ZDC Semilab Summer Workshop

Project Task: UART Transmitter Module

Presented by: ZDC Team

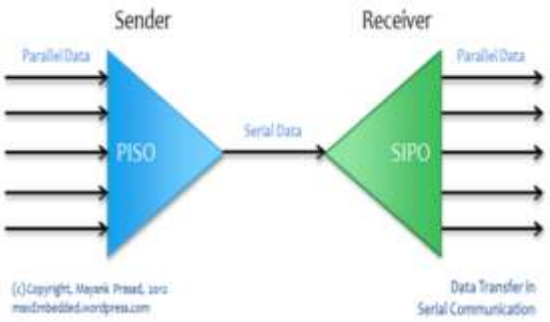
Date: 2/8/2024

Welcome to the ZDC Semilab Summer Workshop! This project task involves the design and implementation of a UART transmitter. You will explore the fundamentals of serial communication and gain hands-on experience in digital IC design.

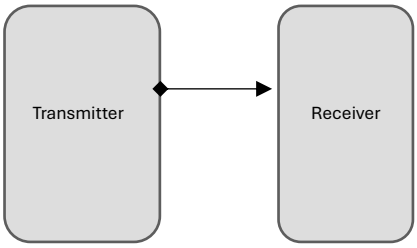
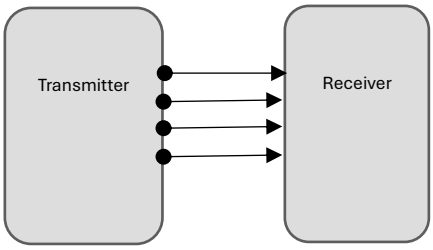
Project Overview:

- **Objective:** To design a functional UART transmitter module.
- **Learning Outcomes:** Understanding of serial communication, hardware design, and implementation.

- **UART** stands for Universal Asynchronous Receiver / Transmitter
- **Definition**: UART is a hardware communication protocol used for asynchronous **serial** communication between devices.
- **Asynchronous Communication**: No common clock signal; data is synchronized using start and stop bits.



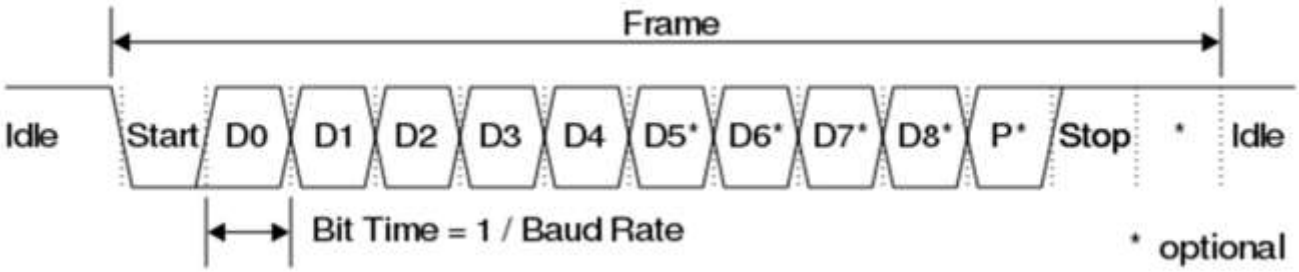
What is the difference between serial and parallel communication ?

Comparison	Serial	Parallel
Bit / Cycle	Only one Bit / Cycle	Multiple Bits / Cycle
Speed	Typically, slower	Typically , faster
Hardware Complexity	Low	High
Disadvantages	Typically, Lower throughput	- Affected by clock Skew - Wires length is limited due to cross talk
		

