Universal Asynchronous Receiver/Transmitter design in VHDL

Table of Contents

Table of Figures

Introduction

1.1 Overview

UART (Universal Asynchronous Receiver/Transmitter) is one of the most widely used communication protocols in embedded systems and digital design. It enables full-duplex serial communication between digital devices. Unlike synchronous protocols, UART does not require a shared clock between transmitter and receiver; instead, it relies on precise timing and baud rates.

This project implements a functional UART communication system using VHDL. The design includes three primary components:

- **UART_TX**: Responsible for serializing and transmitting parallel data.
- **UART_RX**: Responsible for deserializing and receiving serial data.
- **UART_TOP**: Integrates the transmitter and receiver, acting as the top-level module.

1.2 Features

- Programmable baud rates: 9600, 19200, 38400, 57600, 115200 bps.
- Programmable bus width at run time.
- width-N-1 format: 1 start bit, width bits data, no parity bit, 1 stop bit.

Core Design

2.1 Block diagram

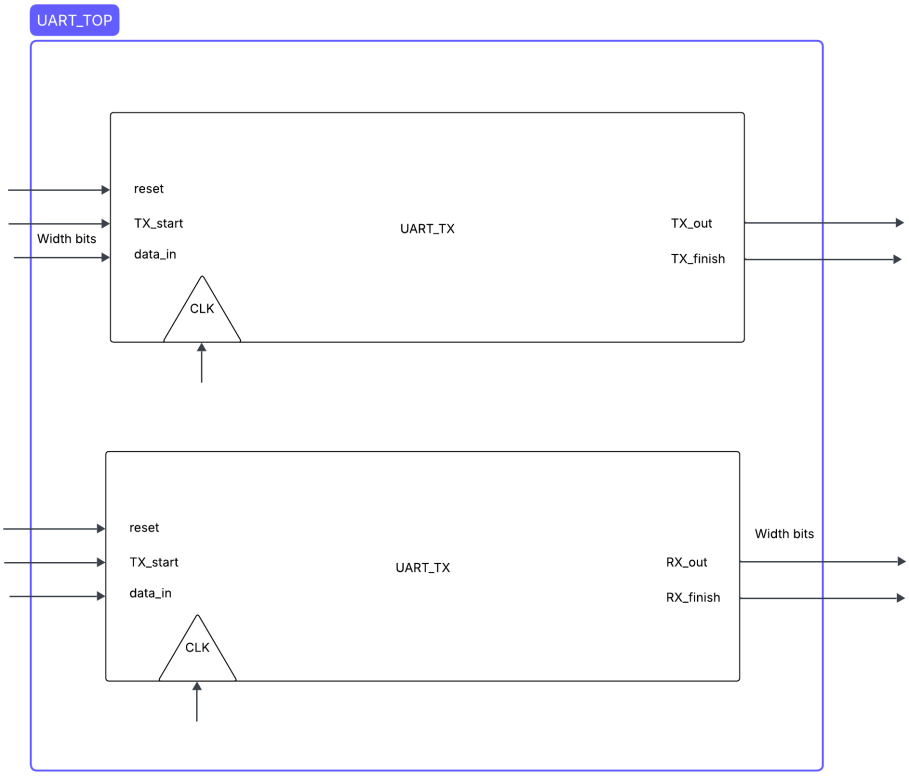

Figure 1: Block diagram

2.2 Signal description

2.2.1 transmitter

Name	Type	Description
Clk	IN	System clock
reset	IN	Asynchronous reset
TX_start	IN	When high system starts transmitting
TX_data_in	IN	Input parallel data of width (width) generic type
TX_out	OUT	Output serial data
TX_busy	OUT	High when transmitting
TX_finish	OUT	High when the TX is done

2.2.2 Receiver

Name	Type	Description
Clk	IN	System clock
reset	IN	Asynchronous reset
RX_data_in	IN	Input serial data

