UART MODULE CONTROLLED BY A STATE MACHINE

PROJECT REPORT ON

A UART MODULE WITH A STATE MACHINE IN VHDL, ENSURING EFFICIENT ECHOING OF CHARACTERS RECEIVED FROM A PC OVER RS232.

By : Nabil OULHAJ

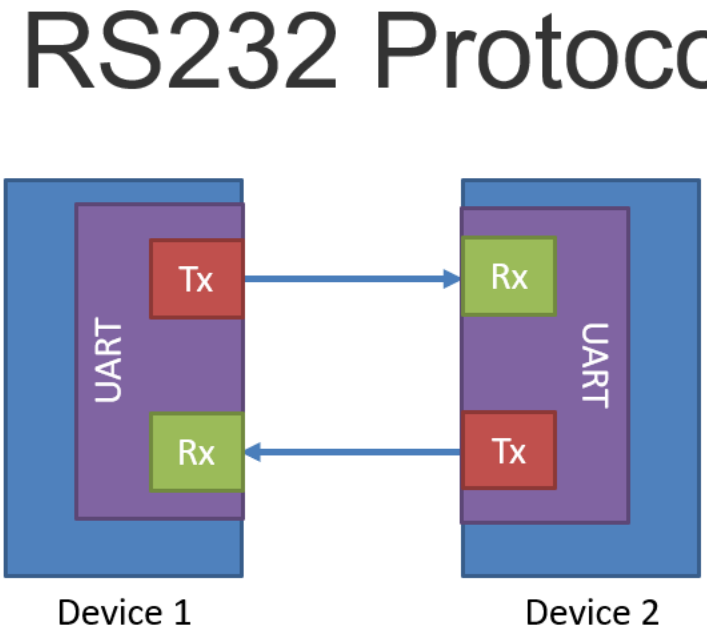
LinkedIn : @nabiloulhaj

Email : nabiloulhaj@outlook.com

2023/2024

1. Introduction

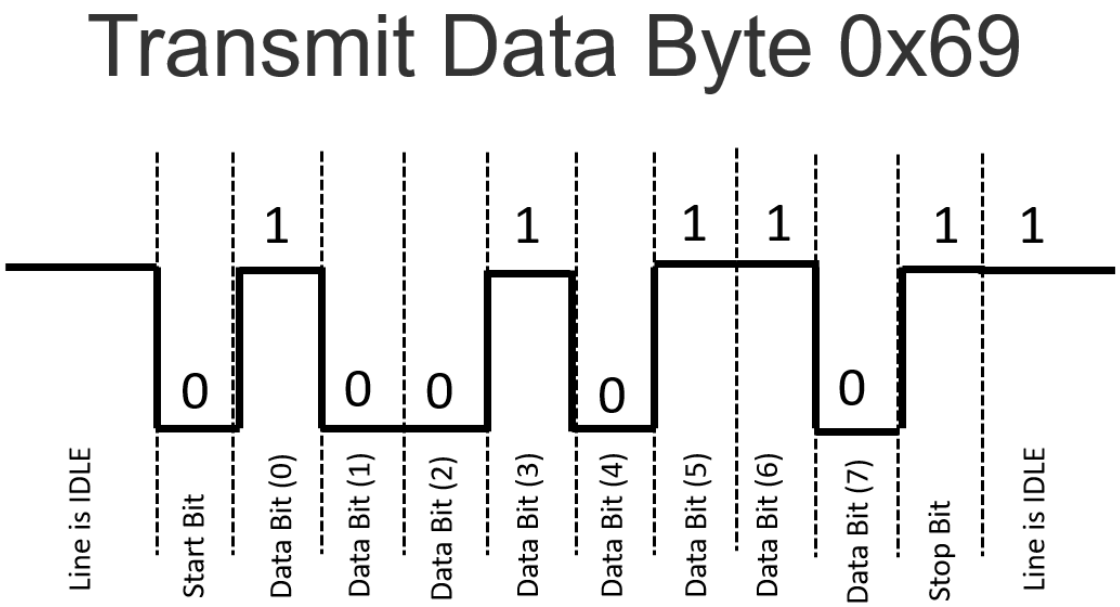
The current project uses RS232 to develop UART (Universal Asynchronous Receiver/Transmitter) connection over RS232 with a PC in an effort to create a strong communication framework. A popular serial communication protocol called UART makes it easier for electronic equipment to communicate data asynchronously. The serial communication transmission standard RS232 is used as the conduit for this data exchange between the PC and embedded system.



