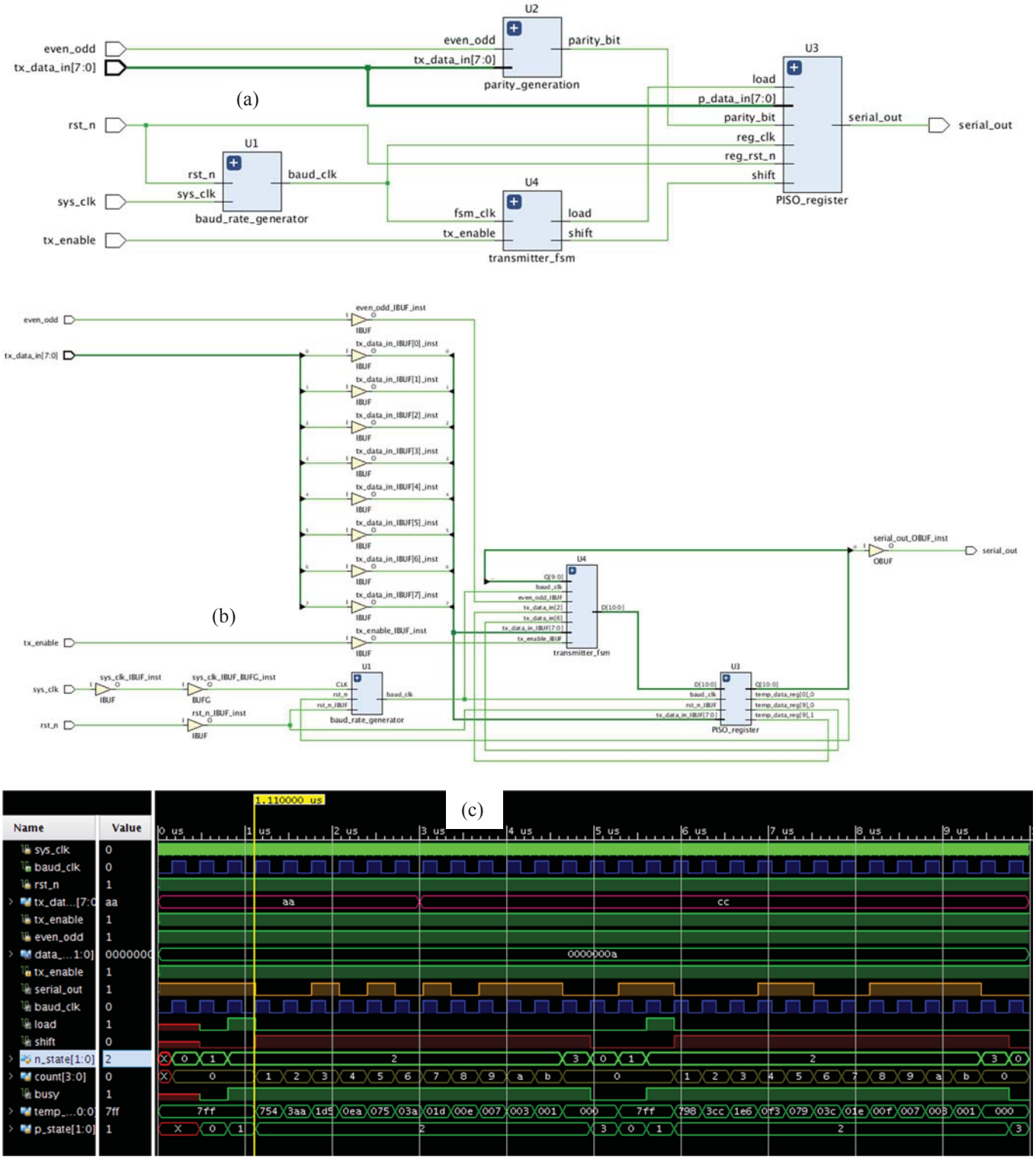

Fig. 7 Proposed UART Transmitter (A) Schametics diagram of UART transmitter (B) Synthesis of UART transmitter (C) Output waveform of UART

IV. Results And Discussions

In this section, simulation, synthesis, and simulation waveform of the transmitter and receiver of UART have been described. Fig. 7 shows the simulation, synthesis and simulation waveform of the UART transmitter, where transmitted data transfer serially. Fig. 8 shows the simulation, synthesis and simulation waveform of the UART receiver, where the receiver receives the data serially and transfer to CPU in a parallel fashion. Logic synthesis is the process of converting a high-level description of the design into an optimized gate-level representation. Fig. 7 (A) shows the schematic diagram of the UART transmitter, which shows the connection of the module used in the transmitter like baud rate generator, parity generator, transmitter FSM and PISO register. Fig. 7 (B) shows the optimized gate-level netlist of the UART transmitter. Netlist describes the area, power and timing constraints of the design. Fig. 7 (C) shows the output

