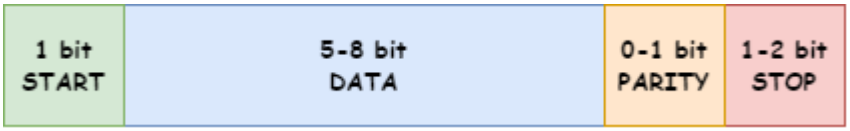
PRELIMINARY

UART (Universal asynchronous receiver-transmitter) is a data communication protocol for asynchronous serial communication. UART has two interfaces which are transmitter (Tx) and receiver (Rx) lines used for transmitting and receiving serial data. The timing issue is handled by a start bit and stop bit(s). There is also an optional parity bit to determine whether the correct data is received **[1]**.

UART baud rates are usually slower than the system clock speeds. Typical baud rates are 9600 and 115200 Hz. In order to sample the data bits properly, Rx driver uses oversampling, generally 8x or 16x times of the baud rate. To increase the performance, most of the UART interfaces include a Tx (transmit) and a Rx (receive) FIFO buffers together with control and status registers (CSRs) **[2]**.

UART frame is consists of:



- 1 Start Bit (logic low)
- 5-8 Data Bits
- Parity Bit (optional)
- 1-2 Stop Bit(s) (logic high)

UART Transmitter drives a high voltage level when it's not transmitting data. To start the transfer of data, the transmitter changes the voltage level from high to low for one period (T). The data is sent with the least significant bit first. After the data bits are transmitted, the transmitter drives a high voltage for one to two bit(s) (T-2T) duration (stop bits) **[3]**.

UART Receiver observes the Rx line in idle mode whether there is a high to low voltage level transition. When the receiver detects low voltage level for half of the period T/2 duration, it starts reading data. In order to sample the data properly, receiver uses 8x or 16x faster frequency than the baud rate. Receiver samples the data bits 8 or 16 times and decide the value of bits (0 or 1) by calculating which logic-level is more sampled. When the receiver detects stop bit(s) for one period T duration, it returns the idle mode and waits for the new transmission requests (start bit).

UART Baud Rate is calculated by

$$f_{baud} = \frac{f_{clk}}{1 + baudDiv}$$

where f_{clk} is the system frequency, f_{baud} is the baud rate and baudDiv is the divider to achieve desired baud rate. To obtain the desired baud rate and its **8x** or **16x** counterpart a **frequency divider** can be used. First, oversampled baud frequency should be calculated and then real baud frequency can be obtain from it to make them synchronized.

PROJECT DESCRIPTION & REQUIREMENTS

NOTE: You may use extra references (videos, websites, research papers, books etc.), just be sure to list them in your report.

It is requested to design and implement an **UART (Universal asynchronous receiver-transmitter) Driver** which receives and transmits asynchronous serial data.

Frame structure should include 1 start bit, 8 data bits, and 1 stop bit. Baud rate should be configurable and support for 9600 and 115200 Hz baud frequencies.

RTL DESIGN

1. There should be 4 different modules for the UART Driver design. Transmitter (**uart_tx)**, Receiver **(uart_rx)**, Baud Rate Generator **(baud_gen)** and the Top Module **(uart_top)** .
2. **UART Transmitter** should be designed with a **Finite State Machine**. Parallel 8-bit data should be taken as an input and transmitted serially in accordance with the frame structure. Example for the transmission states: IDLE, START, DATA, STOP.
3. **UART Receiver** should also be designed with a **Finite State Machine**. Serial 1-bit data should be taken as an input and detected properly. Receiver should use x8 baud frequency. Receiver should detect start bit for T/2 period. The value of data bits (0 or 1) should be decided by calculating which logic-level is more sampled. After receiving is completed, parallel 8-bit data should be given as an output. Example for the receiving states: IDLE, START, DATA, STOP.
4. There should be two control signals enabling Tx and Rx lines **Tx_en** and **Rx_en**. There should be also an reset **rst** signal.
5. There should be 3 output status signals **start**, **busy** and **done** for both transmitter and receiver.
6. **Baud Rate Generator** should be designed with clock dividers. Baud rate should be configurable and should achieve 9600 and 115200 Hz frequencies.
7. **Top Module** should have the instances of **Transmitter**, **Receiver** and **Baud Rate Generator**. Make the proper connections between **Baud Rate Generator** with Transmitter and Receiver.

SIMULATION

1. All of the modules **Transmitter**, **Receiver**, **Baud Rate Generator** and **Top Module** should have testbenches and proper simulations.
2. **Transmitter** should be tested with 4 random 8-bit data. Show that Transmitter outputs the 8-bit data within the UART frame correctly.

3. **Receiver** should be tested with 4 random 8-bit data frames. In order to test the asynchronous timing issues, give the serial input data with **different phases**. Show that receiver detects the 8-bit data correctly.
4. **Baud Rate Generator** should be tested with 9600 and 115200 Hz frequencies. Show that baud rate is configurable.
5. **Top Module** should be tested with both transmission and receiving operations. The same test scenarios in the Transmitter and Receiver testbenches can be applied.

IMPLEMENTATION & REPORT

1. Explain UART protocol: how to operate and how to achieve asynchronous transmission.
2. Draw **state diagrams** of Transmitter and Receiver and add it to your report.
3. Draw a **block diagram** of the general design (top module) including signal names and connections.
4. Obtain timing and resource information from Vivado post-implementation reports. Provide the number of cells used (LUTs and FFs) and the maximum clock frequency of the design.
5. Obtain post-implementation timing simulation waveforms of the testbenches and add them to your report.

REFERENCES

[1] "KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide." Available: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf

[2] "Universal asynchronous receiver-transmitter," *Wikipedia*. https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

[3] E. Pena and M. G. Legaspi, "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices." https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

NOTE: This WILL NOT be a group project. Prepare your projects&reports individually.

Contact me, if you have any questions about the project.

SERDAR DURAN