

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
```

Here is an 8-bit UART design in Verilog based on the provided module definition and parameters:

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

    // RX Interface
    input wire rxEn,
    input wire rx,
    output reg rxBusy,
    output reg rxDone,
    output reg rxErr,
```

```

output reg [7:0] out,

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

// Internal Parameters and Variables
localparam integer BAUD_DIV = CLOCK_RATE / BAUD_RATE;
localparam integer STOP_BITS = (TURBO_FRAMES == 1) ? 1 : 2;

// RX State Machine
reg [3:0] rxState;
reg [7:0] rxShiftReg;
reg [15:0] rxBaudCounter;
reg [3:0] rxBitCounter;

always @(posedge clk) begin
    if (!rxEn) begin
        rxState <= 0;
        rxBusy <= 0;
        rxDone <= 0;
        rxErr <= 0;
    end else begin
        case (rxState)
            0: begin // Idle, wait for start bit
                if (!rx) begin // Start bit detected
                    rxState <= 1;
                    rxBusy <= 1;
                    rxBaudCounter <= BAUD_DIV / 2; // Center of start bit
                end
            end
            1: begin // Start bit sampling
                if (rxBaudCounter == 0) begin
                    if (rx != 0) begin // Error if not still low
                        rxState <= 0;
                        rxBusy <= 0;
                        rxErr <= 1;
                    end else begin
                        rxState <= 2;
                        rxBitCounter <= 0;
                    end
                end else begin
                    rxBaudCounter <= rxBaudCounter - 1;
                end
            end
            2: begin // Data bits sampling
                if (rxBaudCounter == 0) begin
                    rxShiftReg[rxBitCounter] <= rx; // Shift in data bit
                    if (rxBitCounter == 7) begin // Last data bit received
                        rxState <= 3; // Move to stop bit(s)
                    end else begin

```

```

        rxBitCounter <= rxBitCounter + 1;
    end
    rxBaudCounter <= BAUD_DIV - 1;
end else begin
    rxBaudCounter <= rxBaudCounter - 1;
end
end
3: begin // Stop bit(s) sampling
    if (rxBaudCounter == 0) begin
        if (rx != 1) begin // Error if not high during stop bit(s)
            rxErr <= 1;
        end else if (STOP_BITS == 2 && !rxDone) begin // Second stop bit
            rxDone <= !rxDone; // Toggle to indicate completion of stop bit
        end else begin
            out <= rxShiftReg;
            rxDone <= !rxDone;
            rxBusy<=!rxBusy;//toggle busy state
        end
    end
end

```

**