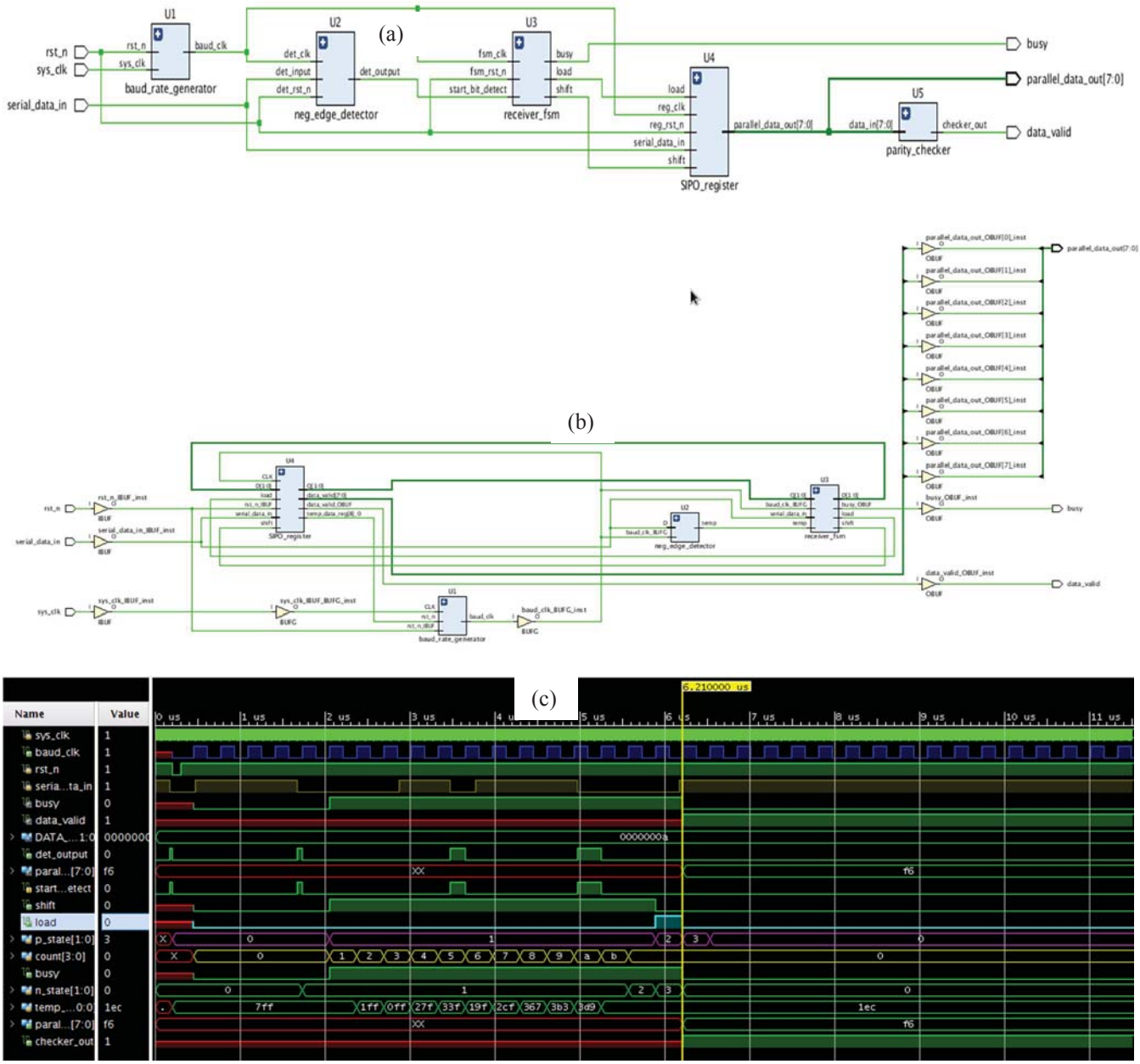

Fig. 8 Proposed UART Receiver (A) Simulation result of UART Receiver (B) Synthesis of UART receiver (C) Output waveform of UART receiver

