UART Tx & Rx Controller Design and Simulation using Verilog

By Tanya Priyadarshini

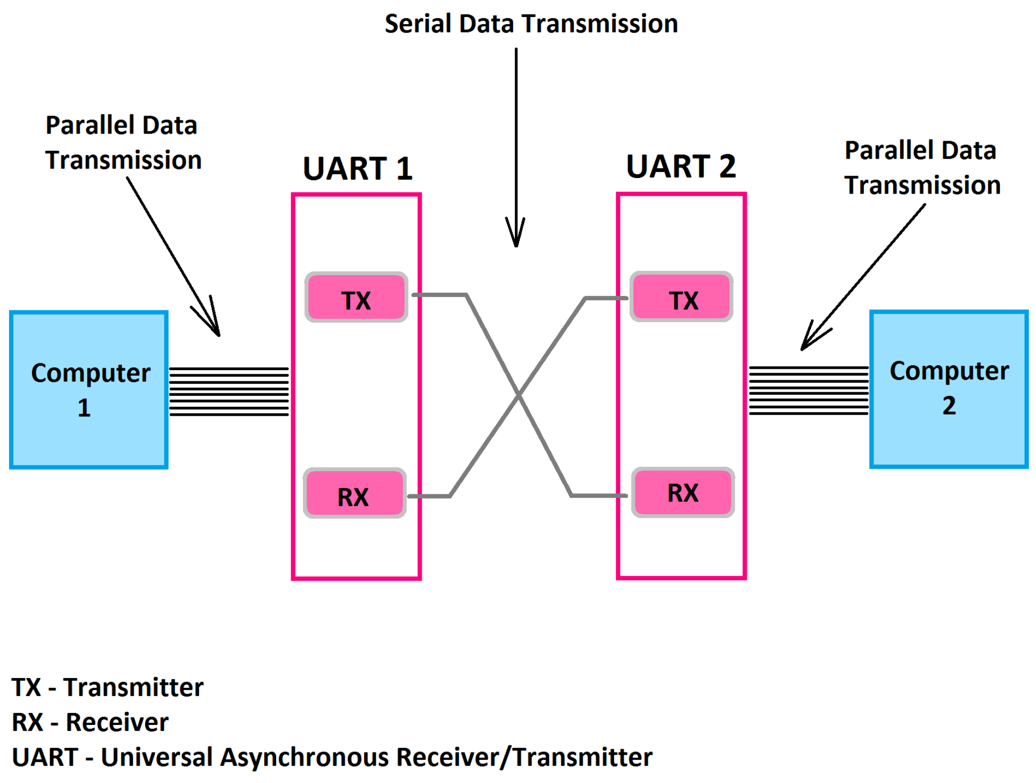
Introduction

UART (Universal Asynchronous Receiver and Transmitter) is a widely used serial communication protocol that allows for full-duplex data transmission between two devices without the use of a clock signal. It converts parallel data from the CPU or memory into serial form for transmission and vice versa for reception. This project involves the design, implementation, and simulation of a UART Transmitter and Receiver in Verilog, targeting communication at 115200 baud with a 25 MHz clock.

Objective

To design and simulate a functional UART transmitter and receiver module in Verilog that can:

- Transmit and receive 8-bit data serially.
- Detect start and stop bits.
- Operate asynchronously with oversampling support.
- Ensure reliable and synchronized communication.


Serial Data Transmission
Parallel Data Transmission
UART 1
UART 2
Parallel Data Transmission
TX
TX
Computer 1
Computer 2
RX
RX
TX - Transmitter
RX - Receiver
UART - Universal Asynchronous Receiver/Transmitter

