UART

Introduction:

UART is a protocol that can be applied to any transmitter and receiver for data communication.
Universal - The protocol can work with any compatible transmitter and receiver.
Asynchronous - No clock signal is used for data communication. Instead, the devices agree on a data rate in advance and synchronize only during the start and stop bits of each data frame.
When two devices communicate using UART, they need at least two connections between them: one for transmission (TX) and one for reception (RX).
UART is a character-oriented protocol, meaning data is sent byte by byte. Each communication uses a specific bit frame structure, including start bits, data bits, optional parity bits, and stop bits. This bit frame structure allows consistent data exchange between devices.

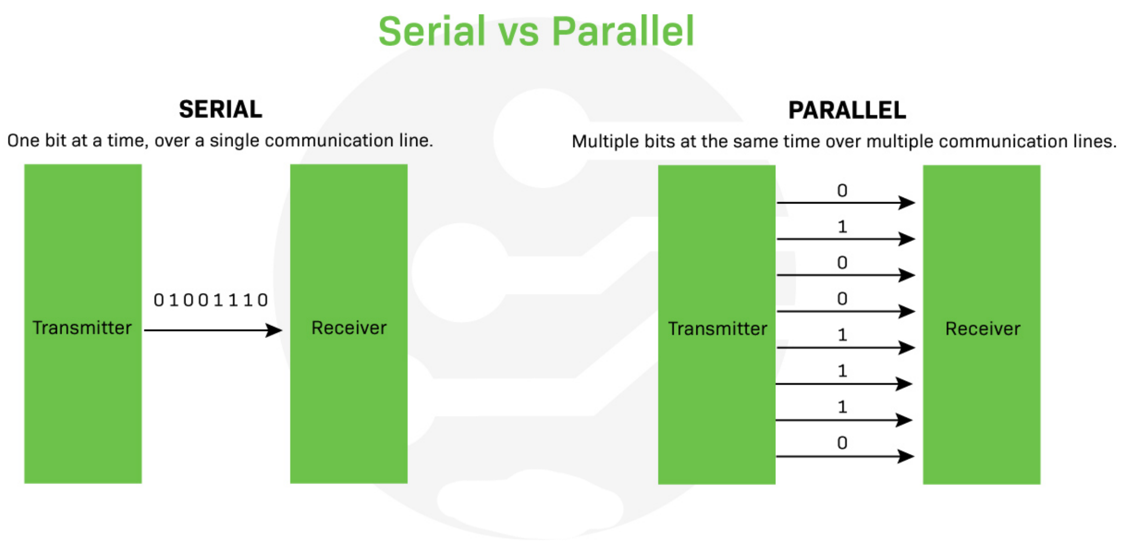
Serial Communication and parallel Communication:

Parallel communication:

- High Speed Communication
- Generally, Processor modules communicating in parallel

Serial communication:

- Low speed communication
- Long distance communication, like computer to other peripherals


Serial vs Parallel
SERIAL
One bit at a time, over a single communication line.
Transmitter
01001110
Receiver
PARALLEL
Multiple bits at the same time over multiple communication lines.
Transmitter
0
1
0
0
1
1
1
0
Receiver

Limitations of parallel communication & Advantage of Serial communication

Generally, computer process the data in parallel and speed of the communication is high.

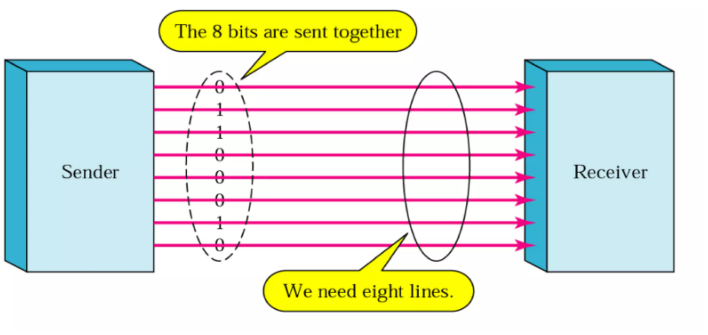
Limitation of Parallel:

