```
`timescale 1ns / 1ps
/**********************************************
**********************************
*
* Module: tx
*
* Author: Eric Christie
* Class: ECEN 220, Section 1, Winter 2021
* Date: 3/23/2021
*
* Description: UART transmitter with 8-bit
messages
*
*
**********************************************
**********************************/
`default_nettype none

module tx(
    input wire logic clk, Reset, Send,
    input wire logic [7:0] Din,
    output logic Sent, Sout);

    //internal wires
```

```
    logic clrTimer, timerDone; //I/O for
Baud Rate Timer
    logic incBit, clrBit; //inputs to
BitCounter
    logic [2:0] bitNum; //Bit counter
output
    logic startBit, dataBit, parityBit;
//timing signals for FSM

    //Baud Rate Timer clk, clrTimer,
timerDone
    BaudRateTimer BRT(clk, clrTimer,
timerDone);

    //BitCounter clk, inc, clr, bitNum
    BitCounter BC(clk, incBit, clrBit,
bitNum);

    //Datapath logic
    always_ff @(posedge clk)
        if(startBit) Sout <= 0;
        else if(dataBit) Sout <=
Din[bitNum];
        else if(parityBit) Sout <= ~^Din;
```

```
        else Sout <= 1;


    //FSM
    typedef enum logic [2:0] {Idle, Start,
Bits, Par, Stop, Ack, ERR='X} StateType;
    StateType ns, cs;

    always_comb begin
        ns = ERR;
        clrTimer = 0;
        clrBit = 0;
        incBit = 0;
        startBit = 0;
        dataBit = 0;
        parityBit = 0;
        Sent = 0;

        if(Reset) ns = Idle;
        else
            case(cs)
                Idle: begin
                    clrTimer = 1;
                    if(!Send) ns = Idle;
```

```
                else ns = Start;
            end

            Start: begin
                startBit = 1;
                if(!timerDone) ns =
Start;
                else begin
                    ns = Bits;
                    clrBit = 1;
                end
            end

            Bits: begin
                dataBit = 1;
                if(!timerDone) ns =
Bits;
                else if(bitNum !=
3'b111) begin
                    ns = Bits;
                    incBit = 1;
                end
                else  ns = Par;
            end
```

```
                Par: begin
                    parityBit = 1;
                    if(!timerDone) ns =
Par;
                    else ns = Stop;
                end

                Stop:
                    if(!timerDone) ns =
Stop;
                    else ns = Ack;

                Ack: begin
                    Sent = 1;
                    if(Send) ns = Ack;
                    else ns = Idle;
                end

            endcase
    end

    always_ff @(posedge clk) cs <= ns;
```

```
endmodule
```