UART Block Diagram

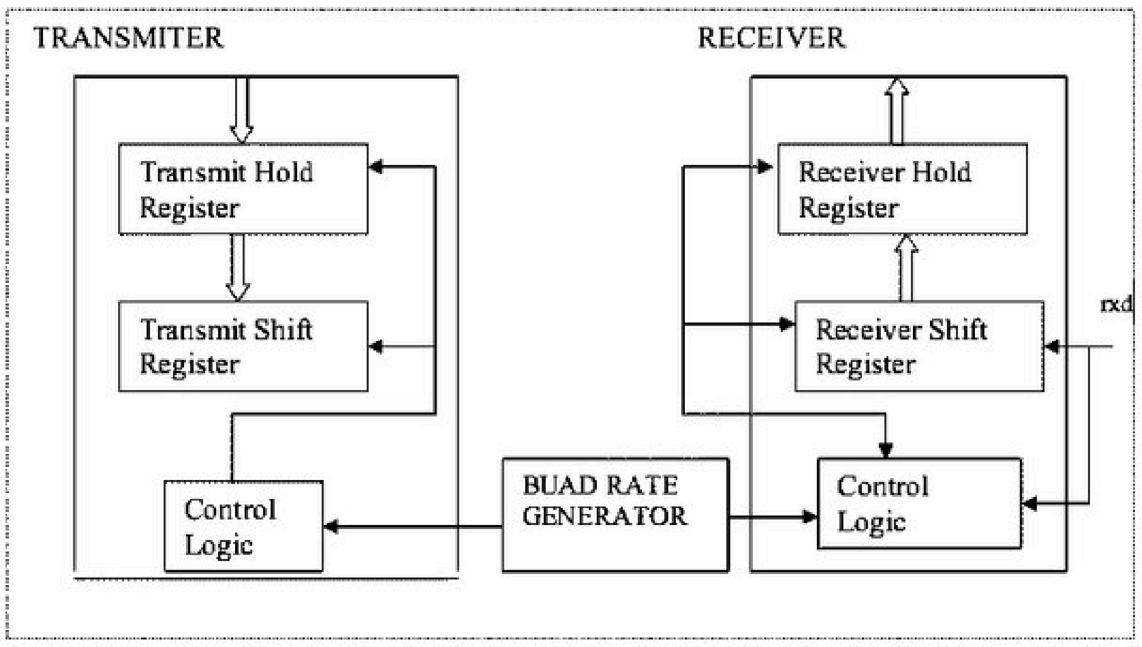
Working Principle of UART

UART communication is asynchronous, meaning no external clock signal is shared between transmitter and receiver. Synchronization is achieved using the structure of the data frame, which includes:

- **1 Start Bit**: Indicates the beginning of a data frame (transition from high to low).
- **5 to 8 Data Bits**: Actual data being transmitted.
- **Optional Parity Bit**: Used for error detection.
- **1 or 2 Stop Bits**: Indicates the end of the data frame.

When the receiver detects a start bit, it begins sampling the incoming bits at a predefined rate (the baud rate). In this project, a **16x oversampling** technique is used to ensure accurate sampling even in the presence of noise or drift.

Block Diagram of UART



Design Details

UART Transmitter

The UART transmitter performs the following:

- Waits for `i_Tx_Ready` to be asserted.
- Adds a **start bit (0)**, transmits 8 bits LSB first, and ends with a **stop bit (1)**.
- Signals `o_Tx_Done` after transmission is complete.

UART Receiver

The UART receiver:

- Detects a start bit (falling edge on `i_Rx_Data`).
- Uses 16x oversampling to sample at the midpoint of each bit.
- Reconstructs the 8-bit data and asserts `o_Rx_Done`.

Simulation Environment

- **Clock Frequency**: 25 MHz
- **Baud Rate**: 115200
- **Clocks Per Bit**: 217
- **Oversampling Factor**: 16
- **Simulator Used**: Icarus Verilog (via EDA Playground)
- **Testbench**: A self-checking testbench was written to send 8 pre-defined bytes and verify the received bytes match the transmitted ones.

Simulation Environment

- **Clock Frequency**: 25 MHz
- **Baud Rate**: 115200
- **Clocks Per Bit**: 217
- **Oversampling Factor**: 16
- **Simulator Used**: Icarus Verilog (via EDA Playground)
- **Testbench**: A self-checking testbench was written to send 8 pre-defined bytes and verify the received bytes match the transmitted ones.