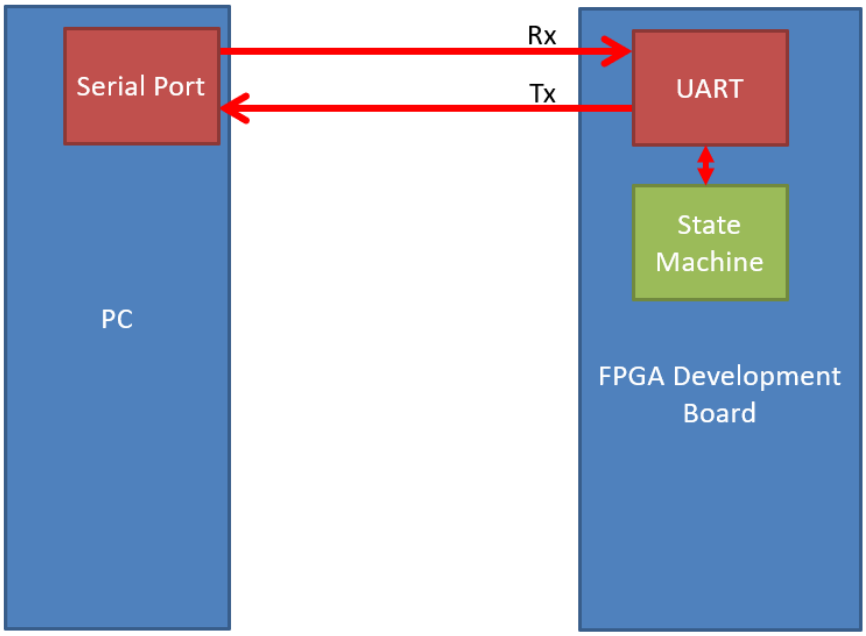
The main goal of this project is to provide a dependable and effective UART-based communication system that facilitates easy connection between a PC and an embedded system. The selection of RS232 as the communication interface was based on the protocol's historical popularity and ongoing applicability as a means of facilitating communication between different electronic equipment.

Start Bit	Data Bits	Parity Bit	Stop Bit(s)

Start Bit = Supports 1 start bit. Logic 0.
Data Bits = Supports 5 to 9 data bits
Parity Bit = Optional (Even or Odd Parity)
Stop Bits = Supports 1, 1.5 or 2 stop bits. Logic 1.



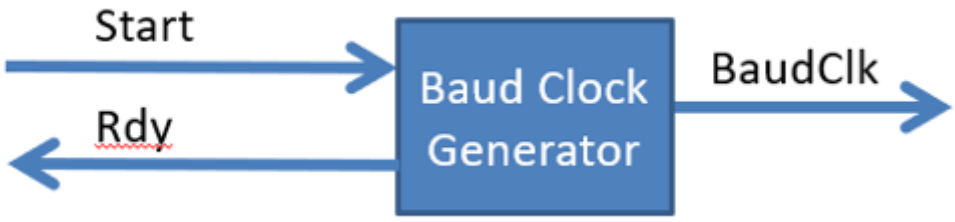
The main goal of RS232-based UART connection is to allow data transmission in both directions between the PC and embedded system. The embedded system will be able to send and receive data in a serial format thanks to this communication link, enabling a flexible and effective method of communication. A flexible and standardized communication channel is what the UART over RS232 solution attempts to provide, whether for sensor data acquisition, control commands, or other applications that require a lot of data.



Through this project, we want to learn a great deal about the complexities of serial communication protocols in addition to successfully implementing the UART via RS232. We expect a thorough grasp of the difficulties and subtleties involved in developing and putting into practice a dependable communication link as we dig into the technical details of Baud rate generation, parallel-to-serial and serial-to-parallel conversions, and the integration of essential UART components.

