```
`timescale 1ns / 1ps
/**********************************************
**********************************
*
* Module: rx
*
* Author: Eric Christie
* Class: ECEN 220, Section 1, Winter 2021
* Date: 4/6/21
*
* Description: UART Receiver
*
*
**********************************************
**********************************/

`default_nettype none
module rx(
    input wire logic clk, Reset, Sin,
    output logic Receive,
    input wire logic Received,
    output logic [7:0] Dout,
    output logic parityError);
```

```
    logic clr;

    //Baud Timer      inputs          outputs
    logic halfBit, fullBit;
    BaudRateTimer Brt(clk, clr, fullBit,
halfBit);

    //Bit Counter
    logic [3:0] BitCount;
    always_ff @(posedge clk)
        if(clr) BitCount <= 0;
        else if(fullBit) BitCount <=
BitCount + 1;

    //Shift Register
    logic enableShift, parity;
    ShiftRegister Sr(clk, clr,
enableShift, Sin, Dout, parity);

    //Parity Check
    assign parityError = (^Dout ==
parity)?1:0;

    //State Machine
```

```
    typedef enum logic[2:0] {Wait,
FullBitWait, HalfBitSample, Send, ERR='X}
StateType;
    StateType ns, cs;

    always_comb begin
        ns = ERR;
        enableShift = 0;
        clr = 0;
        Receive = 0;

        if(Reset) ns = Wait;
        else
            case (cs)
                Wait: begin
                    clr = 1;
                    if(!Sin) ns =
FullBitWait;
                    else ns = Wait;
                end

                FullBitWait:
                    if(BitCount == 0) ns =
FullBitWait;
```

```
                    else if(halfBit) begin
                        ns = HalfBitSample;
                        enableShift = 1;
                    end
                    else ns = FullBitWait;

                HalfBitSample:
                    if(fullBit &&
(BitCount < 9)) ns = FullBitWait;
                    else if(fullBit &&
(BitCount == 9)) ns = Send;
                    else ns =
HalfBitSample;

                Send: begin
                    Receive = 1;
                    if(Received) ns = Wait;
                    else ns = Send;
                end

            endcase
    end

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

```
endmodule
```