Implementation of UART using Verilog HDL

Harsh Modi
Dept. of Electronics and Communication Engineering
Nirma University, Ahmedabad, India
21bec068@nirmauni.ac.in

Aryan Makwana
Dept. of Electronics and Communication Engineering
Nirma University, Ahmedabad, India
21bec061@nirmauni.ac.in

***Abstract*—UARTs, or Universal Asynchronous Receiver Transmitters, are frequently employed in data transmission processes because of their advantages in terms of high reliability, extended range, and affordability. In this paper, we present the design of Verilog HDL-based 8-bit UART modules. The character itself includes automated address identification in this design. We have put the planned 8-bit UART module's VLSI architecture into practise and communicate data between it and a host CPU. A transmitter module, a receiver module, and a prescaler module make up the design. We have described the roles of each individual sub-module as well as the simulation behaviour of the architecture.**

***Index Terms*—Introduction, UART Design, Simulation, Applications**

I. Introduction

The Universal Asynchronous Receiver Transmitter, or UART, converts serial data bits received from a serial device to the host processor into parallel data characters and parallel data characters received from the host processor into a serial data stream. The RS-232 is used as a standard for serial data exchange and must be followed. In addition to RS-232, RS-422 and RS-485 standards are now frequently incorporated into UART chips. Compared to RS-232, these protocols provide more dependable communication over far greater distances. A start bit of "0", 5-8 bits of data, and a stop bit of "1" make up a complete data frame format for UART. The stop bit may be 1, 1.5, or 2 bits long [1]. The format of a UART's data frame is shown in Fig. 1 below. Serial data line will be in logic "1" state when in the idle state. A falling edge on the serial data line is produced by a start bit of 0" at the start of the data frame. This indicates that a data character has been detected. Data synchronisation is the goal of the start bit and stop bit in UART.

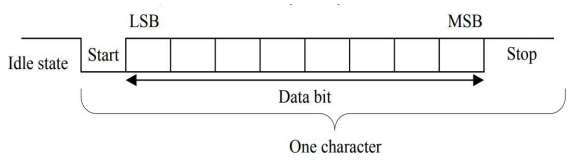

Fig. 1. UART data frame format[2]

II. UART Design

The receiver, transmitter, and baud rate generator, which we referred to as the pre-scaler, are the three fundamental UART submodules that make up the proposed 8-bit UART system. Additionally, this design has internal buffers in the transmitter and receiver. We do this using buffers because our design uses serial clocks and interfaces with parallel clocks on the processor. Smooth data transport between two separate clock domains is made possible by this buffer.

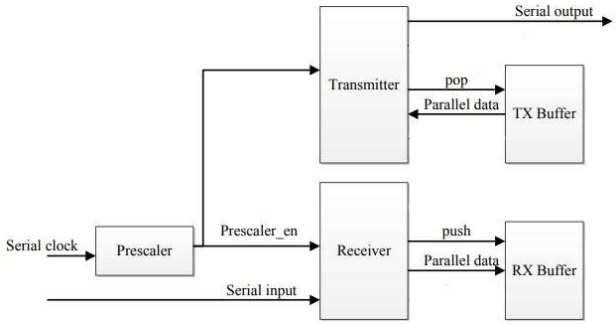

Fig. 2. Overall block diagram of UART[3]

A. Receiver Module:

This is a module for an UART receiver that receives data serially and outputs it in parallel (one byte at a time). The module has an input clock signal (Clk), an active-low reset signal (Rstn), an input enable signal (RxEn), a serial input signal (Rx), a tick signal (Tick), and a parameter input for the number of data bits to receive (NBits). The module has two outputs: a done signal (RxDone) and an output byte (RxData).

The module uses a state machine to receive the data. There are two states: IDLE and READ. When in the IDLE state, the module waits for the start bit (when Rx goes low) and then transitions to the READ state. When in the READ state, the module reads the incoming data and transitions back to the IDLE state when all the data bits have been received.