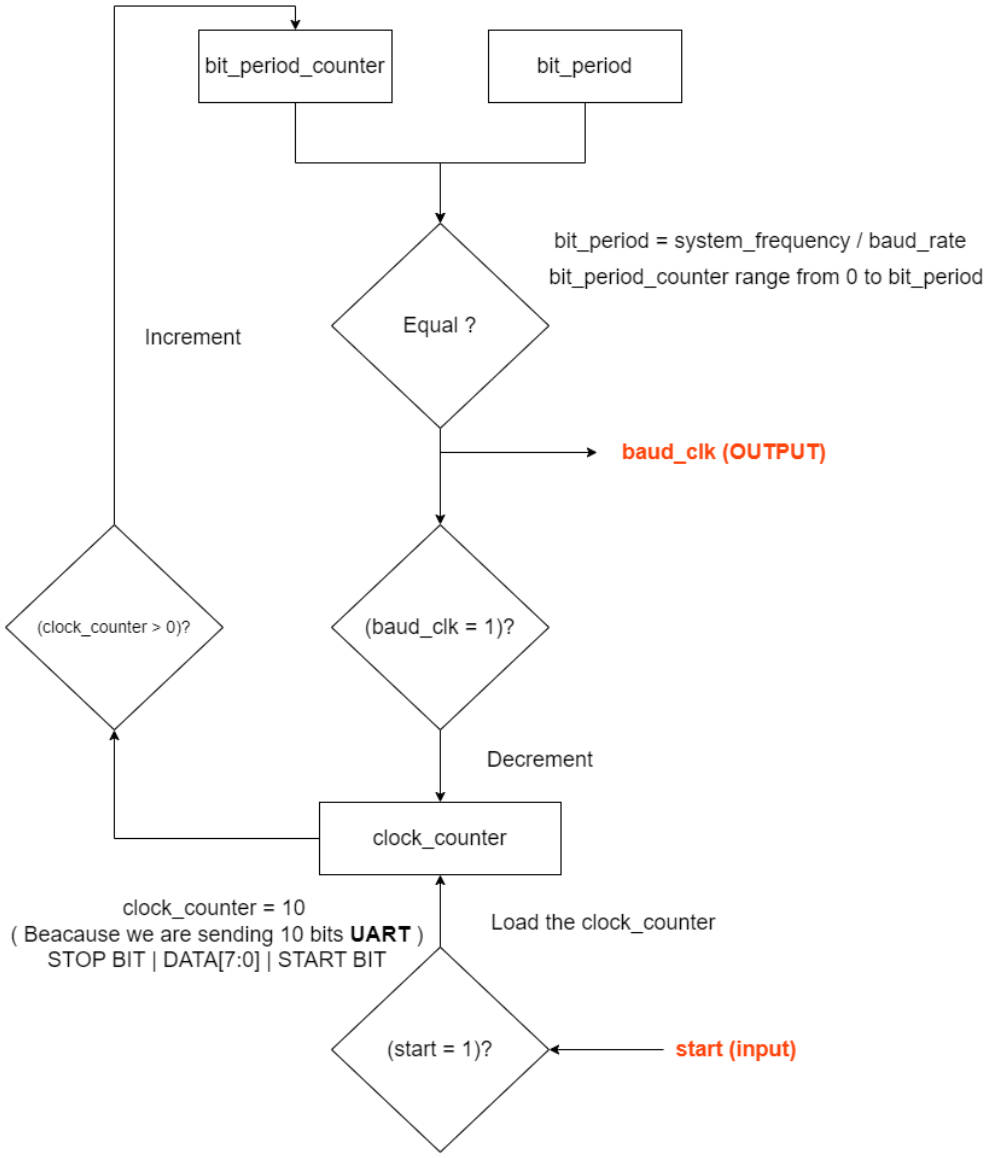
2. Circuit Design

- Baud CLock Generator



A clock signal with the proper frequency is essential for attaining the precise synchronization that the field of digital communication requires. The architecture and operation of a Baud Clock Generator are written in VHDL, a language commonly used to describe hardware for FPGA and ASIC designs. The

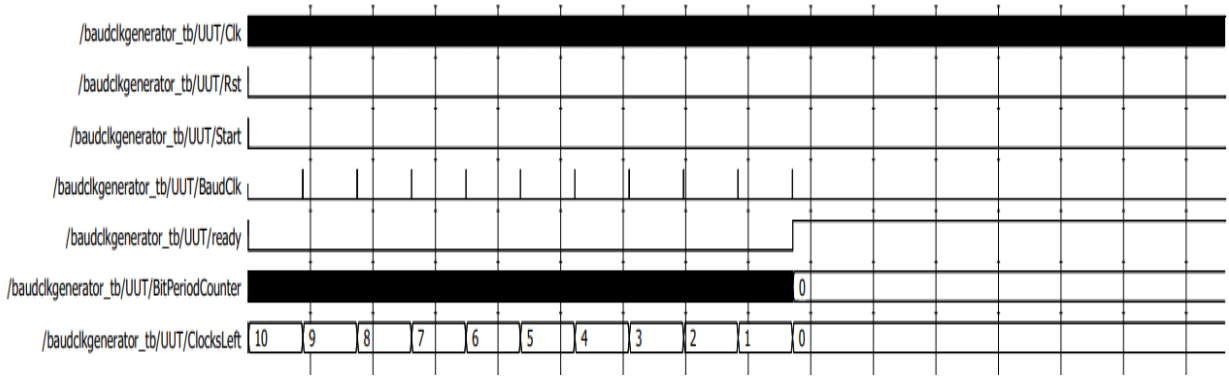
purpose of this baud clock generator is to generate a clock signal appropriate for a given baud rate, which is a crucial variable in serial communication.



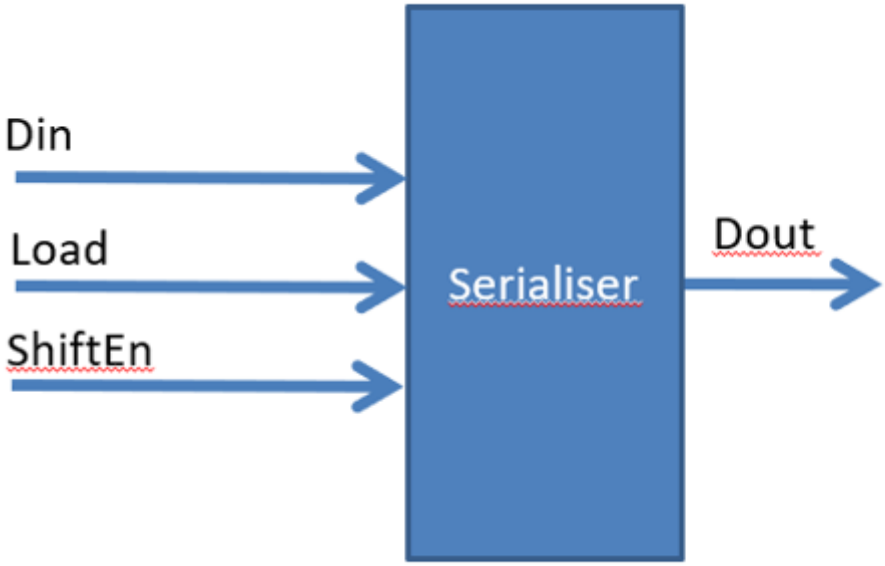
The number of signal changes per second, or baud rate, determines the speed at which data is sent. An extremely precise frequency-controlled clock signal is essential to this connection. The logic for a Baud Clock Generator is contained in the VHDL code that is provided here, providing a flexible solution for a range of digital communication systems.

After applying a system reset, the Baud Clock Generator test bench starts the generator to begin the simulation. The stimuli supplied for the Rst and Start signals replicate a common situation in which the generator is turned on

following a brief initialization phase. Throughout the simulation, the clock (Clk) and output signals (Ready and BaudClk) are observed.



