International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Design and Implementation of UART Based on Verilog HDL with adaptive baud rate detection

Ms. Pavithra D
Embedded V& V,
L&T Technology Services,
Chennai, India.

Ms. Hemalatha P
Power & Utilities,
L&T Technology Services,
Chennai, India.

Abstract - Universal Asynchronous Receiver Transmitter (UART) is a standard communication protocol in digital circuits and embedded systems, offering efficient serial data transfer. Traditional UARTs communicate at a fixed baud rate, requiring preconfiguration of the transmitter and receiver to offer smooth communication. In dynamic systems where devices with varying baud rates communicate, synchronizing is challenging.

This paper presents the implementation and design of an advanced UART system using Verilog HDL, which has an adaptive baud rate detection system. Unlike traditional UART designs based on fixed baud rates, the proposed system detects and adjusts its baud rate dynamically to synchronize with the transmitting device. This is achieved through data stream timing behavior analysis and synchronizing accordingly. The adaptive system offers improved interoperability, offering smooth communication between devices with varying baud rates without manual reconfiguration.

Synthesis and implementation of the design on an FPGA board demonstrated that simulation results validated its effectiveness in offering stable and error-free communication across varying baud rates. The adaptive baud rate detection system reduces latency and offers improved data integrity by reducing synchronization errors, thus offering a robust solution to real-time applications. The flexible UART architecture is highly optimum for embedded systems, IoT applications, and scenarios where heterogeneous devices with varying transmission speeds communicate reliably.

Keywords - Universal Asynchronous Receiver Transmitter (UART), adaptive baud rate detection, serial communication, Verilog HDL, FPGA implementation, dynamic baud rate adjustment, real- time synchronization, embedded systems, error detection and correction, low-latency communication, IoT applications, interoperability in serial communication, high-speed digital communication, flexible UART architecture, reliable, data transmission.

INTRODUCTION

Challenges in Traditional UART Implementations
A standard UART system works based on a fixed baud rate that must be configured in advance for both the transmitter and receiver to successfully communicate. This static approach presents several critical challenges, especially in large-scale and dynamic environments such as IoT, industrial automation, and embedded systems:

Manual Configuration Overhead - Setting the correct baud rate manually for each device can be cumbersome, particularly when integrating multiple devices from different manufacturers.
Low flexibility - Any baud rate change involves reconfiguration; traditional UART implementations are rigid and thus not useful in dynamic applications.
Interoperability issues - A mismatch of baud rates on the devices involved could prevent efficient communication, thereby leading to failure of data transfer and incompatibility issues
Difficulty in High Deployments - In industrial or IoT ecosystems involving thousands of devices connected, the baud rate is quite impossible to synchronize manually as well as tends to be errors.

Data integrity and reliability concerns - The possibility of framing errors, data corruption, and high latency will compromise system performance and reliability if there is an improper baud rate setting.

Considering all these limitations, there is an increasing demand for a more adaptive, intelligent, and flexible UART communication system which can change the baud rate at runtime based on the incoming streams of data.

The idea of this project is to design and implement an advanced UART system with adaptive baud rate detection in order to solve the challenges imposed by traditional UART implementations. Conventional UART designs rely on predefined baud rates; however, the proposed system will automatically detect and synchronize its baud rate with that of the transmitting device. Therefore, this proposed solution will remove the necessity of manual configuration and enhance interoperability between devices running at different baud rates for easy communication.

The system is implemented using Verilog HDL, and it operates on an FPGA platform, hence ensuring high-speed processing, efficient resource utilization, and reliable real-time communication. The adaptive baud rate detection mechanism works by examining the timing characteristics of incoming data packets, identifies the baud rate of the transmitting device, and adjusts the baud rate of the receiver, accordingly, thus allowing UART module to reach real-time synchronization without human interference.

Key Benefits of the Adaptive UART System

Real-time Synchronization - The UART dynamically

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

detects and adapts to varying baud rates, ensuring smooth communication across multiple devices.