The counter value used in a UART design can vary depending on the specific implementation and requirements. In some designs, a counter value of 16 may be used for every bit, while in others a different value may be used. In this simulation example, a counter value of 4 has been used for simplicity and ease of understanding. However, in a real-world implementation, a higher counter value would be used to ensure accurate synchronization between the transmitter and receiver,we also have a read-enable signal to keep track of the received data. When the counter reaches a certain value, the

module either sets the start bit to 0 or shifts the incoming data bit into a register. Once all the data bits have been received, the module sets the RxDone signal to high and outputs the received byte on the RxData signal.

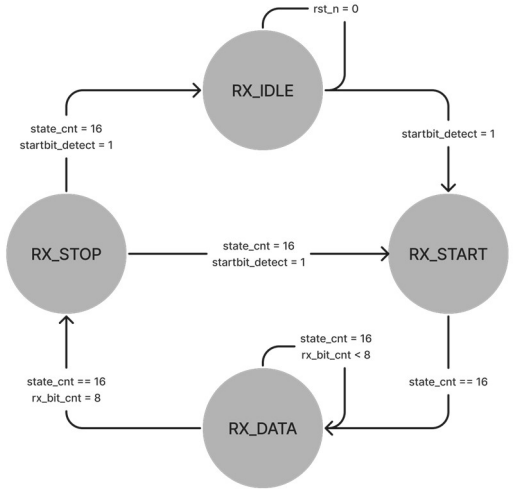

Fig. 3. Reciever Module State Machine[1]

The module includes several registers and variables used for the state machine, such as ”State”, ”Next”, ”TxDone”, ”Tx”, ”writeenable”, ”startbit”, ”stopbit” ,”Bit”, ”counter”, ”indata”, ”Redge” and ”Dedge.”

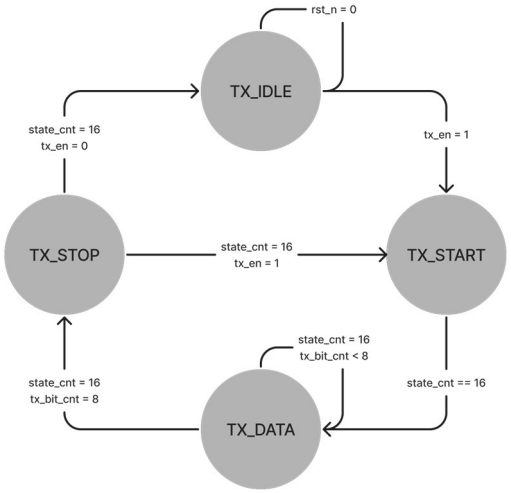

Fig. 4. Transmitter Module State Machine[1]

B. Transmitter Module:

UART transmitter (Tx), which is a common protocol used for serial communication between microcontrollers, computers, and other devices. The module is designed as a state machine that sends data to a receiver over a single data line (Tx).

The module has several input signals: Clk, Rstn, TxEn, Tick, NBits, and TxData. Clk is a clock signal that drives the state machine, Rstn is a reset signal, TxEn is a signal that enables data transmission, Tick is a signal used to count clock cycles, NBits is a 4-bit value that specifies the number of bits to be transmitted, and TxData is an 8-bit value that contains the data to be transmitted. The module also has two output signals: Tx and TxDone.

The state machine has two states: IDLE and WRITE. The module starts in the IDLE state, waiting for the TxEn signal to be enabled. When the TxEn signal rises, the state machine transitions to the WRITE state and starts sending data to the receiver. The data is sent bit by bit using a loop that iterates for the number of bits specified by NBits. The transmission begins with the start bit, followed by the data bits, and then the stop bit.

The TxDone signal is used to indicate when the transmission process is complete. Once the transmission is finished, the module returns to the IDLE state and waits for TxEn to be enabled again.