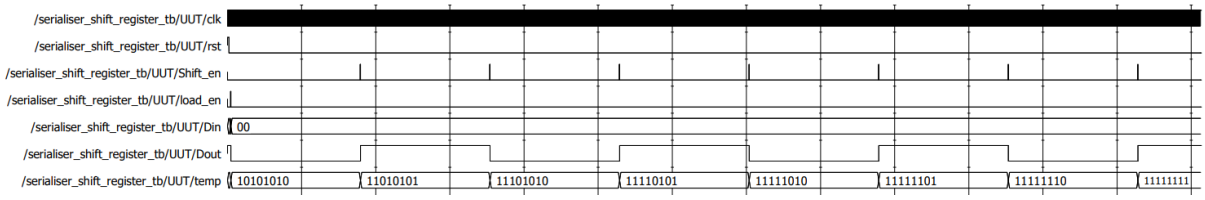
- Serialiser



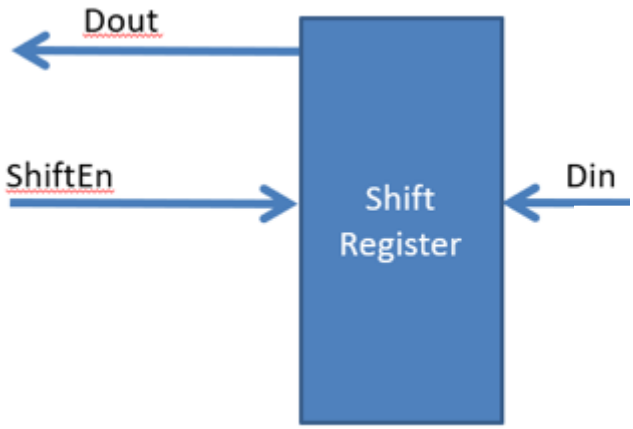
A Parallel-In Serial-Out (PISO) shift register is often referred to as a serializer shift register. The purpose of this module is to translate serial data (Dout) from parallel data (Din) using control signals for loading and shifting, reset conditions, and clock signals.

An essential part of the UART transmitter module is the Serializer Shift Register. It makes it possible to transmit parallel data by converting it into a serial stream. In order to manage the loading and moving of data, the control signals load_en and shift_en are essential. '1' is the default state when the module is reset.

This VHDL code is essential to the overall UART communication design since it encapsulates the functionality of a PISO shift register.



- Shift Register



In order to transform serial data into parallel format for further processing, the UART receiver's Shift Register is essential. Let's examine its role in these processes:

Receiving Serial Data: The Shift Register receives serial data as an input through its din port. The Shift Register receives each bit as it comes in from the serial data stream.

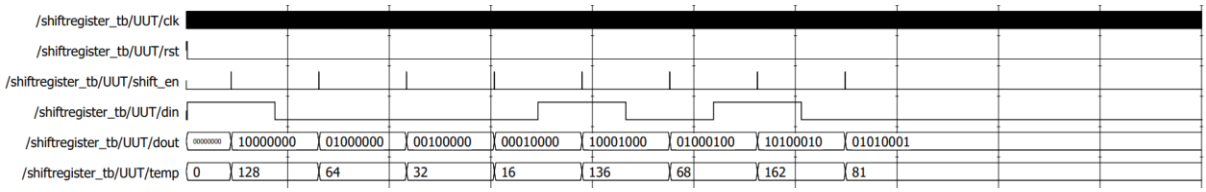
Clock Synchronization: The system clock (clk) and the shift register are in sync with each other. The shift operation is initiated by the clock's rising edge, which makes sure that data processing is synced and in line with the clock signal.

Dynamic Configuration with Reset: The first (reset) signal of the Shift Register permits dynamic configuration. The Shift Register is reset to a predetermined initial state and prepared to accept a fresh data packet upon the activation of the reset signal.

Configuring the Shift Direction: The shift_direction argument can be used to set the shift direction. The Shift Register moves data to the left (MSB first) and to the right (LSB first) depending on which setting it is set to ('R' or 'L').

Data Processing for Further Stages: The shifted parallel output (dout), which shows the Shift Register's current state, is produced when the Shift Register receives and analyzes incoming serial data. The UART receiver then uses this parallel output in later stages, which makes it possible to extract and handle the incoming data.

Versatile Application in UART Receiver: The dual-directional data shifting capabilities of the Shift Register allow for versatile UART communication setups, enabling the transmission of data in either the MSB (most significant bit) or LSB (least significant bit) order.



- Synchroniser

Synchronization is essential in the UART communication system to guarantee precise and dependable data receiving. The synchronizer module assists in lining up incoming data with the system clock. It is commonly built on flip-flops or shift registers. Let's examine how this synchronization process is aided by the supplied Shift Register with synchronization:

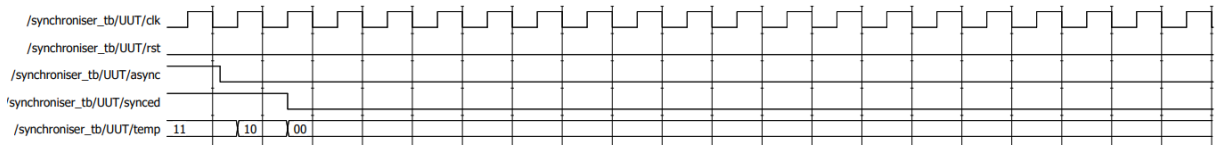
Time-Based Harmonization: The Shift Register's synchronization feature depends on the clock signal's rising edge (clk). This guarantees that everything that happens in the Shift Register, including data shifting, happens exactly at predetermined intervals of time.

