UART

Aim:- Write Verilog Code for UART and Carry out the following:

a) Perform functional Verification using test bench.

b) Synthesize the design targeting Suitable library and by setting area and timing Constraints.

c) For Various Constraints set, tabulate the area, power and delay for the Synthesized netlist.

d) Identify the Critical path and set the Constraints to obtain optimum gate level netlist with Suitable Constraints.

Design Code:-

```
module uart(reset, txclk, ld_tx_data, tx_data, tx_enable,
            tx_out, tx_empty, rxclk, uld_rx_data,
            rx_data, rx_enable, rx_in, rx_empty);
    input reset;
    input txclk;
    input ld_tx_data;
    input [7:0] tx_data;
    input tx_enable;
    output tx_out;
    output tx_empty;
    input rxclk;
    input uld_rx_data;
```

```
output [7:0] rx_data;
input rx_enable;
input rx_in;
output rx_empty;
reg [7:0] tx_reg;
reg tx_empty;
reg tx_over_run;
reg [3:0] tx_cnt;
reg tx_out;
reg [7:0] rx_reg;
reg [7:0] rx_data;
reg [3:0] rx_sample_cnt;
reg [3:0] rx_cnt;
reg rx_frame_err;
reg rx_over_run;
reg rx_empty;
reg rx_d1;
reg rx_d2;
reg rx_busy;
always @ (posedge rxclk or posedge reset)
if (reset)
  begin
  rx_reg <= 0;
  rx_data <= 0;
  rx_sample_cnt <= 0;
  rx_cnt <= 0;
  rx_frame_err <= 0;
  rx_over_run <= 0;
```

```
    rx_empty  <=1;
    rx_d1   <=1;
    rx_d2  <=1;
    rx_busy  <=0;
  end  else  begin
     rx_d1<=rx_in;
     rx_d2 <=rx_d1;
     if(uld_rx_data) begin
        rx_data <= rx_reg;
        rx_empty <=1;
     end
     if(rx_enable) begin
        if(!rx_busy && !rx_d2)  begin
           rx_busy  <=1;
           rx_sample_cnt <=1;
           rx_cnt   <=0;
        end
        if(rx_busy)   begin
           rx_sample_cnt <=rx_sample_cnt+1;
           if(rx_sample_cnt==7)  begin
              if((rx_d2==1) && (rx_cnt==0))
                 begin
                 rx_busy <=0;
              end  else  begin
                 rx_cnt <= rx_cnt+1;
                 if(rx_cnt>0 && rx_cnt<9)
                    begin
                    rx_reg[rx_cnt-1] <=rx_d2;
              end
```

```
if (rx_cnt==9)  begin
  rx_busy <=0;
  if (rx_d2==0)  begin
   rx_frame_err <=1;
   end else begin
    rx_empty <=0;
     rx_frame_err <=0;
     rx_over_run <= (rx_empty) ? 0:1;
    end
   end
  end
 end
end
end
 if (!rx_enable) begin
   rx_busy <=0;
  end
 end
 always @(posedge txclk or posedge reset)
  if (reset)  begin
     tx_reg   <=0;
     tx_empty  <=1;
     tx_over_run  <=0;
     tx_out  <=1;
     tx_cnt  <=0;
  end else begin
     if (ld_tx_data) begin
        if (! tx_empty) begin
  tx_over_run <=0;
end else begin
   tx_reg <=tx_data;
   tx_empty <=0;
  end
 end
  if(tx_enable && !tx_empty) begin
     tx_cnt <= tx_cnt+1;
    if(tx_cnt ==0) begin
      tx_out <=0;
    end
     if(tx_cnt >0 && tx_cnt <9) begin
        tx_out <= tx_reg[tx_cnt-1];
     end
      if(tx_cnt==9) begin
        tx_out <=1;
        tx_cnt <=0;
        tx_empty <=1;
       end
      end
      if(!tx_enable) begin
         tx_cnt <=0;
       end
     end
endmodule
```

Test bench Code :-

```
module uart_tb;
        reg reset;
        reg txclk;
        reg ld_tx_data;
        reg [7:0] tx_data;
        reg tx_enable;
        reg rxclk;
        reg uld_rx_data;
        reg rx_enable;
        reg rx_in;
        wire tx_out;
        wire tx_empty;
        wire [7:0] rx_data;
        wire rx_empty;
    //uncomment lines for convenient access to internal
                                                   Var
    //wire [7:0] rx_reg = uut.rx_reg;
    // wire [3:0] rx_cnt = uut.rx_cnt;
    // wire [3:0] rx_sample_cnt = uut.rx_sample_cnt;
    // wire rx_d2 = uut.rx_d2;
    // wire rx_busy = uut.rx_busy;
          uart uut(.reset(reset), .txclk(txclk), .ld_
          tx_data(ld_tx_data), .tx_data(tx_data),
          .tx_enable(tx_enable), .tx_out(tx_out),
          .tx_empty(tx_empty), .rxclk(rxclk),
          .uld_rx_data(uld_rx_data), .rx_data
          (rx_data), .rx_enable(rx_enable), .rx_in
(rx_in), .rx_empty(rx_empty));
reg clk;
initial
 clk=0;
 always #10 clk=~clk;
reg [3:0] Counter;
  initial
   begin
   rxclk =0;
   txclk=0;
   Counter =0;
  end
  always @(posedge clk) begin
     Counter<=Counter+1;
     if(Counter==15)
       txclk<=~txclk;
       rxclk<=~rxclk;
     end
     always @(tx_out) rx_in=tx_out;
        initial
          begin
          reset=1;
          ld_tx_data=0;
          tx_data=0;
          tx_enable=1;
          uld_rx_data=0;
           rx_enable=1;
           rx_in=1;
```

```
    #500;
    reset = 0;
  tx_data = 8'b0111_1111;
  #500;
  wait (tx_empty == 1);
  ld_tx_data = 1;
  wait ( tx_empty == 0);
  $display(" Data loaded for Send");
   ld_tx_data = 0;
   wait (tx_ empty == 1);
   $display (" Data Sent");
    wait( rx_empty == 0);
    $display(" RX Byte Ready");
           uld_rx_data = 1;
     wait (rx_ empty == 1);
     $display (" RX Byte Unloaded: %b", rx_data);
     #100;
     $finish;
  end
endmodule
```