Improved Flexibility - Eliminates the constraints of fixed baud rate configurations, making the system highly adaptable to diverse applications.

Enhanced Interoperability - Enables seamless communication between different devices without requiring baud rate pre-configuration.

Less Latency - With no requirement of manual intervention, the system cuts down communication latency and thereby efficiency as a whole.

Increased Data Integrity - No mismatch in baud rates that often cause packet loss and error increases

data integrity for the entire communication process. **Suitable for Big Systems** - Used in big IoT networks, industrial automation, and embedded systems in which there are many devices functioning at various speeds.

Implementation of UART

The Universal Asynchronous Receiver/Transmitter (UART) is one of the most used serial communication protocols. UART offers a reliable transfer of data between digital systems with asynchronous operation without requiring a common clock signal to transfer and receive data independently on predetermined baud rates. It has thus become very popular in the fields of embedded systems, industrial automation, and IoT applications. The need for smooth communication among microcontrollers, computers, and peripheral devices calls for the application of UART in these areas. The protocol converts parallel data from a processing unit into a serial bit stream for transmission and reconstructs received serial data back into parallel format for processing.

One of the main benefits of UART communication is that it supports full-duplex transmission using separate TX (transmit) and RX (receive) lines. Data can be sent and received at the same time, which will make the data transfer efficient and reliable. Since UART is asynchronous, baud rate synchronization between the transmitter and receiver is critical in order to prevent data corruption.

Working of UART

UART communication follows a structured process that includes both transmission and reception mechanisms. The transmission process begins when the UART module encapsulates parallel data into a structured frame. A start bit (logic low) is appended to indicate the beginning of transmission, followed by the actual data payload, which typically consists of 5 to 8 bits. If error detection is required, an optional parity bit is included. One or more stop bits conclude the frame, marking its end, where synchronization takes place before sending a new data frame. The frame is sent as a whole across the communication channel in serial order.

At the receiver end, the UART continuously monitors the line for transition from idle, being logic high to active, or logic low to mark the onset of a new data frame. After detecting the start bit, the receiver samples the incoming bits at regular intervals defined by the baud rate configuration. In case a parity bit is involved, error-checking mechanisms are applied to ensure that the data received is intact. The received serial data is then converted back into parallel format and processed appropriately. Since UART communication does not use a shared clock, baud rate synchronization is the key to keeping transmission errors as low as possible.

UART Frame Format

A UART data frame is a formatted sequence of bits that allows reliable transmission and reception. The frame starts with a start bit-a single logic-low signal that aligns the receiver's clock with the transmitter's data stream. This is followed by the data bits, which constitute the primary information payload and can range between 5 to 8 bits in length. Data field lengths are also configurable, based on system requirements and application needs.

To add another layer of data reliability, an optional parity bit can be added. The parity bit is used to detect errors based on the even or odd rule of the number of '1s' that must be found in the frame. This allows for the identification of single- bit errors, making it a great mechanism for increasing security in the transmission of data. The end of the frame is marked with one or more stop bits that indicate the termination of data transfer and ensure the receiver can properly distinguish between two consecutive frames. The number of stop bits will be 1, 1.5, or 2, according to the system's configuration. This type of frame format creates a reliable and error-checked form of communication for UART.

UART Simplified Frame Format for Enhanced Communication Efficiency

In this research, an efficient UART frame format is presented to enhance the efficiency of communication and reduce processing overhead. This modified format eliminates the parity bit but retains a structured transmission process. The new frame format is composed of a start bit (1

bit), followed by a fixed-length data field of 8 bits to ensure consistency across transmissions. Unlike the standard frame format, the parity bit is omitted, thus reducing transmission complexity and processing time. The frame concludes with a stop bit (1 bit), which marks the end of the transmission cycle.