Preventing Metastability: Reducing the impact of metastability is one of a synchronizer's main goals. There is a chance of metastability—undefined intermediate states that can happen during the transition between logic levels—when asynchronous signals reach the receiving clock domain. These dangers are reduced in part by the clock-driven procedures.

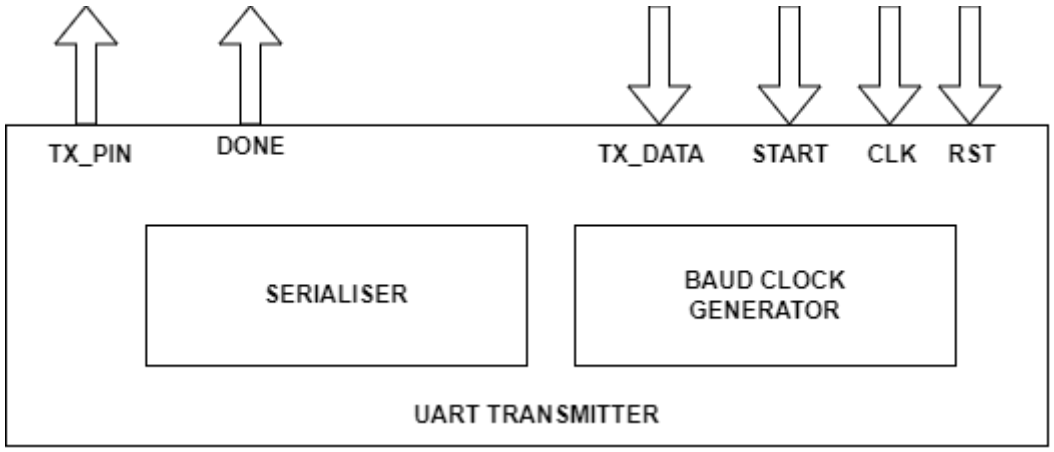
Increased Data Stability: The synchronizer module increases the stability of the received data by syncing incoming data to the system clock. In UART communication, where data integrity is critical, this is extremely important.

Clock Domain Crossing: The synchronizer makes sure that incoming data is reliably transferred to the receiver's clock domain without introducing potential issues like setup and hold time violations in systems with multiple clock domains, such as the UART receiver handling external serial data.

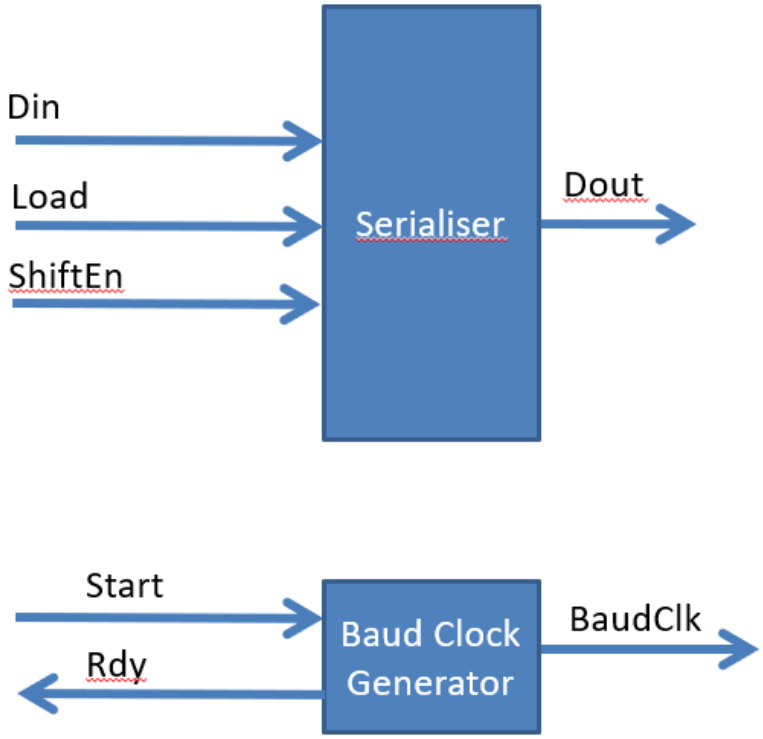
Integration with Shift Register: The received data can be synced as it is processed inside the Shift Register thanks to the synchronization logic's integration into the Shift Register module. This guarantees that the synchronized state of the incoming data is appropriately represented by the parallel output (dout).



3. UART Transmitter

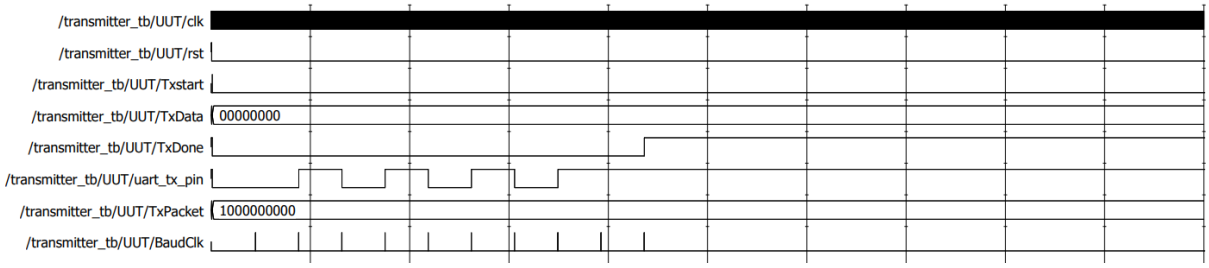


The crucial component in the process of transferring parallel data from the embedded system to a serial format that can be sent to a PC via the RS232 communication channel is the UART Transmitter. A number of essential elements are included in the design, each of which contributes in a different way to the efficient and dependable conveyance of data.