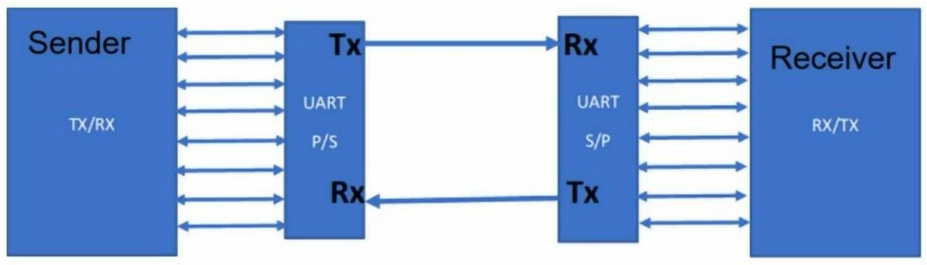
- When we want transfer the data for a longer distance (> 1 feet)
- There may be difference in reaching of bits to the other end due to difference in thickness of the wire of that parallel lines

- Difference can be recognized when speed of operation (clock speed) is more
- So, there is chance to getting errors

Then what you do?

- There is need to convert parallel → serial
- after transferring for a distance
- at receiver, convert serial → Parallel



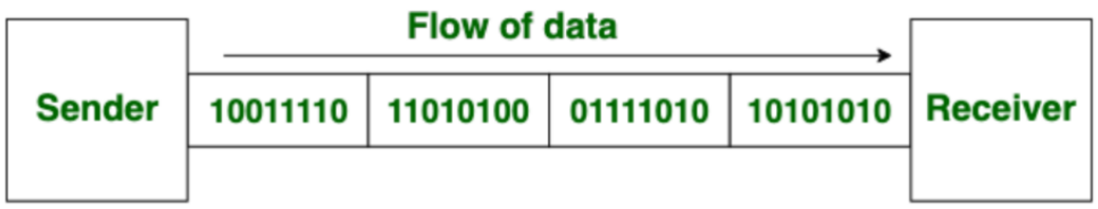


Synchronous & Asynchronous Serial Communication:
Synchronous:

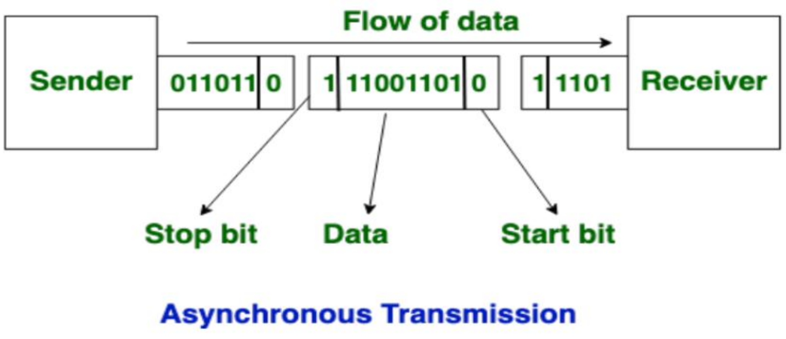
- Transfer Operation takes place using clock signal
- Neg-edge of clock, Transmits data from sender and next pos-edge of clock, Receives data at receiver

Asynchronous:

