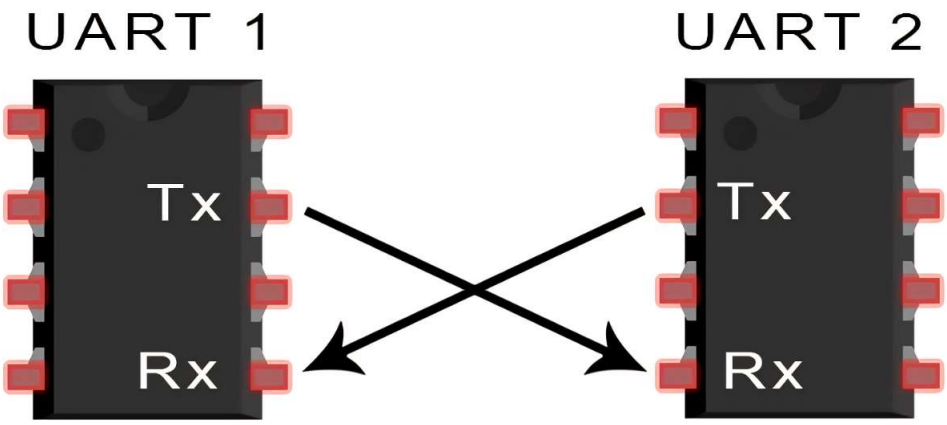
UART_TX

Verify by SVA

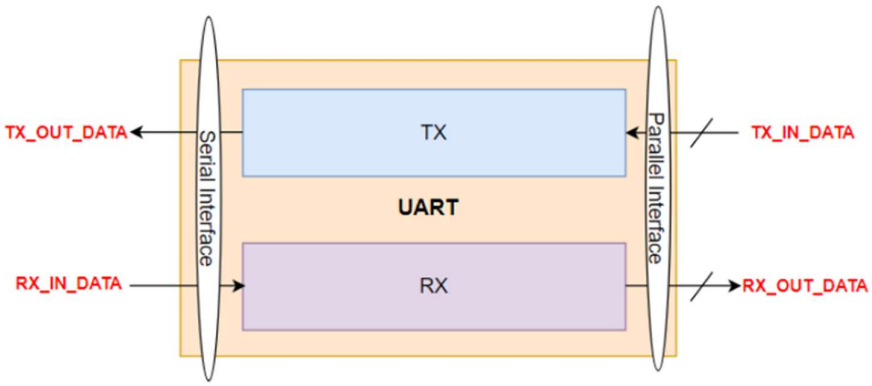
By: Mohammed Anwar
Abd allatif



Verification of UART Transmitter

Introduction :-

- There are many serial communication protocol as I2C, UART and SPI.
- A Universal Asynchronous Receiver/Transmitter (UART) is a block of circuitry responsible for implementing serial communication.
- UART is Full Duplex protocol (data transmission in both directions simultaneously)
- Transmitting UART converts parallel data from the master device (eg. CPU) into serial form and transmit in serial to receiving UART.
- Receiving UART will then convert the serial data back into parallel data for the receiving device.



RTL DESIGN_CODE:

D: > IEEE Verifiction > Final Project > UART_TX_UVM > UART_TX_files_UVM > UART_TX.sv

```
module uart_tx (uart_tx_if.DUT R_if);


  reg [3:0] state = R_if.IDLE;
  reg [3:0] counter = 0;
  reg [7:0] data_reg;
  reg parity_bit;

  always @(posedge R_if.clk or negedge R_if.reset) begin
    if (!R_if.reset) begin
      state <= R_if.IDLE;
      R_if.TX_OUT <= 1'b1;
      R_if.Busy <= 1'b0;
      counter <= 0;
    end else begin
      case (state)
        R_if.IDLE: begin
          R_if.TX_OUT <= 1'b1;
          R_if.Busy <= 1'b0;
        counter <=0 ;
        if (R_if.DATA_VALID) begin
            data_reg <= R_if.P_DATA;
            parity_bit <= (R_if.PAR_TYP == 0 ) ? ~^R_if.P_DATA : ^R_if.P_DATA ;
            state <= R_if.START;
            R_if.Busy <= 1'b1;
        end
          end
        R_if.START: begin
         R_if.TX_OUT <= 1'b0;
         state <= R_if.DATA;
         counter <= 0 ;

        end

        R_if.DATA: begin
          R_if.TX_OUT <= data_reg[counter];
          counter <= counter + 1 ;
          if (counter == 7) begin
            state <= (R_if.PAR_EN) ? R_if.PARITY : R_if.STOP ;
            end
        end

        R_if.PARITY: begin
          R_if.TX_OUT <= parity_bit ;
          state <= R_if.STOP;
        end

        R_if.STOP: begin
          R_if.TX_OUT <= 1'b1;
          state <= R_if.IDLE;
        end
      endcase
    end
  end

