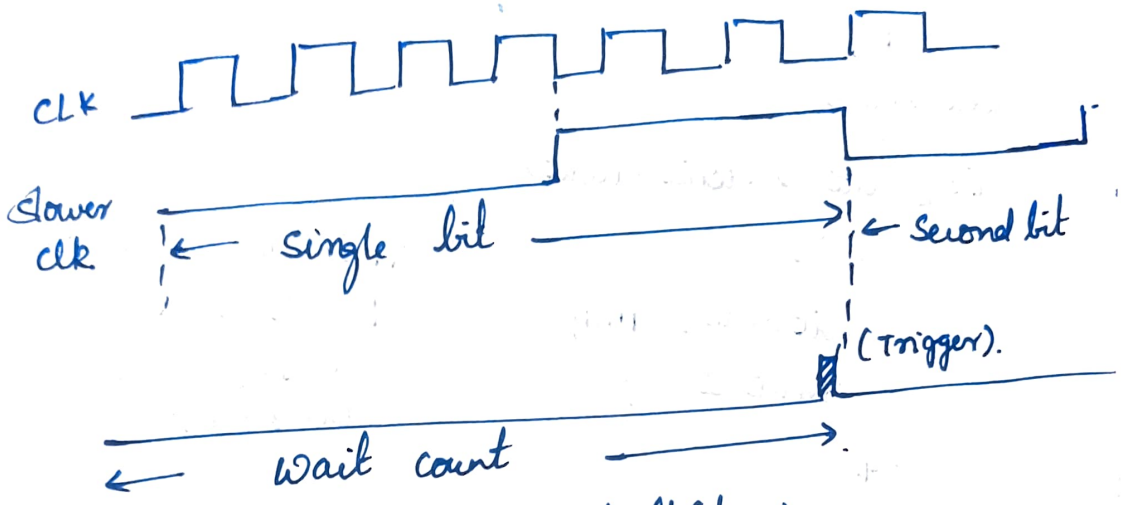
Design (UART)

clk = 1MHz

Baudrate = 9600



Wait count = $\frac{\text{Input clk (freq)}}{\text{Baudrate}}$

Explanation: $\frac{1}{\text{Baudrate}}$ → 1 bit per time unit

1 bit time × i/p clk freq → wait count

```
// Generate trigger.
parameter clk_value = 100_000
parameter baud = 9600
 reg bit done = 0;
integer count = 0;
parameter idle = 0; send = 1, check = 2;
reg [1:0] state = idle;
```

```
always @(posedge clk)
begin
  if (state == idle)
    begin
      count <= 0;
    end
  else begin
  if (count == wait-count)
      begin
        bit done <= 1'b1;
        count <= 0;
      end
    else begin
      // Accumulator
      count <= count + 1;
      bit done <= 1'b0;
    end
  end
end
```

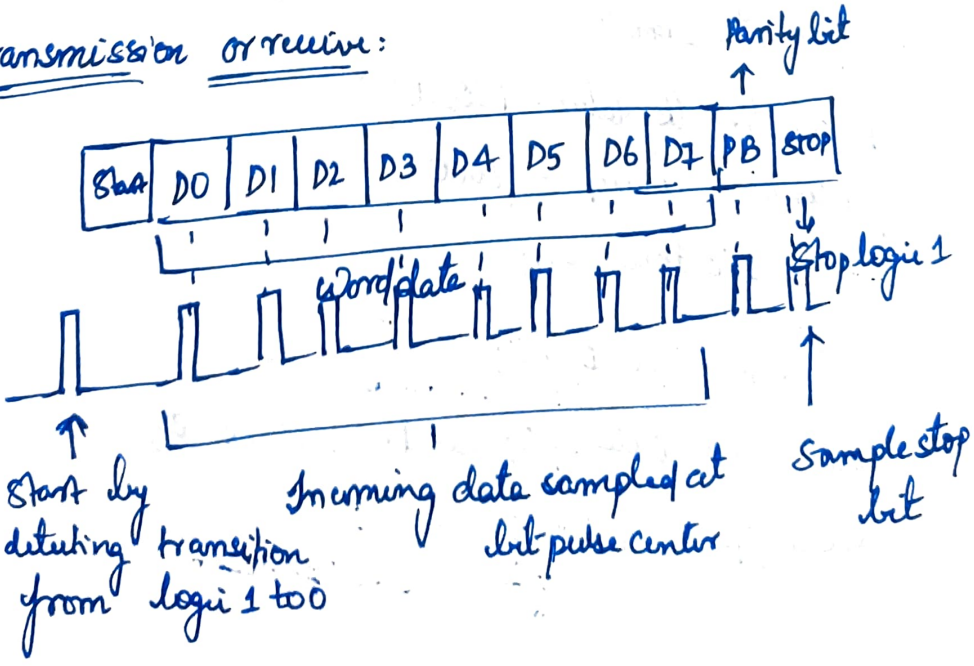
Stays at idle until the state changes to check or receive. & count stays tied on 0.

After succesful wait count the bit transmission is success & again the count turns to be 0.