The modified format aligns with the RS-232 communication standard, which is widely used in serial data transmission. The removal of the parity bit minimizes computational overhead while preserving data integrity, making it particularly suitable for applications requiring real-time communication and optimized performance. This full-duplex capability of UART ensures that the data transmission and reception happen at the same time, thus further improving the overall communication efficiency. The adoption of this modified frame format enhances the system responsiveness

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

and makes it suitable for high- speed embedded applications and industrial communication systems..

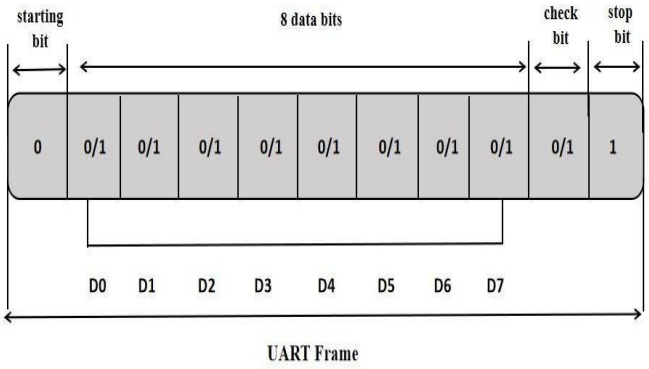


Fig .1 Data frame format of UART

Division of modules

UART consists of two basic sub-modules, i.e., serial port sending module and serial port receiving module that handles different aspects of data transmission and reception, respectively. In other words, these modules work in tandem with each other to facilitate efficient serial communication by converting parallel data to serial format for transmission and reconstructing received serial data into parallel format for processing, respectively.

The function of the serial port transmitting module is to break up parallel information from the source end into a regular serial data stream. It starts by taking in parallel information from the system's data bus. It first adds a start bit indicating the start of a new frame, then it contains actual data bits, which could be any number of bits (5 to 8). If enabled, the parity bit is appended to the frame for error detection, and one or more stop bits are added to signal the end of the transmission. The complete frame is then transmitted bit by bit through the TX (transmit) line at a predetermined baud rate, so that the receiving module can interpret the data based on synchronized timing parameters.

The serial port receiving module detects and interprets the incoming serial data stream. It continuously monitors the RX (receive) line, waiting for the start bit, which signifies the arrival of a new data frame. Upon detecting the start bit, the receiver synchronizes its clock to sample the subsequent data bits at the appropriate intervals. Then the module extracts data bits from the received frame, checks for errors if a parity bit is present, and checks for the stop bits to verify the end of transmission. The extracted serial data is then converted back into parallel format and forwarded to the system's data bus for further processing.

Both the transmission module and the receiving module depend upon baud rate synchronisation for transmitting data accurately. If the sender and receiver baud rates are different, then some data corruption could occur. Together, these sub-modules ensure the reliability of asynchronous communication, so UART is quite a robust and widely used protocol in embedded systems and industrial applications. Figure 2 and Figure 3 show the structural workflow of these modules as it relates to the data transmission and reception process.

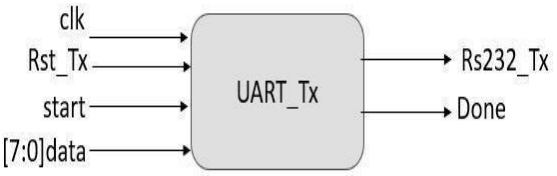


Fig. 2 UART sending module

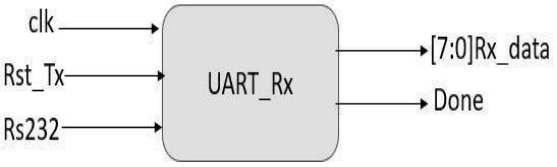


Fig. 3 UART receiving module

UART Transmission and Reception Modules

The Universal Asynchronous Receiver/Transmitter (UART) communication system has two main functional modules: the UART Sending Module and the UART Receiving Module. These modules work in synchronization to ensure efficient and reliable data transmission and reception. The sending module is responsible for converting parallel data into a serial format and sending it. In contrast, the receiving module decodes the incoming serial data and converts it into a parallel format for further processing. The operation of these modules is controlled by a master clock signal, using a start and reset signal.