waveform of the UART transmitter which verified the result of the transmitter. In this paper 8 bits data, 10101010 is transmitted through the transmitter. UART transmitter operates the positive edge of the baud rate (baud_clk) which shows blue color in Fig. 7 (C). The serial output (serial_out) is shown in yellow color. By default the serial output always active high (logic 1). When load signal is high data frame is created in the PISO register which is a combination of 1 start bit (logic 0), 8 data bits (10101010), 1 odd parity bit (logic 1) and 1 stop bit (1). When shift signal is high shifting operation takes place. At 1us time scale first start bit shift which is zero (logic 0) and next positive edge of baud clock (baud_clk) LSB of the data bit shifts and similarly others bits. When transmitter in the SHIFT state counter status and count the bits until all bits are not transmitted. Finally, serial output (serial_out) shows that 11101010100 (11 bits of data frame) transmitted successfully. After successful transmission of 1 Packet of data (10101010), second packet (11001100) is transmitted when the load signal (load) signal is high. If the load signal is high the second time then the data frame for the second packet is created and when the shift signal is high, then shifting of data takes place.

Fig. 8 (A) shows the schematics diagram of the UART receiver, which shows the connection of the module used in the receiver block like baud rate generator, parity checker, negative edge detector, receiver FSM and SIPO register. Fig. 8 (B) shows the optimized gate-level netlist of the UART receiver. Netlist describes the area, power and timing constraints of the design. Fig. 8 (C) shows the output waveform of the UART receiver. UART receiver worked as the same baud rate that's the UART transmitter work. When a negative edge detector detects the start bit, then receiver FSM changes from the IDLE state to the SHIFT state. Shift signal is high until all the serial bits are stored in the temporary

register. When the load signal is high (blue color showed in Fig. 7(C)) store data is the load to the receiver output (parallel_out).

V. Conclusion

In this work, designing, simulation, synthesis and implementation of high-speed Universal Asynchronous Receiver and Transmitter (UART) is successfully and verified using Verilog hardware descriptive language (HDL language). The simulation and synthesis results are implemented Xilinx tool Vivado 2016.2. The high-speed serial data transfer at 4 Mbps rate with a clock frequency of 64 Mbps. UART is used for serial data transfer and the speed of UART is depending upon the transmission media of transmitter and receiver. The baud rate of the UART transmitter and receiver is the same. The verification of UART is verified using the simulation waveform of transmitter and receiver and the synthesis result is verified at the FPGA Zed board.

References

[1] Y. Fang and X. Chen, "Design and Simulation of UART Serial Communication Module Based on VHDL," *2011 3rd International Workshop on Intelligent Systems and Applications*, Wuhan, 2011, pp. 1-4.DOI: 10.1109/ISA.2011.5873448

[2] J. Norhuzaimin and H. H. Maimun, "The design of high-speed UART," *2005 Asia-Pacific Conference on Applied Electromagnetics*, Johor, 2005, pp. 5 pp.-DOI: 10.1109/APACE.2005.1607831

[3] Y. Wang and K. Song, "A new approach to realize UART," *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, Harbin, 2011, pp. 2749-2752. DOI: 10.1109/EMEIT.2011.6023602

[4] G. B. Wakhle, I. Aggarwal and S. Gaba, "Synthesis and Implementation of UART Using VHDL Codes," *2012 International Symposium on Computer, Consumer and Control*, Taichung, 2012, pp. 1-3. DOI: 10.1109/IS3C.2012.10

[5] N. Patel, V. Patel, and V. Patel, "VHDL Implementation of UART with Status Register," *2012 International Conference on Communication Systems and Network Technologies*, Rajkot, 2012, pp. 750-754. DOI: 10.1109/CSNT.2012.164

[6] U. Nanda and S. K. Pattnaik, "Universal Asynchronous Receiver and Transmitter (UART)," *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, 2016, pp. 1-5. DOI: 10.1109/ICACCS.2016.7586376

[7] Poorani, M. and Kurunjimalar, R., 2016, January. Design implementation of UART and SPI in single FGPA. In 2016 *10th International Conference on Intelligent Systems and Control (ISCO)* (pp. 1-5). IEEE.

[8] Liu, B., Wang, Z., Lou, Y., Zhang, M. and Zhong, Q., 2016. Design and implementation of UART based on FPGA. *China Integrated Circuit*, 6, pp.38-41.

[9] Shahu, K., 2019. ASIC Design Implementation of UART using Synopsys EDA tools (Doctoral dissertation, California State University, Northridge).