```

GOLDEN MODEL_CODE :

UART_TX_GOLDEN.sv

```
module uart_golden_model(uart_tx_if.GOLD u_if);
    reg [3:0] counter = 0;
    reg [7:0] DATA_reg;
    reg PARITY_bit;
    reg [3:0] cs, ns;

    // State memory
    always @(posedge u_if.clk or negedge u_if.reset) begin
        if (!u_if.reset) begin
            cs <= u_if.IDLE;
        end
        else begin
            cs <= ns;
        end
    end

    // Next state Logic
    always @(*) begin
        case(cs)
            u_if.IDLE: begin
                if (u_if.DATA_VALID ) begin
                    ns = u_if.START;
                end
            end
            u_if.START: ns = u_if.DATA;
            u_if.DATA: begin
                if (counter == 7) begin
                    ns = (u_if.PAR_EN) ? u_if.PARITY : u_if.STOP;
                end
            end
            u_if.PARITY: ns = u_if.STOP;
            u_if.STOP: ns = u_if.IDLE;
            default: ns = u_if.IDLE;
        endcase
    end

    // Output logic
    always @(posedge u_if.clk or negedge u_if.reset) begin
        if (!u_if.reset) begin
            u_if.TX_OUT_ex <= 1'b1;
            u_if.Busy_ex <= 1'b0;
            counter <= 0;
        end
        else begin
            case (cs)
                u_if.IDLE: begin
                    u_if.TX_OUT_ex <= 1'b1;
                    u_if.Busy_ex <= 1'b0;
                    counter <= 0;
                    if (u_if.DATA_VALID) begin
                        DATA_reg <= u_if.P_DATA;
                        PARITY_bit <= (u_if.PAR_TYP == 0) ? ~^u_if.P_DATA : ^u_if.P_DATA;
                        u_if.Busy_ex <= 1'b1;
                    end
                end
                                ns = (u_if.PAR_EN) ? u_if.PARITY : u_if.STOP;
                            end
                        end
                        u_if.PARITY: ns = u_if.STOP;
                        u_if.STOP: ns = u_if.IDLE;
                        default: ns = u_if.IDLE;
                    endcase
                end

                // Output logic
                always @(posedge u_if.clk or negedge u_if.reset) begin
                    if (!u_if.reset) begin
                        u_if.TX_OUT_ex <= 1'b1;
                        u_if.Busy_ex <= 1'b0;
                        counter <= 0;
                    end
                    else begin
                        case (cs)
                            u_if.IDLE: begin
                                u_if.TX_OUT_ex <= 1'b1;
                                u_if.Busy_ex <= 1'b0;
                                counter <= 0;
                                if (u_if.DATA_VALID) begin
                                    DATA_reg <= u_if.P_DATA;
                                    PARITY_bit <= (u_if.PAR_TYP == 0) ? ~^u_if.P_DATA : ^u_if.P_DATA;
                                    u_if.Busy_ex <= 1'b1;
                                end
                            end
                            u_if.START: begin
                                u_if.TX_OUT_ex <= 1'b0;
                                counter <= 0;
                            end
                            u_if.DATA: begin
                                u_if.TX_OUT_ex <= DATA_reg[counter];
                                counter <= counter + 1;
                            end
                            u_if.PARITY: begin
                                u_if.TX_OUT_ex <= PARITY_bit;
                            end
                            u_if.STOP: begin
                                u_if.TX_OUT_ex <= 1'b1;
                            end
                        endcase
                    end
                end
            endmodule
```