UART Sending Module

The UART Sending Module is an important module in serial communication, which takes parallel data from the system and converts it into a structured serial data stream. This is done by encapsulating the data within a frame that follows the UART transmission protocols. The main signals used in this module are as follows: **Clk:** The master clock signal that drives the transmission process.
stun: The active-low reset signal that initializes the module.
Rs232_tx: The transmission line over which the serialized data is sent.
data[7:0]: An 8-bit parallel data input that is serialized for transmission.
valid: A signal indicating that the data is valid and ready for transmission.
This UART sending module will see that every bite is sent out in the order of its receipt and has a start bit, which signals when to start sending data bits, followed by stop bits as

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

it reaches the end. Therefore, the module for receiving can decipher the data received correctly.

UART Receiving Module

The UART Receiving Module captures and decodes incoming serial data. It constantly checks the Rs232_rx line for the start bit, which indicates the presence of a new data frame. The key signals in the receiving module are as follows:

Calk: Master clock signal controlling the sampling process.
stun: Reset signal to initialize or reset the module. **Rs232_rx:** Serial reception line where data is received.
data[7:0]: The 8-bit parallel data output, rebuilt from the serial data received.
valid: The signal indicating that the received data is valid and ready for processing.
The receiving module captures data from the serial input through sampling of every bit at certain intervals set using the baud rate. The end of the frame is also searched for a stop bit that helps confirm whether the received data was correct and valid before sending it to the processing stage.

Process of Transmission and Control of States

The transmission module acts in a step-by-step order to ensure effective data transmission. The process is initiated by the Start Signal that marks the initiation of a cycle of data transfer. When the Start Signal reaches high, it initiates transmitting by sending out the start bit, followed by the 8 data bits transmitted in Least Significant Bit (LSB) First order. Transmitted last was the most significant bit (MSB) so that the MSB and the LSB are of the same polarity.

Data collision is prevented using Data Caching, where redate temporarily holds the data for transmission so that new data will not interfere with an ongoing transmission. This makes data flow smooth and free of corruption.

The module works using State Logic in which:

State = 0: The module is idle, waiting for new data.
State = 1: The module is transmitting data.

The Baud Rate Setting is controlled by abducent, which determines the number of clock cycles required for one data bit to be transmitted. For example, if abducent is set to 5208, the module transmits one bit for every 5208 clock cycles, ensuring synchronization between the transmitter and receiver.

In Bit Transmission, a bitflag is set to logic 1 every time a bit is transmitted. The bit_cnt counter counts the number of bits that have been transmitted, counting up from 0 to 10 for every complete frame that is made up of the start bit, 8 data bits, and the stop bit.

Once the entire frame is transmitted, the Done Signal is triggered when bit_cnt reaches 10, indicating that the data transfer is complete. At this point, the state signal is set back to idle (logic 0), allowing the module to prepare for the next transmission cycle.

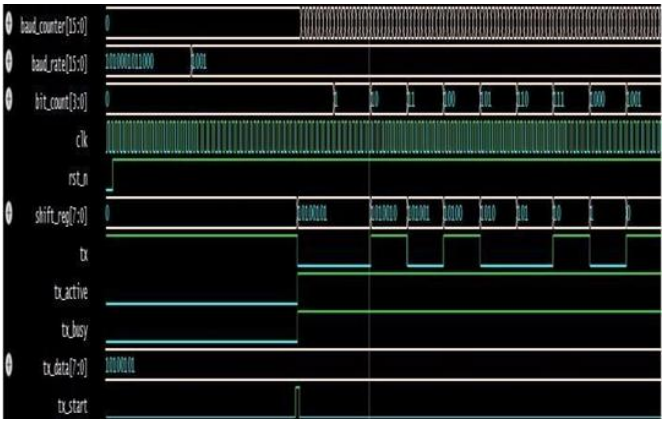


Fig. 4 Timing diagram of sending module Clk :