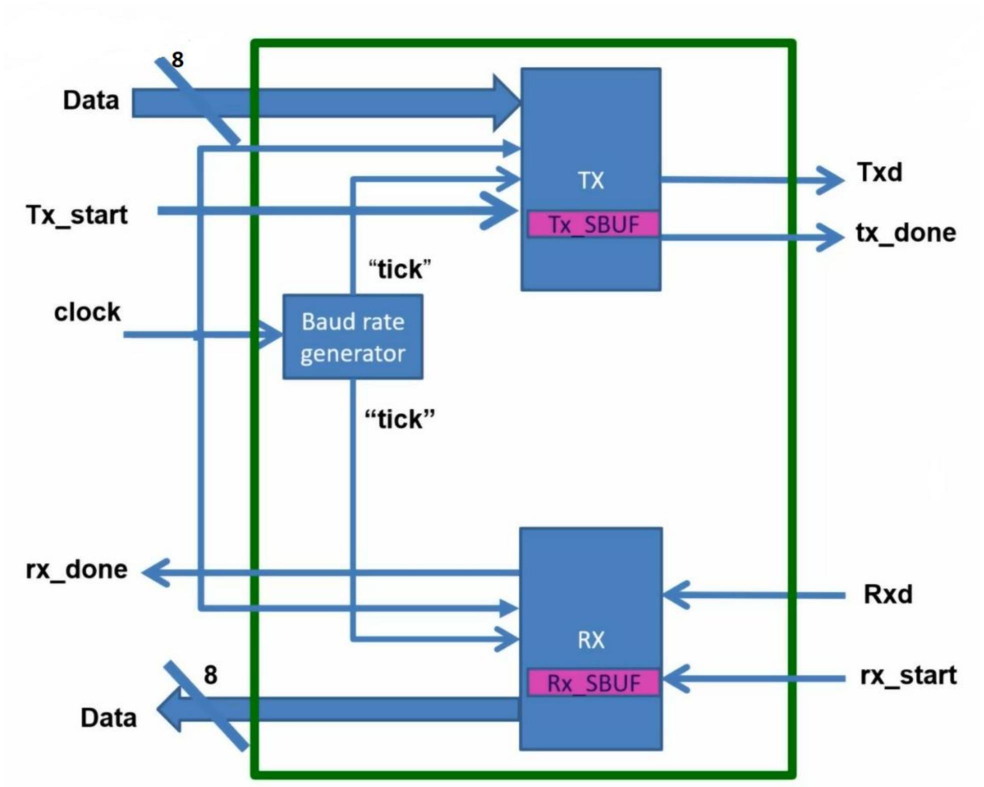
- No Clock
- START BIT(0), STOP BIT(1)
- Data is provided between these two



Synchronous Transmission



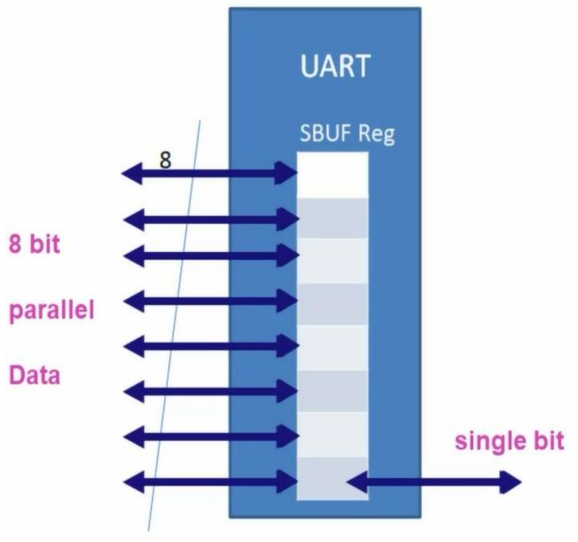
UART TOP Block Diagram:



What is an UART:
Universal Asynchronous Receiver Transmitter

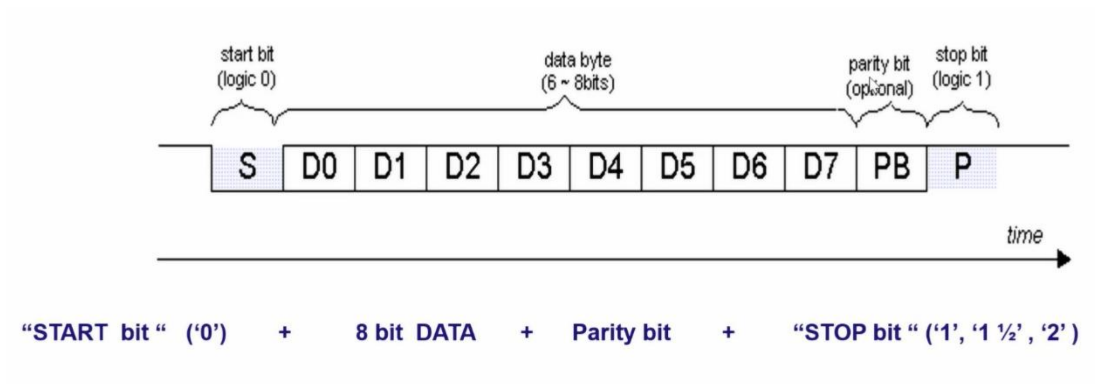
UART is an Integrated Circuit (IC)
It USES to converts
parallel data $\longleftrightarrow$ Serial Data

SBUF is a Serial Buffer, is a shift register
Parallel $\longrightarrow$ Serial: Shifting & transmits
Serial $\longrightarrow$ Parallel: Receiving & shifting



Frame format of UART:

for



Buad rate generator UART:

Baud rate defined as bits/sec, generally, baud rates are like 4800, 9600, 19200.

