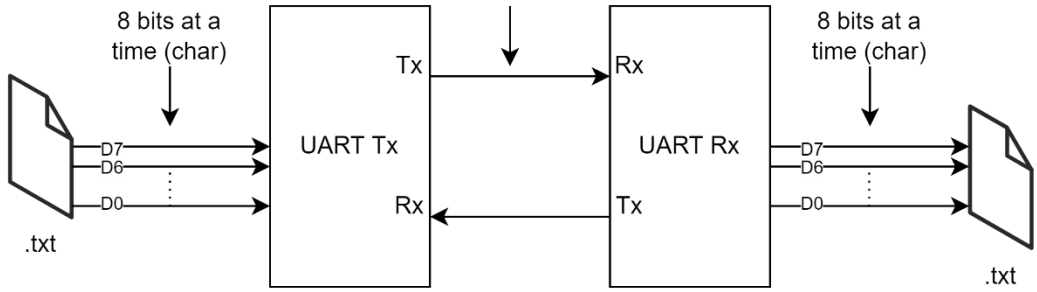
CS2204 Project D

Serial Communication using UART

This project will involve designing and implementing Tx and Rx controller using UART



Serial Communication: Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. UART stands for Universal Asynchronous Receive Transmit.

Theory

In serial communication, data is transferred bit by bit using a single line or wire. In two-way communication, we use two wires for successful serial data transfer. Depending on the application and system requirements, serial communications needs less circuitry and wires, which reduces the cost of implementation.

Embedded systems, microcontrollers, and computers mostly use UART as a form of device-to-device hardware communication protocol. Among the available communication protocols, UART uses only two wires for its transmitting and receiving ends.

The transmitting UART is connected to a controlling data bus that sends data in a parallel form. From this, the data will now be transmitted on the transmission line (wire) serially, bit by bit, to the receiving UART. This, in turn, will convert the serial data into parallel for the receiving device.

UART Frame

1 Start Bit	5 to 9 data bits	1 or 2 Stop Bits

START BIT

The UART data transmission line is normally held at a high voltage level when it's not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

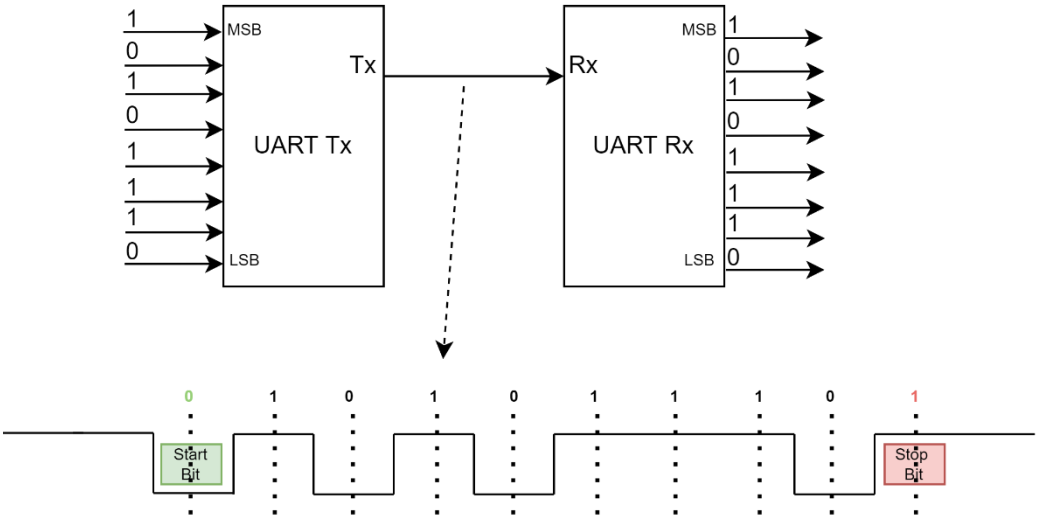
DATA FRAME

The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In most cases, the data is sent with the least significant bit first.

STOP BITS

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.

Eg. To transmit the sequence **"10101110",** the serial line would look as follows



Task

This project is broken down into 3 major milestones.

Milestone	Description	Progress
Task 1	Design and code the UART Tx/Rx FSM with given specs	70%
Task 2	Add Parity Check to Task 1	10%
Task 3	Testing and verification using .txt or .jpg files	20%

Milestone 1:

Design a UART Tx and a UART Rx Controller with the following specifications.

1 Start Bit

1 Stop Bit

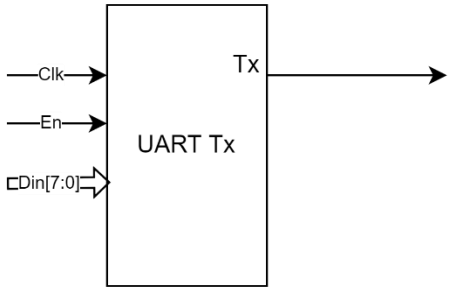
8 Data Bits

MSB first

(Assume any baud rate for this milestone)

Design an FSM that can take the parallel data and convert it to a stream of serial bits by appending a start and a stop bit to it.

Hint: Your Tx Module should look something like follows.



Working: When en=1, the module will store the data on the Din[7:0] line and transmit it bit by bit on the Tx line till all the Din bits as well as the stop bit is transmitted.

Number of clock cycles required to transmit 8 bits = 1(start bit) + 8(data bits) + 1(stop bit)

= 10 cycles

After 10 cycles, the En input signal is checked again and the same procedure is followed if En=1 else the module remains idle.

FSM: Please try and draw the FSM diagram which can implement this design before you move onto the implementation.

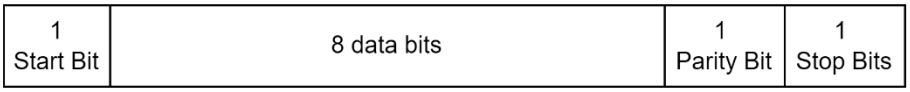
(The FSM diagram is not shown here. Please try it out yourself once then I will share the reference for comparison)

Milestone 2:

Add parity bit check to your design.

Parity: Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission.

After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a 0 (even parity), the 1 bits in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bits in the data frame should total to an odd number. When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.



This would only require adding one extra state to your FSM design!!!

Milestone 3:

This milestone requires us to test the design by taking inputs from files. This is a verification method used heavily in the industry and great way to enhance your testing skills.

This part was not covered in the coursework and will be taken up during a zoom session whenever requested.