Master clock signal
rst_n : Reset signal start : Start signal
data[7:0] : block 8-bit data to be transmitted Rs232_tx : Serial data output
done : Transmission completion signal state : Idle/working state indicator baud_cnt : Baud rate counter
bit_flag : Bit transmission flag bit_cnt : Bit counter

Metastability Removal

The UART receiver experiences metastability whenever the incoming serial data signal crosses over near the clock edge, violating the setup and hold time requirements of the sampling flip-flop of the receiver. This gives rise to unstable or non-deterministic output that could lead to incorrect data interpretation. Since UART communication is asynchronous, there is no shared clock between the transmitter and receiver module, and it is subject to metastability when sampled in a wrong phase of the inputting data.

To overcome this problem, the UART receiver includes a two-stage synchronization method. The first-stage flip-flop captures the serial data signal into the receiver at the rising edge of the system clock. If the transition of the data falls within this margin of time from the rising edge, the flip-flop can go into a metastable state, and its output will be uncertain. However, this unstable state self-resolves in a few nanoseconds. When metastability occurs, the second-stage flip-flop captures the stabilized output of the first stage. In this case, the definite logic level is ensured to propagate to the rest of the system. It effectively prevents metastability-induced errors and prevents invalid data from being processed further.

The effectiveness of metastability removal depends on the clock frequency and the timing characteristics of the flip-flops. A higher clock frequency allows a shorter resolution

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

time between the two sampling stages. This increases the probability that the metastable state settles before the second-stage flip-flop captures the signal. In addition, oversampling techniques in UART receivers further minimize the effects of metastability by creating more sampling points within a single bit duration. This means that the real data transfer is sensed with greater accuracy and, hence, minimizes the timing uncertainty effects. A metastability removal mechanism in the UART receiver can ensure the dependable processing of incoming data without being corrupted. This would ensure that asynchronous data transfer remains stable, in the presence of varying signal conditions, during operation at higher baud rates.

Reliable UART Reception with Noise Filtering and Timing Synchronization

UART-based serial communication reception starts with the transition of the r signal from logic "1" to "0," which indicates a start bit and the beginning of a new data frame. However, noise present on the communication line can cause unwanted transitions that result in wrong data reception. To counteract this, there is a mechanism for falling edge detection, thereby making the system differentiate between valid transitions and spurious glitches resulting from noise.

The reception process is well synchronized with key control signals that handle data accurately without losing synchronization.
The key signals that are in reception are given as follows:
baud_cnt: This defines the baud rate, which then controls the reception of data bits.
bit_cnt and bit_flag: Are used as a pair to keep track of and check for reception of individual bits. bit_cnt counts the number of received bits, and bit_flag checks the readiness of a bit for processing.
rx_data: Temporary buffer that holds the correctly sampled received data until processed.
done signal: The done signal indicates that the data frame is received completely. After the sampling counter reaches the last count, meaning all the bits are received, the done signal is asserted.
After successful reception, the FSM returns from active reception to the idle state, so the receiver is ready for the next received data frame. The use of the falling edge detection, baud rate synchronization, and metastability handling
Experimental Results

The following experimental results verify the design and implementation of the UART system with adaptive baud rate detection. The waveform captures the critical signals involved in baud rate estimation, data reception, and state transitions within the system.

1. Adaptive Baud Rate Detection

The detected baud rate (baud_rate[15:0]) stabilizes at 1458 (decimal), which means that the system correctly determines the incoming baud rate dynamically.

improves UART reception by noise filtering and providing precise data sampling, thus providing robust communication in the system; hence, it is reliable for use in real-time embedded applications and IoT-based implementations.