RX_data_out	OUT	Output parallel data
RX_finish	OUT	High when reviver is done

2.3 TX entity

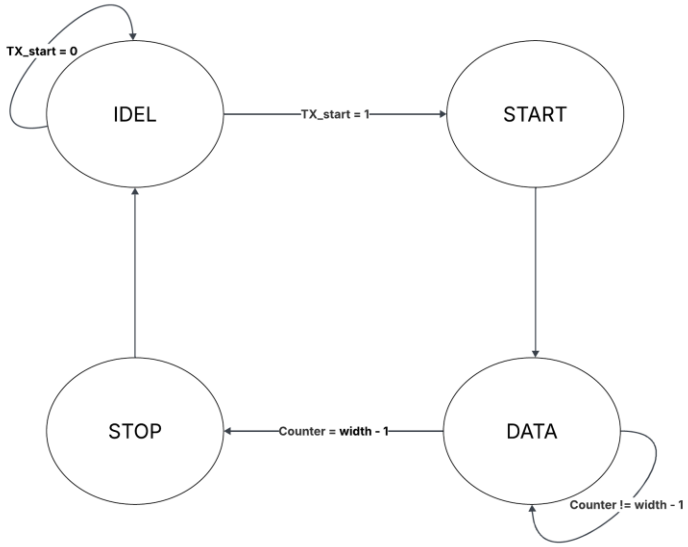

Figure 2: TX finite state machine

This module works on baud clock where it's equal to $\frac{clk\ freq}{baud\ rate}$ for example 50MHZ clock and baud rate of 9600 then the baud clock is 5208 which means we send bit every 5208 clocks.

The FSM starts in the idle state waiting for the start signal to be high while setting signal finish, busy, counter to zero and TX_out to 1, once TX_start triggers it goes to START state where TX_out sited to 0 as a start bit and then wait a baud clock (baud clk is)then go to the DATA state where we loop width times and send a bit of the data input every baud clock.

Then when the counter reaches width – 1 it goes to the Stop state and set TX to 1 and go back to IDEL state after a baud clock.

2.4 RX entity

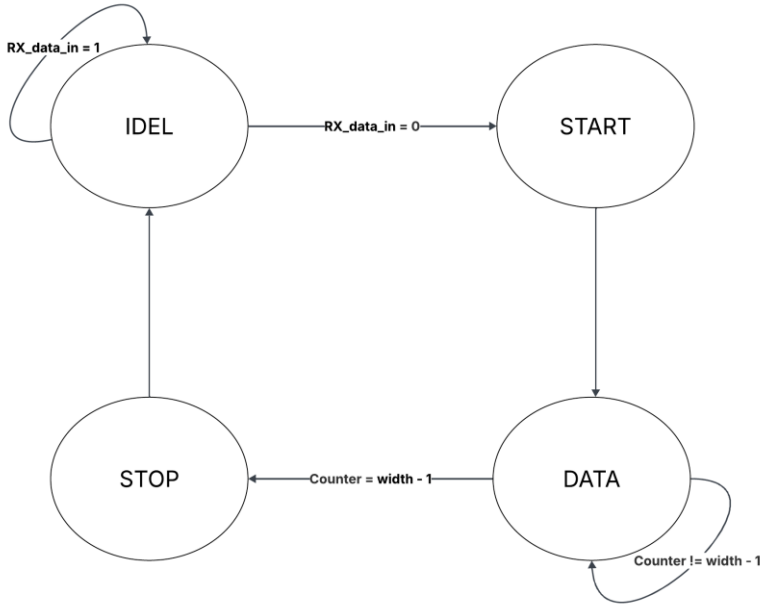

Figure 3: RX finite state machine

RX_data_in is connected to the TX_out. So RX is waiting in the IDEL state for the TX_out to toggle to 0 which is the starting point (start bit) then go to the START state and wait there for half baud clock then check the TX_out if its 0 then it goes to the DATA state and sample the data but if not this means that there is an error and the system resets back to IDEL state.

The reason for sampling in the middle of the baud clock is to avoid timing issues and make sure that the data is stable.

