We are generating a 8-bit UART hierarchically in Verilog. We have generated BaudRateGenerator defined as follows:

```
module BaudRateGenerator #(
parameter CLOCK_RATE = 100000000,
parameter BAUD_RATE = 9600,
parameter RX_OVERSAMPLE_RATE = 16
)(
input wire clk,
output reg rxClk,
output reg txClk
);

reg [31:0] rx_counter;
reg [31:0] tx_counter;
localparam RX_DIVISOR = CLOCK_RATE / (BAUD_RATE * RX_OVERSAMPLE_RATE);
localparam TX_DIVISOR = CLOCK_RATE / BAUD_RATE;

always @(posedge clk) begin
rx_counter <= rx_counter + 1;
if (rx_counter >= RX_DIVISOR) begin
rxClk <= ~rxClk;
rx_counter <= 0;
end
end

always @(posedge clk) begin
tx_counter <= tx_counter + 1;
if (tx_counter >= TX_DIVISOR) begin
txClk <= ~txClk;
tx_counter <= 0;
end
end

endmodule
```

Please include the previous module(s) in your response and use them to hierarchically generate a Receiver defined as:

```
module Uart8Receiver(
input wire clk,
```

input wire en,
input wire in,
output reg busy,
output reg done,
output reg err,
output reg [7:0] out
)
//Insert code here
endmodule

```
module BaudRateGenerator #(
    parameter CLOCK_RATE = 100000000,
    parameter BAUD_RATE = 9600,
    parameter RX_OVERSAMPLE_RATE = 16
)(
    input wire clk,
    output reg rxClk,
    output reg txClk
);

reg [31:0] rx_counter;
reg [31:0] tx_counter;
localparam RX_DIVISOR = CLOCK_RATE / (BAUD_RATE * RX_OVERSAMPLE_RATE);
localparam TX_DIVISOR = CLOCK_RATE / BAUD_RATE;


always @(posedge clk) begin
    rx_counter <= rx_counter + 1;
    if (rx_counter >= RX_DIVISOR) begin
        rxClk <= ~rxClk;
        rx_counter <= 0;
    end
end

always @(posedge clk) begin
    tx_counter <= tx_counter + 1;
    if (tx_counter >= TX_DIVISOR) begin
        txClk <= ~txClk;
        tx_counter <= 0;
    end
end

endmodule

module Uart8Receiver(
    input wire clk,
    input wire en,
    input wire in,
    output reg busy,
    output reg done,
    output reg err,
    output reg [7:0] out
);
```

```
reg [3:0] state;
reg [3:0] bit_counter;
reg [9:0] data_buffer; // Start bit + 8 data bits + stop bit
reg rxClk;

localparam IDLE = 0, START = 1, DATA = 2, STOP = 3, ERROR = 4, DONE = 5;

BaudRateGenerator baud_rate_gen (
    .clk(clk),
    .rxClk(rxClk),
    .txClk() // Not used in receiver
);

always @(posedge rxClk) begin
    if (en) begin
        case (state)
            IDLE: begin
                busy <= 0;
                done <= 0;
                err <= 0;
                if (~in) begin // Start bit is 0
                    state <= START;
                    busy <= 1;
                    bit_counter <= 0;
                    data_buffer <= 0;
                end
            end
            START: begin
                state <= DATA;
                bit_counter <= 0;
            end
            DATA: begin
                data_buffer[bit_counter] <= in;
                if (bit_counter == 7) begin
                    state <= STOP;
                end else begin
                    bit_counter <= bit_counter + 1;
                end
            end
            STOP: begin
                if (in) begin // Stop bit is 1
                    state <= DONE;
                    out <= data_buffer[0+:8];
                    done <= 1;
                    busy <= 0;
                } else begin
                    state <= ERROR;
                    err <= 1;
                    busy <= 0;
                    done <= 0;
                end
            end
            ERROR: begin
                busy <= 0;
                done <= 0;
            end
            DONE: begin
                 state <= IDLE;
            end
            default: state <= IDLE;
        endcase
    end else begin
        state <= IDLE;
        busy <= 0;
        done <= 0;
        err <= 0;
    end
end

endmodule
```