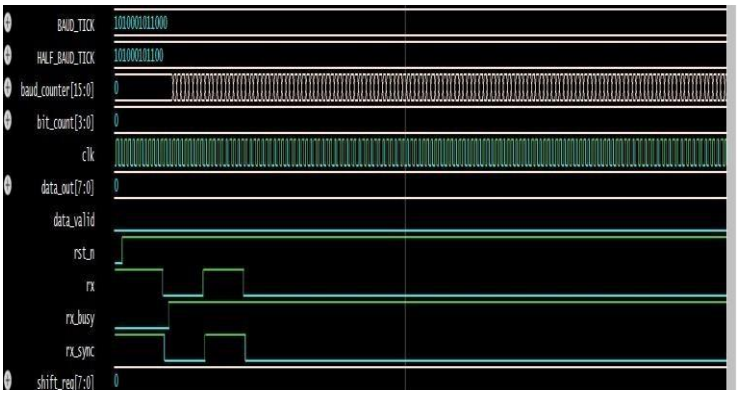


Fig. 5 Timing diagram of receiving module

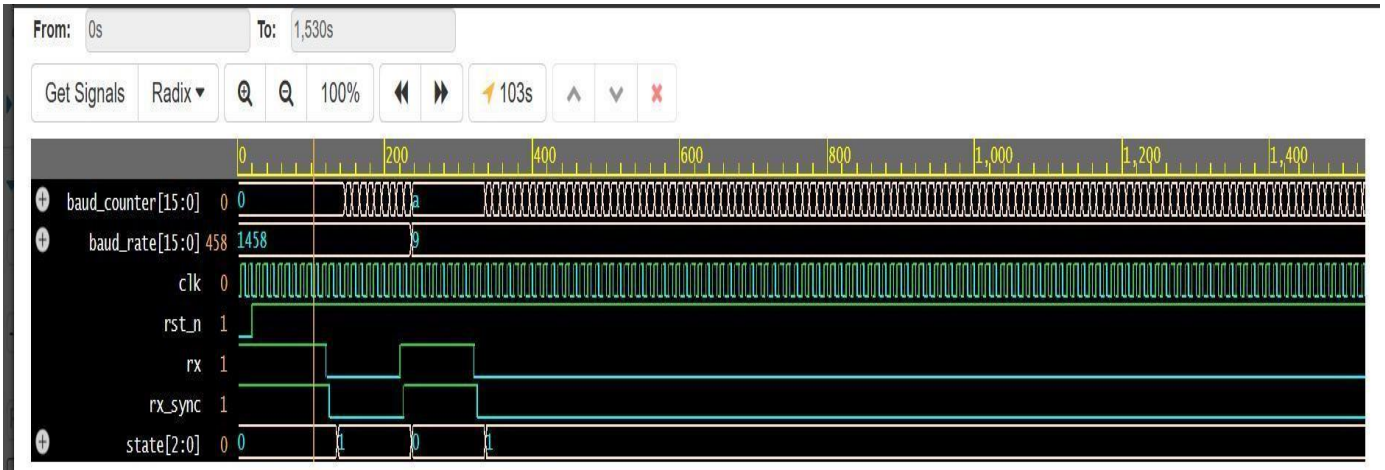
Testbench for UART System with Adaptive Baud Rate Detection

The Verilog testbench for the UART system with adaptive baud rate detection verifies the receiver, transmitter, and baud rate estimation modules. It generates a clock signal, applies reset, and injects serial data at varying baud rates to test dynamic adaptation. The baud rate detection mechanism

estimates the correct baud rate by monitoring data transitions. Key signals like baud_rate, baud_counter, tx_data, and rx_data are analyzed to ensure proper synchronization and error-free communication.
The testbench generates waveform data using VCD files for signal visualization to facilitate debugging. Controlled scenarios, including baud rate variations and noise effects, assess system robustness. The receiver is tested for accurate data decoding, while the transmitter ensures proper serialization. Results confirm the system's reliability across varying transmission speeds, validating the adaptive baud rate detection mechanism.

The baud counter (baud_counter[15:0]) is incremented appropriately, helping to achieve an accurate baud rate estimation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026



These values validate that the UART receiver can synchronize to any baud rate without needing a prior setup.