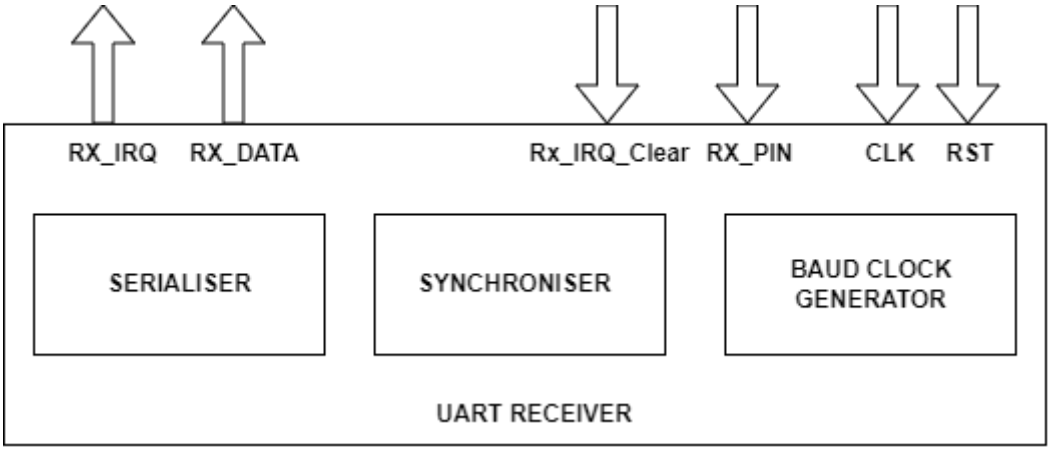


The parallel data obtained from the shift register is processed by the PISO Serializer, which then transforms it into a serial format fit for transmission. By doing this step, data is made sure to be properly formatted for serial communication.

One important aspect in UART communication is the rate at which bits are transmitted over the communication channel, which is determined by the Baud Rate Generator. The Baud Rate Generator ensures exact timing for bit transmission by producing clock pulses at a rate that matches the intended baud rate. For the transmitter and the PC to communicate in unison, this is necessary.



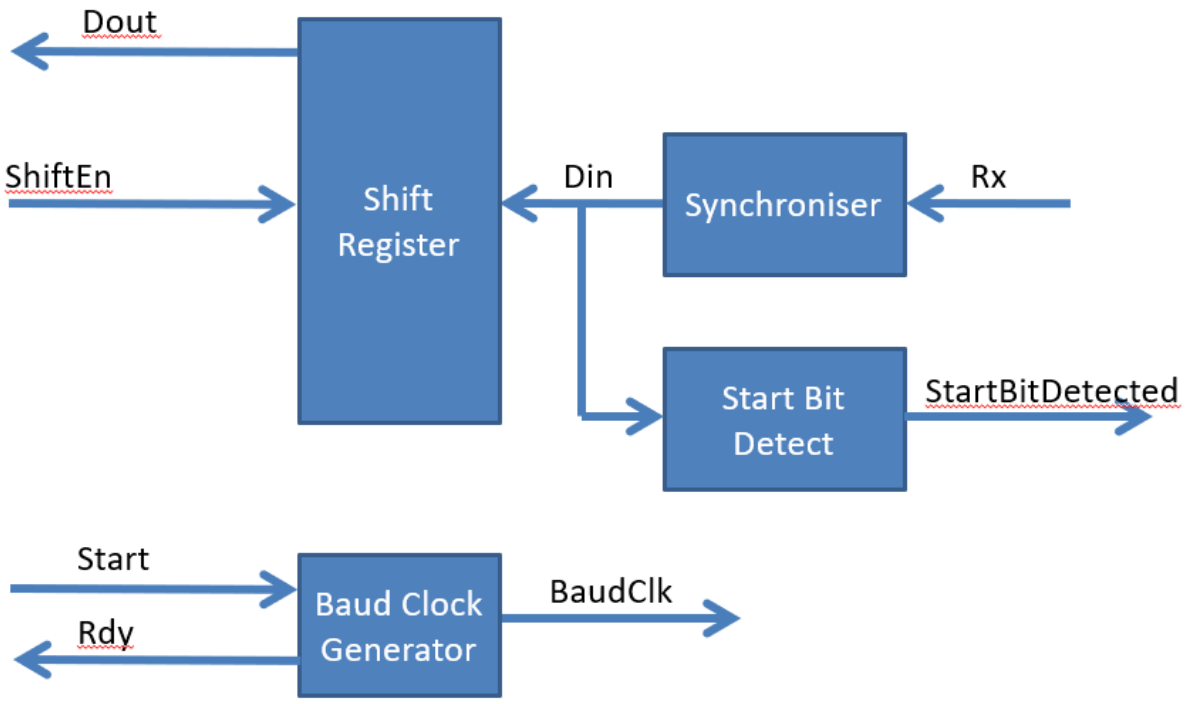
4. UART Receiver



An essential part of the system is the UART Receiver, which transforms the incoming serial data stream from the RS232 communication channel into parallel data that the embedded system can process. The receiver design consists of multiple essential modules that work together to ensure that transmitted data is received and used in an efficient manner.