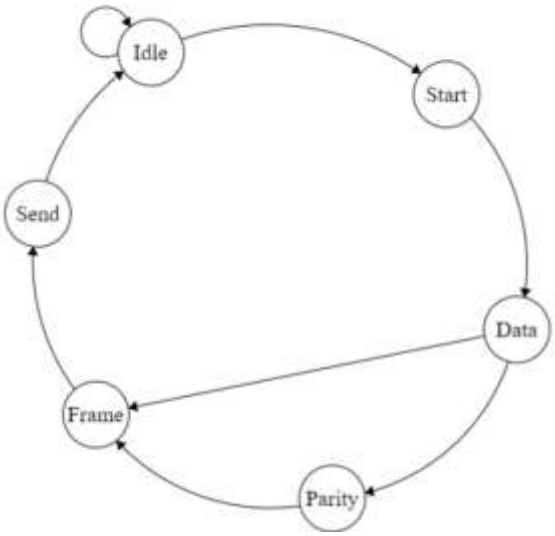
Although serial protocols use a single channel, they may be preferred in certain applications due to their simplicity and the potential to achieve higher clock frequencies compared to parallel protocols

Noted that : the choice depends on specific application requirements

Data Format: Typically, 8 data bits, 1 start bit, (1 or 2) stop bits, and optional parity bit.

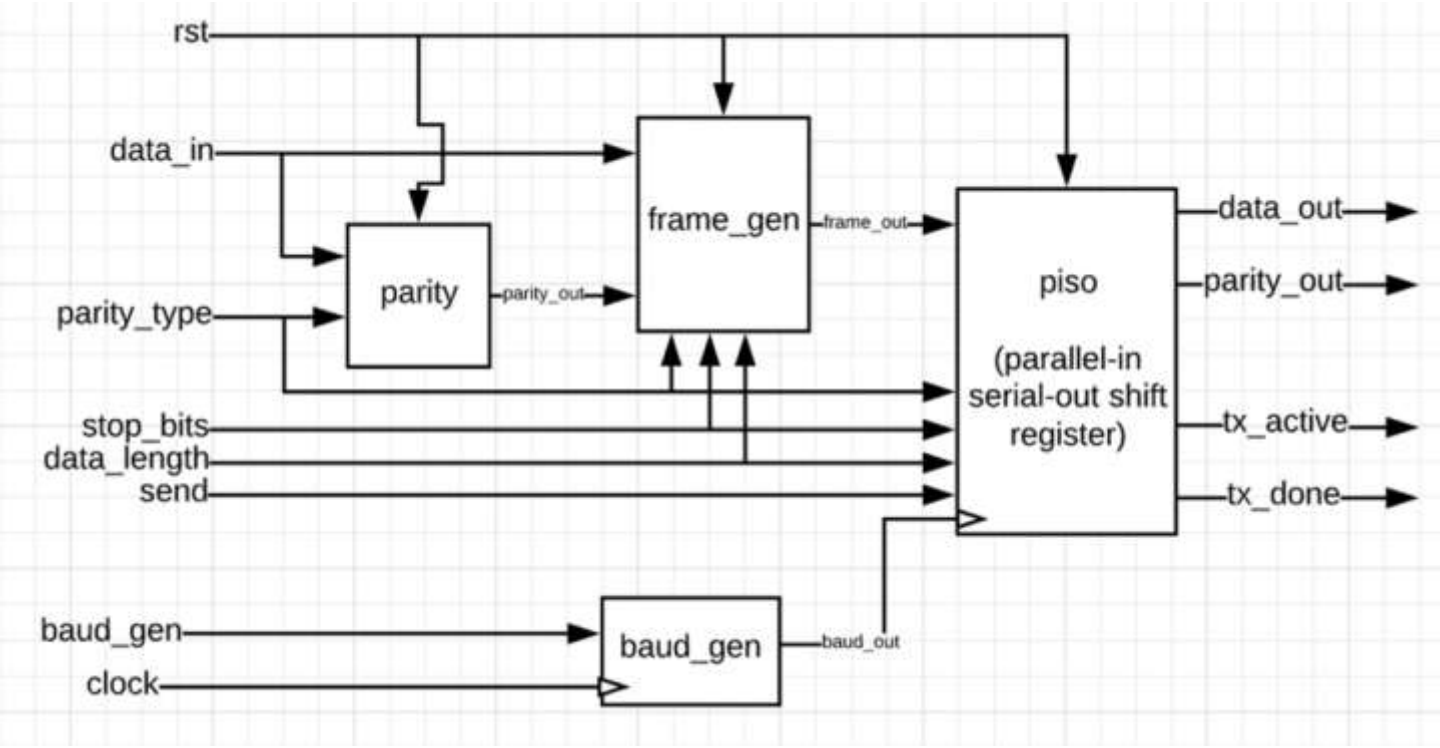


Transmitter Operation :