2. Serial Data Reception and Synchronization

The rx input goes from high to low, signifying that a start bit has been received.
The rx_sync is asserted, ensuring that the incoming data is sampled correctly.
The state machine is self-aligning with the falling edge of the start bit so that accurate bit sampling can occur at the baud rate determined.

3. FSM Transition

The state of the state machine, State[2:0], changes from 0 to 1 to 0 again and forms different operating states of the UART module, which are as follows: State 0: Idle state
State 1: detecting the start bit and baud rate State 0: idle state after data reception
This ensures that the system can correctly identify the start bit, change baud rate, and process the received data accordingly.

4. Clock and Reset Behavior

The system clock (clk) ensures that there is a constant time for the data sampling.
The reset signal (rst_n) initializes the system properly to ensure stable operation when released.

Key Findings

The experimental results show that UART dynamically detects and adapts to an incoming baud rate without the need for predefined configurations. The synchronization process involved eliminates errors due to variability in baud rates. FSM transitions indicate the correct acceptance of start bits, adaptation to transmission speeds, and back to idle state when no data is being transmitted. These results confirm that the adaptive baud rate detection mechanism is effective for robust UART communication at different transmission speeds.

CONCLUSION

This paper explains the definition, functionality, and modular breakdown of UART along with verification of the program's feasibility through simulation. The experiments have successfully demonstrated the implementation of UART using Verilog HDL with the simulation of transmission and reception modules while considering the speed and accuracy. It was able to achieve high-speed and precise data exchange between the computer and peripheral devices. The system supports full-duplex communication, ensuring an efficient, clear, accurate, and stable data transmission process. This paper offers valuable insights for the further development of UART.

REFERENCES

[1] Kamath, A., Mendez, T., Ramya, S., & Nayak, S.
[2] G. "Design and Implementation of Power-Efficient FSM based UART." Journal of Physics: Conference Series, 2022, 2161(1), 012052.
[3] Thai, N. P., Ngoc, B. H., Duy, T. D., Truong, P. Q., & Phan, V. C. "A novel multichannel UART design with FPGA-based implementation." International Journal of Computer Applications in Technology, 2021,
[4] R, Y., & V, R. M. "Design and Verification of UART using System Verilog." International Journal of Engineering and Advanced Technology, 2020, 9(5), 1208–1211.
[5] Kashyap, B., & Ravi, V. "Universal Verification Methodology Based Verification of UART Protocol." Journal of Physics: Conference Series, 2020, 1716(1), 012040.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

[6] International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958 (Online), Volume-9 Issue-5, June 2020.
[7] Kumari Amrita, Avantika Kumari. "Design And Verification Of UART Using Verilog HDL." International conference on Recent innovations in Management, Engineering, Science and Technology, RIMEST, 2018.
[8] Wei Ni, Xiaotian Wang. "Functional Coverage-Driven UVM-based UART IP Verification." IEEE 11th International Conference on ASIC (ASICON), 21 July 2016.
[9] Amrit P. Singh, Khushboo Gupta, & Juhee Mala. "Determination of UART Receiver Baud Rate Tolerance." International Conference on Computer and Communication Technology, 2015, 265–270, 2818655.
[10] Spear, Chris, Tumbush, Greg. "SystemVerilog for Verification" Book.
[11] Fang, Y. Y., & Chen, X. J. "Design and Simulation of UART Serial Communication Module Based on VHDL." 2011 3rd International Workshop on Intelligent Systems and Applications, 2011, 5873448.
[12] Han, X., & Kong, X. "The Designing of Serial Communication Based on RS232." 2010 First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems, 2010, 80.
[13] Choi, J. I., Jain, M., Srinivasan, K., Levis, P., & Katti, S. "Achieving single channel, full duplex wireless communication." Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking - MobiCom '10, 2010, 1859997.
[14] Wu, J., Ma, Y., Zhang, J., Kong, Y., Song, H., & Han, X. "Research on metastability based on FPGA." 2009 9th International Conference on Electronic Measurement & Instruments, 2009, 5274693.