To control the timing of the data transmission, the module uses the Tick signal to count clock cycles. The counter variable counts the number of clock cycles since the start of the transmission. When the counter reaches a certain value, the transmitter changes the value of the Tx output signal to transmit the next bit of data.

C. Baud Rate Generator Module:

The ”uart baud” module in Verilog is used to generate a periodic ”Tick” signal that is used for timing in a UART transmitter module. This Tick signal is generated based on the desired baud rate and the clock frequency of the FPGA.

The module has four input and output signals: Clk, Rstn, BaudRate, and Tick. Clk is the clock signal for the module, Rstn is the active-low reset signal for the module, BaudRate is a 16-bit input signal that specifies the desired baud rate for the UART transmitter, and Tick is the output signal that generates a pulse every ”BaudRate” clock cycles.

Inside the module, there is a register called ”baudRateReg” that counts the number of clock cycles that have elapsed since the last Tick pulse. The module uses this register to generate a pulse on the Tick output signal every ”BaudRate” clock cycles.

The baudRateReg register is initialized to 1 when the reset signal (Rstn) is low, which sets the initial value for the Tick signal to be high. When the reset signal is high, the module waits for the next rising edge of the Clk signal to increment the baudRateReg counter. The counter is incremented on each rising edge of the Clk signal unless the Tick signal is high, which means that the module is currently generating a pulse on the Tick output signal.

When the baudRateReg counter reaches the value of ”BaudRate,” the Tick signal is set to high, and the counter is reset to 1 to start counting again. The Tick signal remains high for one clock cycle before returning to low.

The value of ”BaudRate” is calculated based on the desired UART baud rate and the clock frequency of the FPGA. In this case, the module assumes a clock frequency of 50MHz and a desired UART baud rate of 9600. The module calculates that

it needs 16 clock cycles per bit of data, so it sets ”BaudRate” to 325.

Overall, this module provides a simple way to generate a periodic pulse signal at a specific frequency, which is necessary for timing in the UART transmitter module. The Tick signal is used to control the timing of the data transmission and ensure that the data is transmitted at the correct baud rate.

$$\text{Divide ratio} = \frac{\text{Serial clock frequency}}{16 * \text{Baud rate}}$$

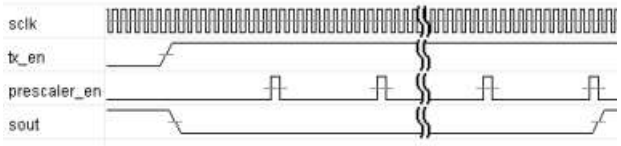

Fig. 5. Overall block diagram of UART[5]

D. Main Module:

The ”uart main” serves as the top-level module that connectsthe UART transmitter and receiver modules with the baud rate generator module.It includes modules for receiving data (uartrx), transmitting data (uarttx), and generating the baud rate clock (uartbaud).

The input signals to this module are Clk (clock) and Rstn(reset), and the output signals are Tx (transmit), RxData (received data), Rx (receive), RxDone (reception completed),TxDone (transmission completed), and tick (baud rate clock).The Rx signal is a reg (register) type and is assigned the value of the Tx signal at every positive edge of the tick signal. This means that data is received synchronously with the baud rate clock.

The TxData signal is an input that holds the data to be transmitted, and TxEn is a wire that enables the transmission of data. Similarly, RxEn is a wire that enables the reception of data. The NBits signal is a wire that specifies the numberof bits to be transmitted or received (in this case, 8 bits). The BaudRate signal is a wire that specifies the baud rate of the communication (in this case, 9600 bits per second).

The uartrx module takes the Clk, Rstn, RxEn, RxData, RxDone, Rx, tick, and NBits signals as inputs and outputs RxData, RxDone, and Rx. The uarttx module takes the Clk, Rstn, TxEn, TxData, TxDone, Tx, tick, and NBits signals as inputs and outputs Tx. The uartbaud module takes the Clk, Rstn, tick, and BaudRate signals as inputs and outputs tick.