This increments the count variable till the value reaches wait count, where bit done stays 0.

∴ Here bit_done acts to be as trigger.

For transmission or receive:

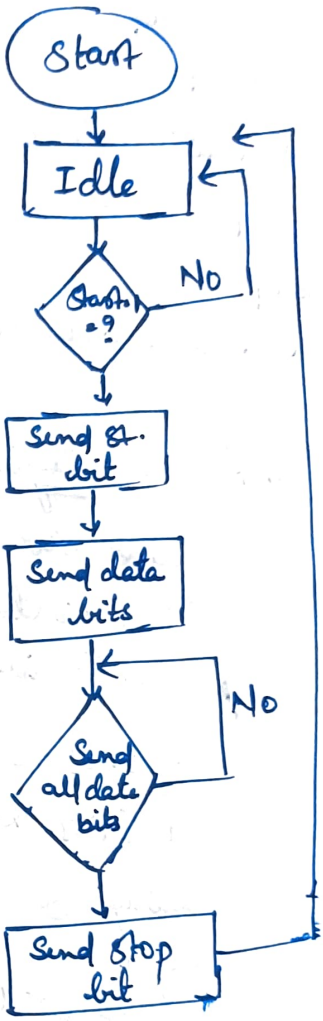


```
//Transmission logic:
reg [9:0] txData;
integer bitIndex = 0;
reg [9:0] shifttx = 0; //fordebug

always @(posedge clk) begin
case (state)
  idle: begin
          tx <= 1'b1;
          txData <= 0;
          bitIndex <= 0;
          shifttx <= 0;
        if (start == 1'b1) begin
           txData <= {1'b1, txin, 1'b0};
                      ↳stop  ↳data  ↳start
           state <= send;
        end
        else begin
          state <= idle;
        end.
    end.

send: begin
        tx <= txData[bitIndex];
        state <= check;
        shifttx <= {txData[bitIndex], shifttx[9:1]};
end.

check: begin
       if (bitIndex < 9) begin
            if(bitDone == 1'b1) begin
              begin
                state <= send;
                bitIndex <= bitIndex+1;
              end
            end
```



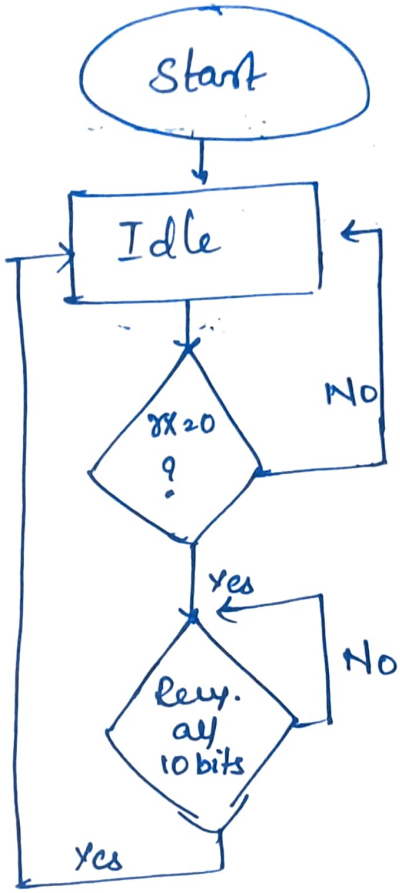
```
else begin
    State <= idle;
    bit Index <= 0;
    end
  end
default: State <= idle
endcase
end
assign tx done = (bit Index == 9 && bit done == 1'b1) ? 1'b1 : 1'b0;

// Receiver logic
  integer rcount = 0;
  integer rindex = 0;
  parameter ridle = 0, rwait = 1, recv = 2, rcheck = 3;

  reg [1:0] rstate;
  reg [9:0] rxdata;
  always @ (posedge clk)
  begin
    case(rstate)
      ridle : begin
                rxdata <= 0;
                rindex <= 0;
                rcount <= 0;
                  if (rx == 1'b0) begin
                      rstate <= rwait;
                  end
                else
                  begin
                    rstate <= ridle;
                  end
              end
rwait : begin
          if (rcount < wait count/2) begin  // sampling rx data at middle of bit duration
              rcount <= rcount + 1;
              rstate <= rwait;
            end
```

```
else
  begin
    rcount <= 0;
    rstate <= rcen;
    rxdata <= {rx, rxdata[9:1]};
  end
end

rcen: begin
        if(rindex < 9)
        begin
          if(bitDone == 1'b1) begin
              rindex <= rindex + 1;
              rstate <= rwait;
            end
        end
        else begin
            rstate <= ridle;
            rindex <= 0;
          end
    end
default : rstate <= ridle;
endcase
end.
assign rxout = rxdata[8:1];
assign rxdone = (rindex == 9 && bitdone == 1'b1) ? 1'b1 : 1'b0);
endmodule
```



Ports for the module.

```
module (input clk, start, input [7:0] txin, output tx,
        input rx, output [7:0] rxout, output rxdone,
        txdone);
```