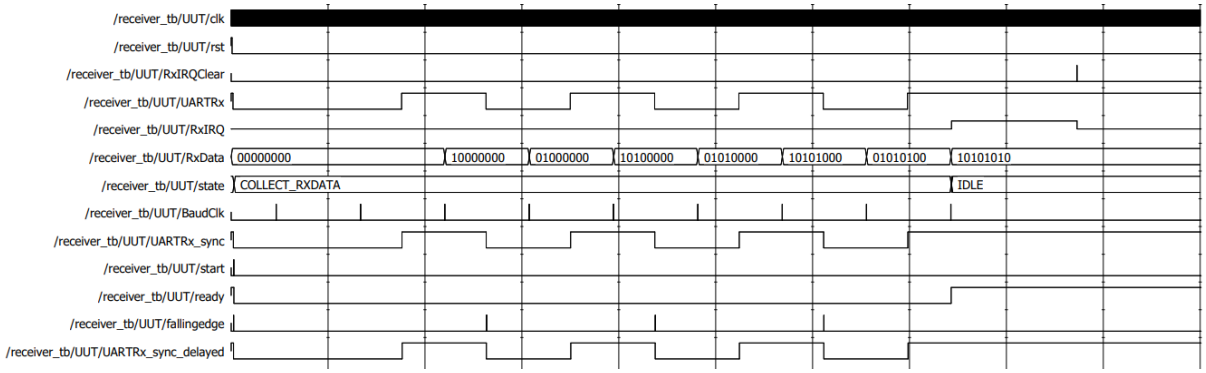
The Synchronizer makes sure that incoming data is synced with the clock domain of the receiver by operating on the rising edge of the clock signal (clk). This improves the stability of the received data and reduces the dangers associated with metastability.

The Synchronizer helps reduce the consequences of metastability by acting as a buffer between the receiver's clock domain and the asynchronous incoming data, guaranteeing a dependable and well-defined input for the receiver's subsequent stages.