The module is implemented using always block that is sensitive to the positive edge of the tick signal. Within the always block, the Rx signal is assigned the value of the Tx signal.

Here we used baud rate generator to give use the baud rate at which we will transmit the data here it is known as ”Tick”, then this tick is passed to both the transmitter and receiver block and based on this generated output of baud rate generator transmitter will send the data (txen=1) and will receiver block is ready to receive the data (rxen=1) then it will receive thetransmitted data from transmitter and this data will be storedin the output register of receiver block.

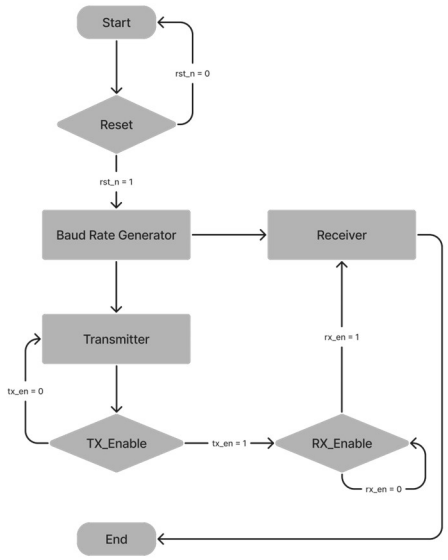

Fig. 6. Flow Chart of main module[7]

III. Simulation

A. Baud Rate Generator Module

Fig. 7 shows the simulation waveform of the Baud Rate Generator module and Fig. 8 shows the RTL view of baud rate generator module. In this simulation, the baud rate is generated from the main clock using a counter and registers. A divisor of 2 is used, which means that the counter will count to give the desired waveform. The generated waveform producesa pulse known as a tick, which is generated after every second cycle.

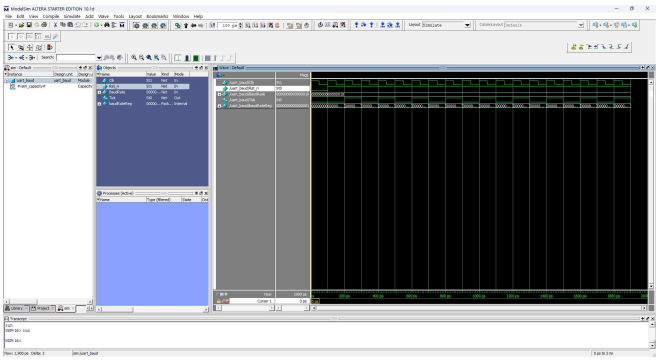

Fig. 7. Simulation waveform of baud rate generator module

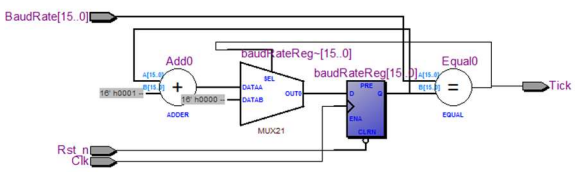

Fig. 8. RTL view of baud rate generator module

B. Transmitter Module

Fig. 9 shows the simulation waveform of the transmitter module and Fig. 10 shows the RTL view of transmitter module. In this simulation, the data to be transmitted is stored in a register. When the txen input is set to 1, the transmission process begins. The first step is to send a start bit, which is always set to 0. Then, the data from the register is transmitted.bit by bit, starting with the least significant bit (data[0]). Once all the data bits are sent, a stop bit is sent, which is alwaysset to 1. Finally, the txdone signal is set to high to indicatethat the transmission is complete.