Results

3.1 Simulation

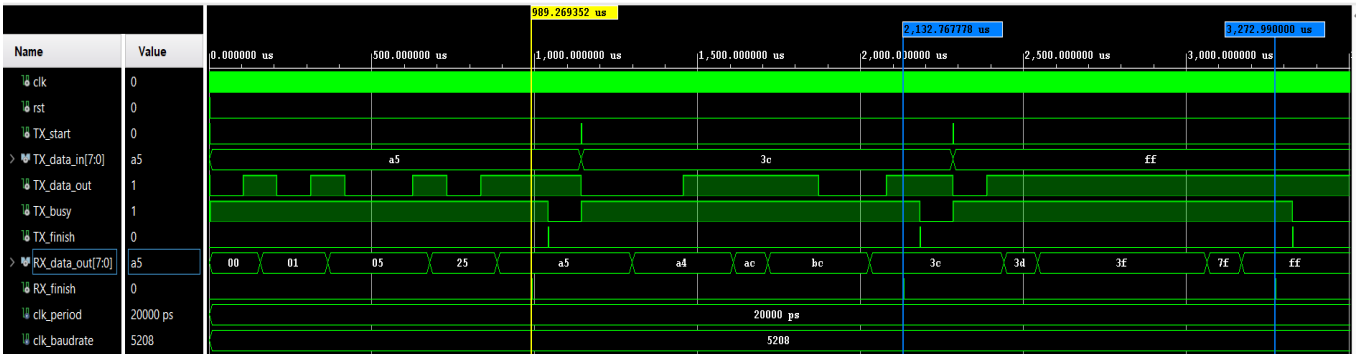

Figure 4: Wave form

3.2 Synthesis

Targeted FPGA Board **Spartan-7** version **xc7s6cpga196-2.**

3.2.1 System synthesis

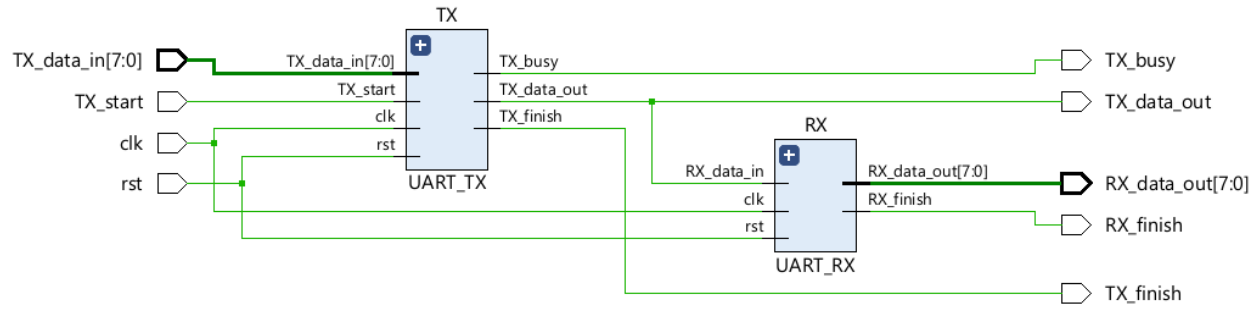

Figure 5: System synthesis

3.2.2 TX synthesis

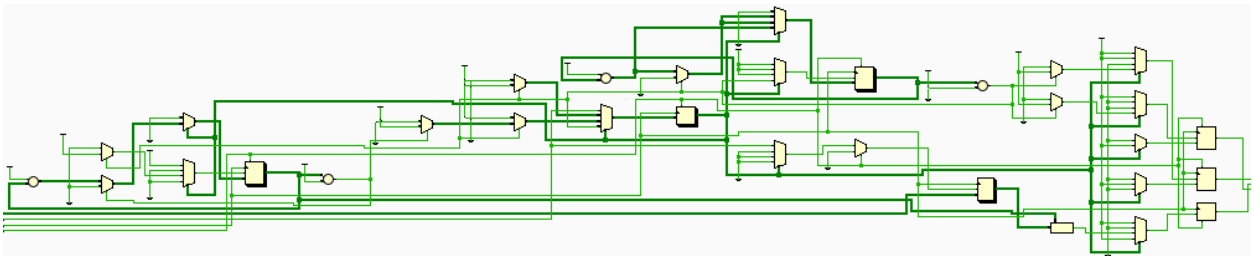

Figure 6: TX synthesis

3.2.3 RX synthesis

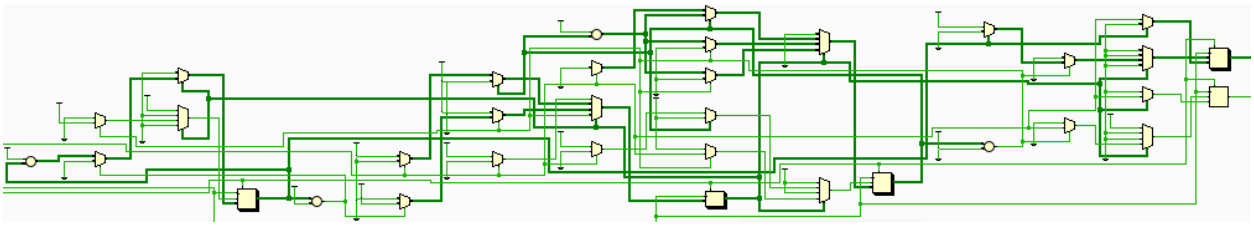

Figure 7: RX synthesis

References

1-“Basics of UART Communication,” *Electronics Hub*, Jun. 17, 2017. https://www.electronicshub.org/basics-uart-communication/