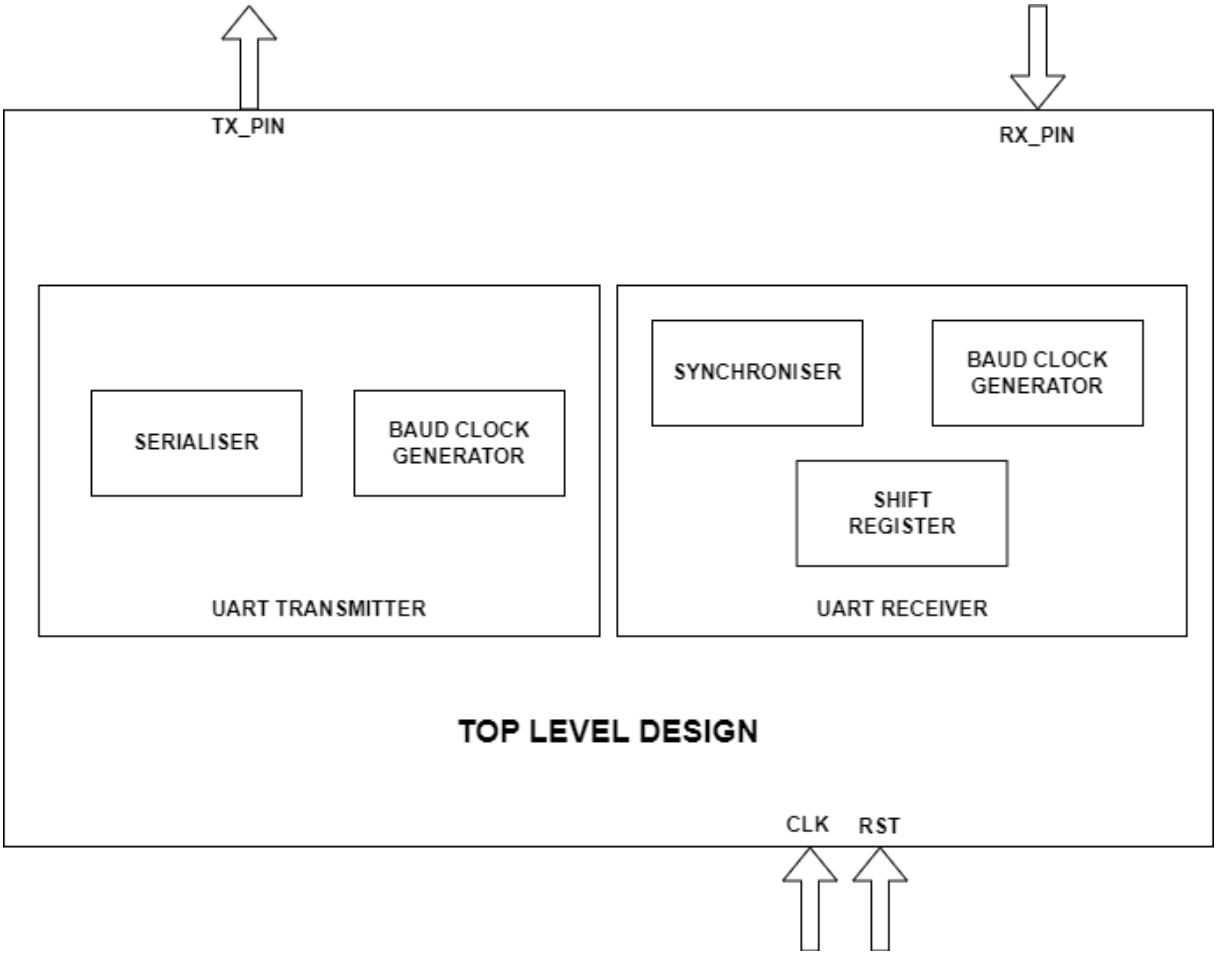


Like its counterpart in the transmitter, the shift register makes it easier to convert receiving serial data into parallel format. The Shift Register, which functions in the Serial-In Parallel-Out (SIPO) mode, processes the synchronized serial data and transforms it into parallel form so that the receiver can use it further.

To precisely ascertain the baud rate, the Baud Rate Detector examines the temporal properties of the incoming data stream. The receiver may accept different baud rates because to its adaptive capacity.



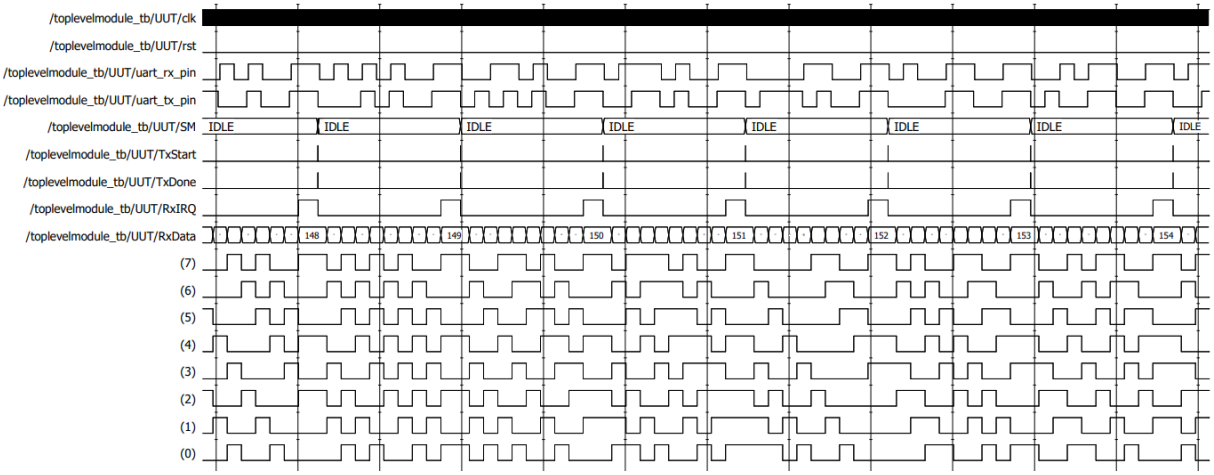
5. Top-Level Design



Through the integration of all the component parts, the top-level design creates a unified UART communication system that allows data transfer in both directions between the embedded system and a PC via the RS232 interface. The UART Transmitter and Receiver modules are combined in the top-level architecture to form a single data transmission system. The top-level design includes an Echo Mode feature that allows data to be received and echoed back to the PC. The system may receive data from the PC, process it using the UART Receiver, and then echo the data back through the UART Transmitter because Echo Mode is bidirectional. This feature increases the UART communication system's adaptability.

The Baud Rate Generator, Shift Register, Serializer (PISO), and Baud Clock Generator are all integrated within the UART Transmitter. Together, the modules transform parallel data from the embedded system into a serial format that can be sent through the RS232 interface.

The Synchronizer, Shift Register, Deserializer (SIPO), and Baud Rate Detector are all smoothly integrated into the UART Receiver. The processing and receipt of serial data supplied from the PC are guaranteed by this integration.



6. Conclusion

As a result of this UART communication system design, a thorough and effective solution has been developed to enable smooth bidirectional data transfer via the RS232 interface between an embedded system and a PC. The UART Transmitter and Receiver are among the essential parts that have been integrated to provide a dependable, flexible communication system with adaptive capabilities.

Parallel-to-Serial Conversion: Efficient transmission across the RS232 interface is ensured by the UART Transmitter's successful conversion of parallel data from the embedded system into a serial format. This procedure is more accurate and reliable since it incorporates modules like the serializer, baud clock generator, shift register, and baud rate generator.

Serial-to-Parallel Conversion: By employing modules like the Synchronizer, Shift Register, Deserializer, and Baud Rate Detector, the UART Receiver effectively handles incoming serial data. By guaranteeing synchronized reception and precise data parallelization, the design strengthens the communication system's resilience.

Baud Rate Synchronization: Accurate timing and synchronization are ensured by the transmitter and receiver having a common knowledge of baud rates,

which is made possible by the Baud Rate Generator and Baud Rate Detector. The successful interpretation of conveyed data depends on this attribute.

Echo Mode Capability: The system's bidirectionality, demonstrated by Echo Mode, increases adaptability. This feature demonstrates the versatility and flexibility of the UART communication system by enabling received data to be repeated back to the PC.