Advantages of UART Communication

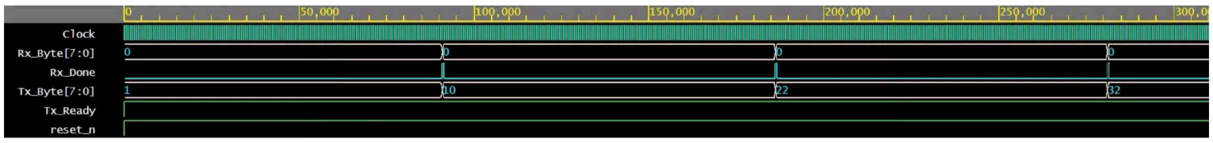
- Requires only two signal lines (Tx and Rx).
- No need for a separate clock signal.
- Allows for error checking using parity.
- Simple and cost-effective for short-range, point-to-point communication.

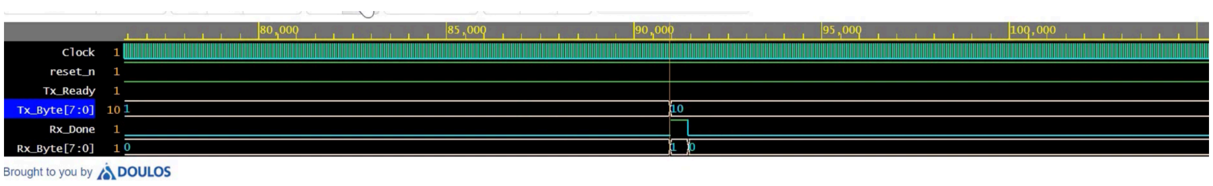
Limitations of UART

- Supports only one transmitter and one receiver at a time.
- Lower throughput compared to parallel communication.
- Frame size is limited (typically 8 data bits).
- Both devices must agree on the same baud rate and frame format.

Waveform and Output Analysis

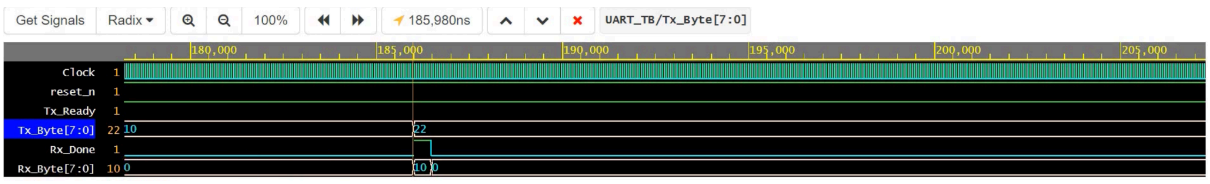
The functionality of the UART Transmitter and Receiver was verified using simulation waveforms generated from a testbench implemented in Verilog. The simulation involved transmitting a series of 8-bit data values through the UART interface and confirming their correct reception. The testbench provided visual validation through signal observation of key control and data lines such as `Tx_Byte`, `Rx_Byte`, `Tx_Ready`, and `Rx_Done`.





In this waveform, a stream of three bytes is transmitted in sequence:

- The `Tx_Byte` signal changes from `0x01` to `0x22`, then to `0x32`, with appropriate spacing.
- Each byte transmission is initiated by asserting `Tx_Ready`.
- After each transmission, the `Rx_Done` signal pulses, confirming the reception of a complete byte.
- The `Rx_Byte` signal updates accordingly to reflect `0x01`, `0x22`, and `0x32` at each instance
- The waveform confirms that the UART system can correctly handle continuous byte transfers without data corruption or misalignment.



This waveform shows the UART transmitting the byte `0x22`:

- The signal `Tx_Ready` is asserted high, allowing the transmitter to initiate data transmission.
- `Tx_Byte` is updated to `0x22` and held stable during the transmission period.
- The receiver detects the start bit and begins sampling using 16x oversampling.
- After a complete byte is received, the `Rx_Done` signal pulses high.
- The `Rx_Byte` signal then reflects the received value `0x22`, which matches the transmitted byte.

This validates the start-to-finish correctness of a single-byte transmission.

Conclusion

The Verilog-based UART transmitter and receiver were successfully designed and simulated. The waveform results confirm that the system reliably transmits and receives 8-bit data using asynchronous serial communication with 16x oversampling. This project demonstrates a complete, functioning UART system suitable for integration in FPGA or ASIC designs.