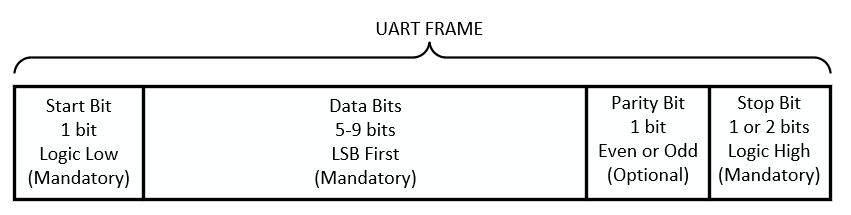
Universal Asynchronous Receiver Transmitter (UART) is a common method of serial communication, defined by a unique frame sequence. Certain characteristics of UART are universal to the protocol, while others vary based on application.



The UART protocol works by requiring that the data lines idle in a logic high state, meaning that if no data is being transmitted, then the *uart\_tx* will be driven high. The start bit is always a logic low, indicating that a new frame is being received. The data is then received serially, LSB first. A parity bit is added to the end for error detection. This bit is optional and can either be for even parity or odd parity. One or two stop bits are appended to the end to indicate the end of the frame and are always logic high. After a frame is ended, the line remains idling at logic high, so that the next time a logic low appears on the line, a new frame is arriving.

Data is sent serially LSB first, so if you want to send the byte 0x7A (01111010), you would send it in the following order: 0->1->0->1->1->1->1->0.

UART is considered asynchronous because the speed at which it is transmitted has no relationship to the receiving clock. This means that in order to accurately receive a UART frame, the receiver needs to have a clock fast enough to detect the change on the *uart\_rx* line. This is not a problem as most digital systems have clocks in the MHz range, while UART is constrained to the low kHz range.

For your design to accurately transmit data at a given BAUD rate, you will need to implement a counter which will let you know when you have transmitted one bit. The equation to determine the number of clock cycles per UART bit and the number of bits in your counter are as follows:

Design an FSM-based UART transmitter module, written in Verilog HDL, whose input is a parametric data word accompanied with a *data­\_vld* signal, and whose output is a serial uart­\_tx line along with a data\_rdy signal. Your design must have the following parameters; width of the data word, the desired BAUD rate, and the transmitter clock frequency.

This assignment will build on the previous two assignments to completely design a functionally useful UART RX/TX module capable of operating at different BAUD rates with the use of a FIFO buffer.

Your UART TX module will use the industry standard *ready/valid* handshaking protocol to communicate between digital systems *within* your design. In this case, your module will output a *data\_rdy* signal which will be used to indicate that it is **READY**to accept and transmit a new data word. In terms of the FSM you can think of it as a flag which will be set high when in the IDLE state and set low otherwise. Additionally, your module will take a *data\_vld* input signal which will tell your module that the data on the input data line is valid.

It is only when **BOTH** the *data\_vld* and *data\_rdy* flags are high **AT THE SAME TIME** that a valid transaction can occur. In the case of your design, when both ready and valid flags are high and you are in the IDLE state, you should register the input data word and transition to the START state. If done correctly, there should only be one clock cycle in which both the *data\_vld* and *data\_rdy* flags are high at the same time (THIS IS IMPORTANT).

In the case of this assignment, you will take the inverted empty flag from the FIFO as the *data\_vld* flag, meaning that if the FIFO is not empty, there is valid data on the line. You should pass your *data\_rdy* flag into the FIFO as a read\_enable signal. This will result in the FIFO word being read out whenever your UART TX is ready to send a new byte.

You may design either a Mealy or Moore FSM, but you will need to indicate which type of design you chose in your comments. The clock signal and reset signal have not been drawn.