- **When idle, the transmitter send a stream of ones.**
- **The start bit is 0.**
- **The data length can either be 7 or 8.**
- **The parity bit is optional and can either be odd or even.**
- **• The stop bit/s can be 1 or 2.**

Basic Transmitter Architecture :



Component :

1) Parity Detector

- A parity bit is a basic method to check whether the received data is correct
- It can be either even parity or odd parity.
- With even parity, the bit is only 0 when the number of ones is even.
- With odd parity, the bit is only 0 when thenumber of ones is odd.

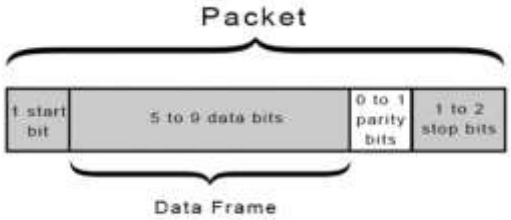
Parity type signal:

if 00 no parity , (parity will not exist in fram) .
if 01 odd parity.
if 10 even parity.
if 11 use the output "**parity _out**" as an **odd parity** bit and no parity in the serial frame(like 00).

2) Frame Generator

- receives **data_in** may be 7 or 8 bits in **parallel**
- Receives parity bit "Parity_out" , may not accept it in case of parity_type "00" , "11"
- Receives **stop_bits** :
  if **low :** One stop bit in the frame
  If **High:** Two stop bits in the frame
- Receives **data_length**

if **low** : 7 bits in the frame.
If **High:** 8 bits in the frame.



3) **Parallel in Serial out "PISO"**
   - Receives the full frame in **parallel** and convert it to **serial** respectively at **data_out**
   - The frame is transmitted when "**Send"** is asserted
   - As long as the frame is still transmitted , signal "**tx_active**" is asserted
   - Once the frame (i.e. start , data , parity , stop) is comleted regardless of frame length, Signal "**tx_done**" is asserted
   - Signal "**parity_out**" is asserted only when signal "**parity_type**" = 11 as previously mentioned

4) **Baud Generator**
   - Baud rate is the number of serial bits per second(**Bits/Sec**) that can be transmitted/received over a certain communication line
   - It receives **baud_gen :**
     if 00 , 2400 baud (**Bits/Sec**)
     If 01 , 4800 baud (**Bits/Sec**)
     If 10 , 9600 baud (**Bits/Sec**)
     If 11 , 19.2K baud (**Bits/Sec**)
   - Assume the system **clock** frequency, for frequency division, is 50MHz.
   - **Baud_out is** directly connected to **PISO** as a clock input

Notes :

1) You can use any microarchitecture as long as you will verify the specifications mentioned.
2) In the top file , Use **only** the port names specified in the given architecture and do not add any other signals .
   You can use this for the top file :

```
module uart_tx(
    //DO NOT EDIT any part of this port declaration
    input           clock, rst, send,
    input [1:0]     buad_rate,
    input [7:0] data_in,
    input [1:0] parity_type,  //refer to the block comment above.
    input           stop_bits,          //low when using 1 stop bit, high when using two stop bits
    input           data_length,    //low when using 7 data bits, high when using 8.
    output reg      data_out,           //Serial data_out
    output reg      p_parity_out,   //parallel odd parity output, low when using the frame parity.
    output reg      tx_active,          //high when Tx is transmitting, low when idle.
    output reg      tx_done             //high when transmission is done, low when not.
);
```

3) You must insert both RTl "code , schematic" and the simulation "Testbench".

Don't be shy to ask for help.