Let clock signal clk = 50 MHz = 50 x 10$^{\wedge 6}$

Let us say we required baud rate of data in UART is 19200

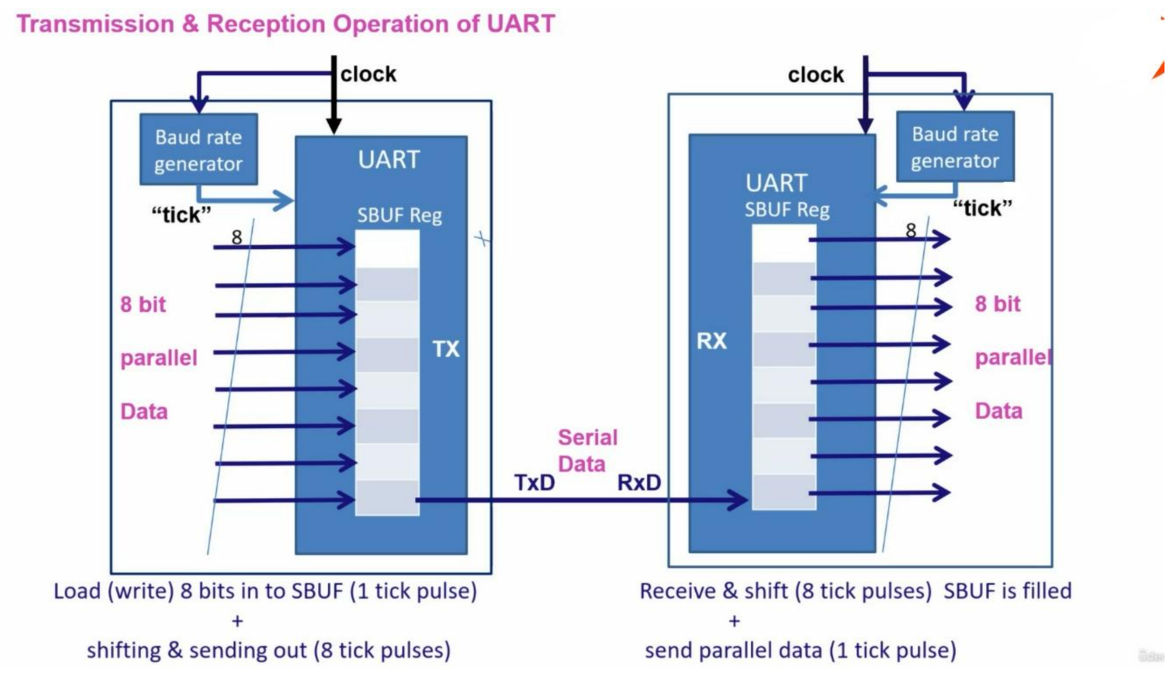
So, based on our requirements, pulses will be generated at every = (50x106)/(19200)= 2604

So, we need to design a counter Modulo-2605, i.e., counter counts (0-to-2604)

At every count value = 2604, one pulse need to generate, named as "tick" signal

At every "tick" signal, data need to send or receive.

Transmission and Reception:


Transmission & Reception Operation of UART
clock
Baud rate generator
"tick"
UART
SBUF Reg
8
8 bit
parallel
Data
TX
Serial
Data
TxD
RxD
clock
Baud rate generator
"tick"
UART
SBUF Reg
RX
8
8 bit
parallel
Data
Load (write) 8 bits in to SBUF (1 tick pulse)
+
shifting & sending out (8 tick pulses)
Receive & shift (8 tick pulses) SBUF is filled
+
send parallel data (1 tick pulse)

