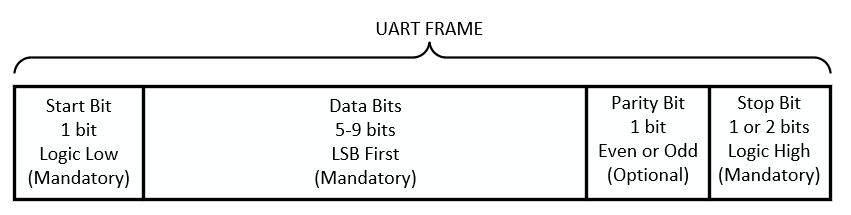
Universal Asynchronous Receiver Transmitter (UART) is a common method of serial communication, defined by a unique frame sequence. Certain characteristics of UART are universal to the protocol, while others vary based on application.



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.

In order for your design to accurately receive asynchronous data, you will need to implement a counter which will let you know when you have received ONE HALF of a bit. It is at this point that you will sample the data line. Refer to the drawing below.



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 receiver module, written in Verilog HDL, which accepts a parametric data word, an even parity bit, and one stop bit. Your module should take in a parameter to indicate the length of the data word, the expected BAUD rate (in bits/second), and the receiver clock frequency (in MHz). Your module should take in a serial input, *uart\_rx*, and output the word along with a *data­­­\_vld* flag and a *par\_err* flag. Although you will be receiving data LSB first, your word must be output in the correct format (MSB to the left)

The size of your counter MUST be calculated based on the input parameters. It CANNOT be hardcoded.

After an entire frame has been received, your design should output the received word, minus the start bit, parity bit, and stop bit, and assert the *data\_vld* flag if there was no error detected, otherwise assert the *par\_err* flag. These flags should only be asserted for one clock cycle, on the transition from STOP state to IDLE state.

Your FSM should transition out of the IDLE state as soon as a logic low is detected on the *uart­\_rx* line. 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.

The output of your UART receiver will be connected to the FIFO you designed in your last assignment. You will need to modify your old FIFO design very slightly for this assignment.