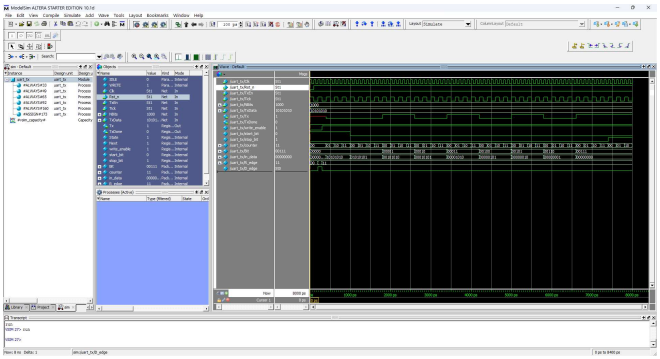

Fig. 9. Simulation waveform of transmitter module

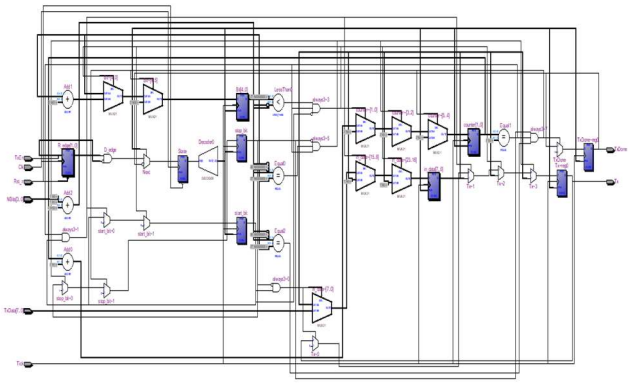

Fig. 10. RTL view of transmitter module

C. Receiver Module

Fig. 11 shows the simulation waveform of the Receiver module and Fig. 12 shows the RTL view of receiver module. In this simulation, we have a receiver module that receives data from the transmission module when the rxen signal is high. The received data is stored in the rx bit, and based on the tick (baud rate), the data is stored in the buffer register. As new data is received, the previously stored data in the buffer is shifted. After receiving 8 bits of data, the data from the buffer is sent to the output register, and the rxdone signal is set high to indicate that the reception is complete.

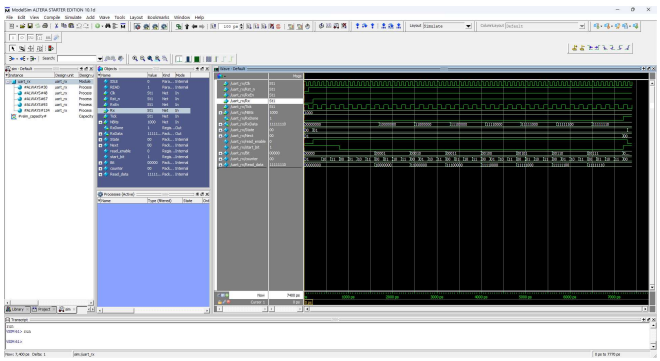

Fig. 11. Simulation waveform of receiver module

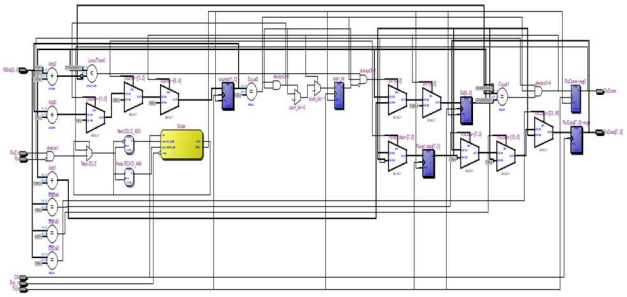

Fig. 12. RTL view of receiver module

D. Main Module

Fig. 13 shows the simulation waveform of the main module and Fig. 14 shows the RTL view of main module of UART. In this simulation, the main module acts as a top-level module and connects the previous modules, including the baud rate generator, transmitter, and receiver modules. The baud rate generated from the generator is passed to both the transmission and receiver modules. During operation, the main modulewaits for the reset signal to go high. Once the reset signal is high, data is transmitted or received depending on the valuesof txen and rxen signals. The transmission module receives data to be transmitted from a register and sends a start bit (0), followed by the data bits (data[0] is transmitted first) and a stop bit (1). The txdone signal is high once the transmission is complete. Similarly, the receiver module receives the data transmitted by the transmission module through the rx bit. Thereceived data is stored in the buffer register according to the tick (baud rate), and the data is shifted when new data is storedin the buffer. Once 8 bits of data have been received, the data from the buffer is sent to the output register, and the rxdone signal is set high. Overall, the transmission and reception of data is synchronized with the tick signal generated by the baudrate generator.