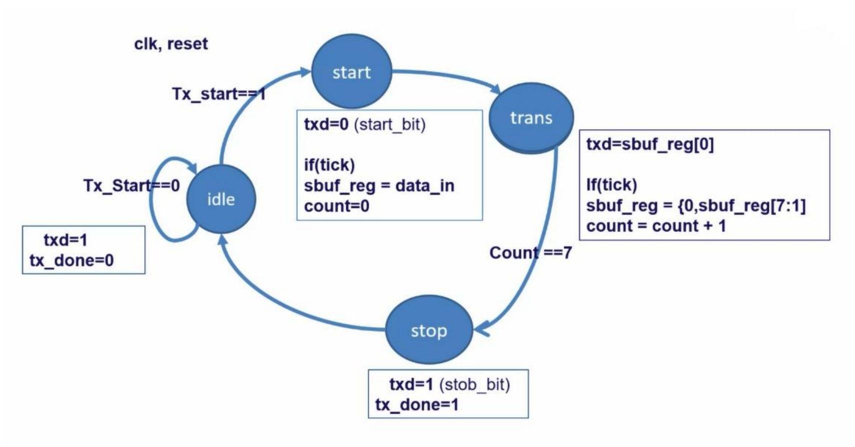
Data Transmission:

1. **Parallel Data Input:**
    - The transmitting UART receives **8-bit parallel data** from a source (e.g., a microcontroller or processor).
    - This data is loaded into the **Serial Buffer Register (SBUF)** in one tick pulse.
2. **Data Conversion:**
    - The parallel data is shifted out **bit by bit** in synchronization with the **baud rate generator**.
    - The baud rate generator provides clock pulses ("ticks") to control data shifting and transmission speed.
3. **Serial Transmission:**
    - The bits are sent serially through the **TxD (Transmit Data) line**.
    - The complete 8-bit data transmission requires **8 tick pulses**.
    - The data is transmitted with start and stop bits (not shown in the diagram but essential in UART communication).

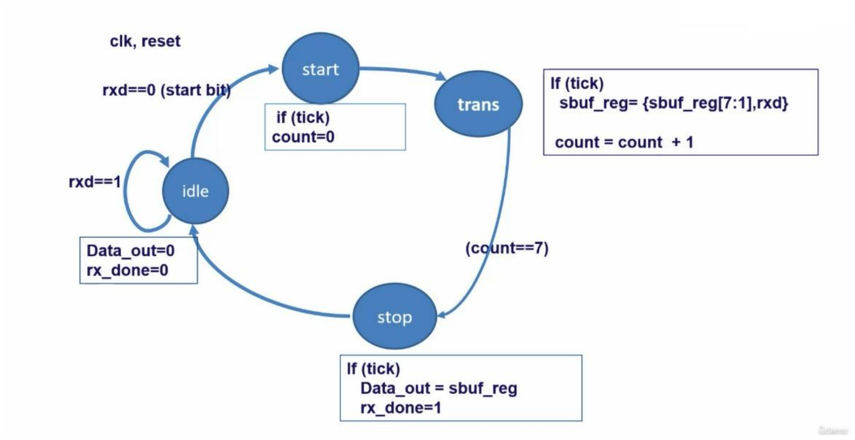
Data Reception:

1. **Serial Data Reception:**
    - The receiving UART captures the incoming serial data stream from the **RxD (Receive Data) line**.
    - The data is shifted into the **SBUF Register** bit by bit over **8 tick pulses**.
2. **Parallel Conversion:**
    - Once all 8 bits are received, the data is converted from serial to parallel format.
    - The complete **8-bit parallel data** is made available for processing in **1 tick pulse**.
3. **Data Output:**
    - The received parallel data is then forwarded to the connected system for further processing.

Transmission FSM:



Reception FSM:



NOTE:

To understand the concept of UART practically, it is important to note that even though the **transmitter (TX module), receiver (RX module), and baud rate generator** exist within a single UART chip, actual data transfer occurs between two separate UARTs. The **TX module of one UART** transmits data to the **RX module of another UART**.

For **simulation (verification)**, it is not possible to test both transmission and reception using a single top module. A **top module is required only for synthesis** (i.e., hardware implementation).

If you want to **simulate both transmission and reception**, you need to use **two UARTs**—one UART's **TX module** will send data to the second UART's **RX module**. This introduces several **synchronization challenges** that need to be resolved. Such testing is typically performed at the **block level verification** stage rather than at the **system level**.