# **UART Transmitter Design in Verilog**

This repository contains a Verilog implementation of a **Universal Asynchronous Receiver-Transmitter (UART) Transmitter** module, along with its corresponding testbench. The UART Transmitter module sends data byte-by-byte over a serial connection. The module supports basic functionality such as:

* Idle state when no transmission is occurring.
* Start bit signaling to begin transmission.
* Data bit transmission in sequence (LSB first).
* Stop bit signaling to mark the end of the transmission.
* Transmission done signaling once the data is fully transmitted.

## **Features**

* Configurable clock period for controlling baud rate via CLOCKS\_PER\_BIT parameter.
* Supports a basic transmission protocol with 8-bit data.
* Includes a testbench (UART\_TB) for simulation and verification.
* The module transmits data byte (8-bit) in the format: start bit, 8 data bits, stop bit.

## **Modules**

### **UART**\_**TX**

This is the main UART transmitter module. It contains the following ports:

#### **Inputs**:

* reset: Active-low reset signal to initialize the transmitter.
* clock: Clock signal.
* data\_valid: Indicates when valid data is available for transmission.
* data\_in: 8-bit data to be transmitted.

#### **Outputs**:

* transmitting: High when the transmitter is active.
* serial\_out: The serial output, which carries the data being transmitted.
* transmission\_done: High once the transmission is complete.

### **UART\_TB**

This is the testbench for the UART\_TX module. It simulates the behavior of the UART transmitter, applies input signals, and checks the output to verify correct operation.

#### **Testbench Operation:**

* It initializes the system by applying the reset signal.
* Once the reset is deasserted, it sets data\_valid to 1 and provides a sample byte (8'h3F) for transmission.
* The testbench waits for the transmission to complete and checks if the transmission\_done signal is asserted, indicating a successful transmission.

## **Simulation**

To simulate the design, you can use the following tools:

* **ModelSim** for compiling and running the simulation.

### **Running the Testbench**

1. Compile the Verilog files and the testbench.
2. Run the simulation to observe the behavior of the UART\_TX module.
3. Check the output in the waveform viewer or the console to verify correct transmission.

**The expected output in the console is:**

Test Passed - Byte 0x3F Transmitted Successfully

**If there's an issue with transmission, the output will be:**

Test Failed - Transmission Error

## **Usage**

1. Clone the repository to your local machine:

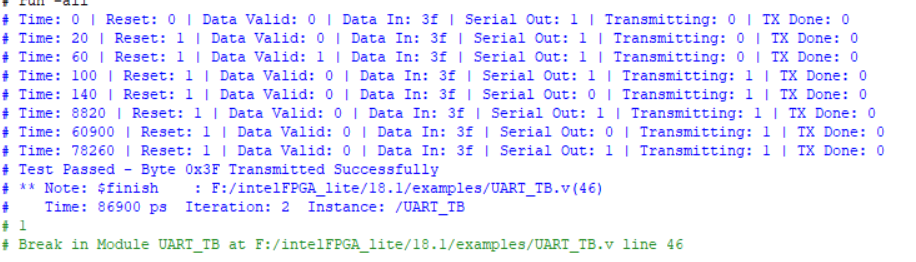
git clone https://github.com/vishwa-patwari/verilog\_projects.git

1. Open the Verilog files and modify the parameters to suit your requirements (e.g., **CLOCKS\_PER\_BIT** for different baud rates).
2. Run the testbench to verify the functionality.

## **Tools Required**

* A Verilog simulator like **ModelSim**, **Vivado**, **Quartus Prime.**
* Optionally, a waveform viewer to visualize signal transitions during simulation.

## **Simulation output**



## **Simulation waveforms**

## **C:\Users\sunil\OneDrive\Pictures\Screenshots\uartwave.png**