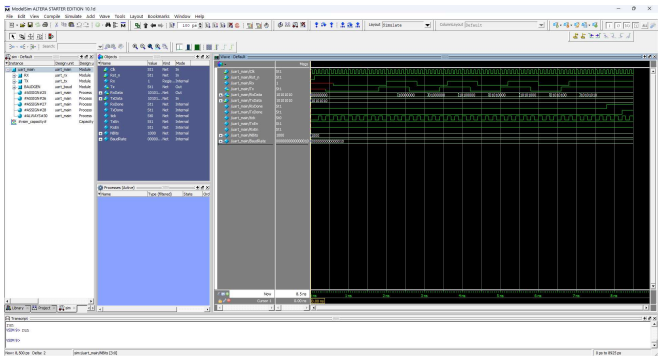

Fig. 13. Simulation waveform of main module

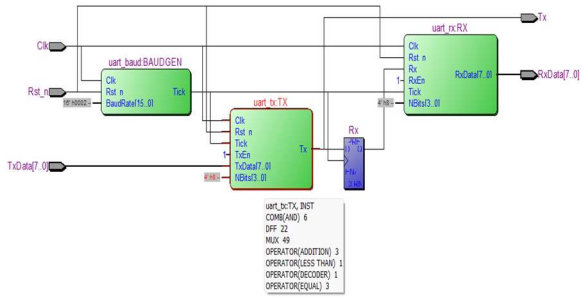

Fig. 14. RTL view of main module

IV. Advantages

1) Simplicity: UART is a relatively simple protocol thatis easy to implement and use. It requires only a few connections between devices, typically just two wiresfor data transmission (TX and RX).
2) Versatility: UART can be used to communicate betweena wide range of devices, including microcontrollers, sensors, displays, and other peripherals. This makes it a very versatile protocol for many different applications.
3) Low cost: Because UART is a relatively simple protocol,it can be implemented using low-cost hardware, making it an affordable option for many applications.
4) Speed: UART can support data transmission rates up to several megabits per second, depending on the specific implementation and hardware used.
5) Asynchronous Communication: UART operates asynchronously, meaning that the timing of data transmission is not dependent on a shared clock signal between the transmitter and receiver. This simplifies the interface and allows communication between devices with different clock rates.
6) Flexibility in Baud Rate: UART supports a wide range of baud rates, allowing flexibility in communication speed according to the requirements of the system.
7) Full Duplex Communication: UART supports full-duplex communication, enabling simultaneous transmission and reception of data. This is particularly useful in applications where real-time bidirectional data exchange is needed.
8) Wide Range of Applications: UART is used in a diverse range of applications, including data logging, sensor interfacing, debugging, and communication between microcontrollers and peripheral devices.

V. Disadvantages

1) Limited distance: UART is a point-to-point communication protocol, which means that it is designed to communicate between only two devices over a short distance. If longer distances need to be covered, additional hardware, such as modems or converters, may be needed.
2) Slow communication: UART is a relatively slow com- munication protocol compared to other communication protocols such as Ethernet or USB. This is because data transmission occurs in a sequential manner, one bit at a time, which limits the overall speed of the communication.
3) Limited functionality: UART is a basic communication protocol that supports only simple data transmission. It does not support more complex communication features such as error correction, flow control, or multi-device communication.
4) Susceptibility to noise: Because UART communication occurs over a physical wire, it is susceptible to interference from external sources such as electromagnetic interference (EMI) or radio frequency interference (RFI), which can cause errors in data transmission.
5) Asynchronous communication: Asynchronous communication used in UART protocol can lead to timing issues in the communication if both the transmitting andreceiving devices are not configured with the same baudrate.

