perplexity

Please generate a 8-bit UART design in Verilog defined as:

module Uart8 #(
parameter CLOCK_RATE = 100000000,
parameter BAUD_RATE = 9600,
parameter TURBO_FRAMES = 0
)(
input wire clk,

input wire rxEn,
input wire rx,
output wire rxBusy,
output wire rxDone,
output wire rxErr,
output wire [7:0] out,

input wire txEn,
input wire txStart,
input wire [7:0] in,
output wire txBusy,
output wire txDone,
output wire tx
)
//Insert code here
endmodule

```
module Uart8 #(
    parameter CLOCK_RATE = 100000000,
    parameter BAUD_RATE = 9600,
    parameter TURBO_FRAMES = 0
)(
    input wire clk,

    input wire rxEn,
    input wire rx,
    output wire rxBusy,
    output wire rxDone,
    output wire rxErr,
    output wire [7:0] out,

    input wire txEn,
    input wire txStart,
    input wire [7:0] in,
    output wire txBusy,
    output wire txDone,
    output wire tx
);

    // Calculate the number of clock cycles per bit
    localparam integer CLOCKS_PER_BIT = CLOCK_RATE / BAUD_RATE;

    // RX Section
    reg [15:0] rx_data;
    reg [3:0] rx_bit_count;
    reg rx_busy_int;
    reg rx_done_int;
    reg rx_err_int;
    reg [7:0] rx_out_int;
    reg rx_tick;

    assign rxBusy = rx_busy_int;
    assign rxDone = rx_done_int;
    assign rxErr = rx_err_int;
    assign out = rx_out_int;

    always @(posedge clk) begin
        if (rxEn) begin
            if (rx_busy_int) begin
                rx_tick <= rx_tick + 1;
                if (rx_tick == CLOCKS_PER_BIT - 1) begin
                    rx_tick <= 0;
                    if (rx_bit_count < 10) begin
                        rx_data[rx_bit_count] <= rx;
                        rx_bit_count <= rx_bit_count + 1;
                    end
                    if (rx_bit_count == 10) begin
                        rx_busy_int <= 0;
                        rx_done_int <= 1;
                        rx_err_int <= (rx_data[0] != 0) || (rx_data[9] != 1);
                        rx_out_int <= {rx_data[8], rx_data[7], rx_data[6], rx_data[5], r
                    end
                end
            end else begin
                if (~rx) begin
                    rx_busy_int <= 1;
                    rx_done_int <= 0;
                    rx_err_int <= 0;
                    rx_bit_count <= 0;
                    rx_tick <= 0;
                end else begin
                    rx_done_int <= 0;
                end
            end
        end else begin
            rx_busy_int <= 0;
            rx_done_int <= 0;
            rx_err_int <= 0;
        end
    end

    // TX Section
    reg [15:0] tx_data;
    reg [3:0] tx_bit_count;
    reg tx_busy_int;
    reg tx_done_int;
    reg tx_out_int;
    reg tx_tick;

    assign txBusy = tx_busy_int;
    assign txDone = tx_done_int;
    assign tx = tx_out_int;

    always @(posedge clk) begin
        if (txEn) begin
            if (tx_busy_int) begin
                tx_tick <= tx_tick + 1;
                if (tx_tick == CLOCKS_PER_BIT - 1) begin
                    tx_tick <= 0;
                    if (tx_bit_count < 10) begin
                        tx_out_int <= tx_data[tx_bit_count];
                        tx_bit_count <= tx_bit_count + 1;
                    end
                    if (tx_bit_count == 10) begin
                        tx_busy_int <= 0;
                        tx_done_int <= 1;
                        tx_out_int <= 1;
                    end
                end
            end else begin
                if (txStart) begin
                    tx_busy_int <= 1;
                    tx_done_int <= 0;
                    tx_bit_count <= 0;
                    tx_tick <= 0;
                    tx_data <= {1'b1, in, 1'b0};
                    tx_out_int <= 0;
                end else begin
                    tx_done_int <= 0;
                end
            end
        end else begin
            tx_busy_int <= 0;
            tx_done_int <= 0;
        end
    end

endmodule
```