VI. Applications

1) Serial Communication: UART is widely used for serial communication between devices. Many electronic devices, such as microcontrollers, sensors, and GPS receivers, use UART to transmit and receive data ina serial format. For example, a microcontroller can communicate with a computer using UART to send and receive data over a serial connection.
2) Industrial Control Systems: UART is often used in industrial control systems, such as PLCs (Programmable Logic Controllers) and SCADA (Supervisory Control and Data Acquisition) systems. These systems use UART to communicate with various sensors and de- vices, such as temperature sensors, pressure sensors, andvalves.

3) Automotive Industry: UART is used in automotive applications, such as engine management systems, airbag systems, and entertainment systems. UART is used to communicate between various components, such as the engine control module and the dashboard display.
4) Internet of Things (IoT): UART is widely used in IoT devices, such as smart home devices and wearable devices. These devices often use UART to communicate with other devices, such as smartphones and routers.
5) Medical Devices: UART is used in medical devices, such as patient monitors, ultrasound machines, and ECG machines. UART is used to communicate between various components, such as the display and the sensor.
6) Robotics: UART is often used in robotics applications, such as autonomous vehicles and drones. UART is used to communicate between various components, such as the motor controller and the sensors.
7) Aerospace: UART is used in aerospace applications such as flight control systems, communication systems, and navigation systems. UART is used to communicate between various components, such as avionics systems and GPS receivers UART has a wide range of applications in various industries and devices, where it is used to enable com- munication between different components and devices in a reliable and efficient manner.

VII. Conclusion

We have covered the basic functioning of UART and how it is implemented in a Verilog simulation. UART is a communication protocol used for serial communication between two devices. It consists of a transmitter and a receiver, both of which use a common baud rate to communicate.

In the Verilog simulation, we have seen how the baud rate is generated using a counter and register, and how the transmission and receiver modules work to send and receive data, respectively. The transmission module sends data bit by bit with a start and stop bit, while the receiver module receives data bit by bit and stores it in a buffer register. Once the bufferregister is full, the data is sent to the output register and the done signal for receiver is set high.

Overall, UART is a widely used communication protocol due to its simplicity and reliability. In the Verilog simulation, we have demonstrated how UART can be implemented in a digital design using the Verilog hardware description language.

VIII. Acknowledgement

We would like to extend our heartfelt appreciation to Prof. Dhaval Shah for his valuable guidance and support in completing this research project. Their expertise and guidance have been instrumental in our success, and we are deeply grateful for their efforts.

Furthermore, we would like to acknowledge all the teachers who have helped us throughout this project. Their contributions and insights have been essential in shaping our understanding and knowledge in the field of Verilog HDL and UART implementation.

We would also like to thank our colleagues and classmates for their assistance, encouragement, and motivation during the course of this project. Their input and feedback were invaluable in improving the quality of our work.

Lastly, we express our gratitude to our families for their unwavering support and understanding throughout this project. Their love and encouragement were a source of motivation and inspiration for us to persevere and complete this research work.

REFRENCES

[1] UART: A Hardware Communication Protocol - Analog Devices
[2] Basics of UART Communication - Circuit Basics
[3] Universal Asynchronous Receiver-Transmitter - Wikipedia
[4] What is UART? - Tutorials Point
[5] Umakanta Nanda, Sushant Kumar Pattnaik ”Universal AsynchronousReceiver and Transmitter (UART)” ICACCS 2016
[6] S. S. Limaye, Hema Kale”UART Transmitter Experiment in FPGA”
IJIRT 2019
[7] Nennie Farina Mahat, Member ”Design of a 9-bit UART Module Basedon Verilog HDL” IEEE-ICSE2012