Interface :

UART_TX_IF.sv

```
interface uart_tx_if(clk);

// parameter Declaration
parameter IDLE = 0;
parameter START = 1;
parameter DATA = 2;
parameter PARITY = 3;
parameter STOP = 4;

//---------------- input declaration ---------------
input bit clk;
logic reset ;
logic PAR_EN;
logic PAR_TYP;
logic  DATA_VALID;
logic [7:0] P_DATA;
//---------------- output declaration --------------
logic TX_OUT , TX_OUT_ex ;
logic Busy ,Busy_ex ;

//_____________________________ MODPORTS _____________________________________//

modport DUT (input clk , reset , PAR_EN , PAR_TYP , DATA_VALID , P_DATA ,output TX_OUT , Busy );


modport TEST(input clk, TX_OUT , Busy, TX_OUT_ex , Busy_ex   , output  reset , PAR_EN , PAR_TYP , DATA_VALID , P_DATA);


modport MONITOR(output clk ,  reset , PAR_EN , PAR_TYP , DATA_VALID , P_DATA , TX_OUT , Busy , TX_OUT_ex , Busy_ex  );


modport GOLD (input clk , reset , PAR_EN , PAR_TYP , DATA_VALID , P_DATA ,output TX_OUT_ex , Busy_ex );


endinterface
```

Top module :

UART_TX_TOP.sv

```
module UART_TX_TOP;
  bit clk;

  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end

  uart_tx_if R_if(clk);
  uart_tx dut(R_if);
  uart_tb tb(R_if);
  uart_golden_model golden(R_if);
  bind uart_tx uart_tx_assertions T1(R_if);
endmodule
```

UART_TX TEST_ :

UART_TX_TEST.sv

```
import UART_TX_pkg::*;
module uart_tb (uart_tx_if.TEST u_if) ;

uart_tx_pkg uc = new () ;

int correct_counter = 0 ;
int error_counter = 0 ;

initial begin
  //UART_1
  assert_rst;
  //UART_2
  repeat(786) begin
  assert (uc.randomize())
  u_if.DATA_VALID = uc.DATA_VALID;
  u_if.P_DATA = uc.P_DATA ;
  u_if.PAR_EN = uc.PAR_EN ;
  u_if.PAR_TYP = uc.PAR_TYP ;
  @(posedge u_if.clk);
  u_if.DATA_VALID = 1'b0;
  repeat (9) begin
  @(posedge u_if.clk) ;
    uc.TX_OUT = u_if.TX_OUT ;
    uc.Busy = u_if.Busy ;
    @(posedge u_if.clk) ;
    check_result ;
  end
  end


  #2 $display ("NO. of Correct operations : 0d%0d , No. of errors : 0d%0d "
  ,correct_counter,error_counter);
  #2;
  $stop;
end

always @(posedge u_if.clk ) begin
  uc.uart_cg.sample();
end

task assert_rst ;
    u_if.reset = 0 ;
    @(posedge u_if.clk);
    u_if.reset = 1 ;
endtask

task check_result ;
    if ((u_if.TX_OUT === u_if.TX_OUT_ex)&&(u_if.Busy === u_if.Busy_ex)) correct_counter ++ ;
    else begin
      $display ("Error at time 0t%0t " , $time );
      error_counter++ ;
    end
endtask

endmodule
```

UART_TX MONITOR :

UART_TX_MON.sv

```
module uart_monitor (uart_tx_if.MONITOR u_if) ;
  always @(posedge u_if.clk ) begin
    $display("STIMULUS : Reset = 0b%0b , P_DATA = 0h%0h , PAR_EN = 0b%0b , PAR_TYP = 0b%0b ,
    DATA_VALID = 0b%0b " , u_if.reset , u_if.P_DATA , u_if.PAR_EN , u_if.PAR_TYP ,
    u_if.DATA_VALID);
    $display("OUTPUT : TX_OUT = 0b%0b , TX_OUT_ex = 0b%0b ,
              Busy = 0b%0b , Busy_ex = 0b%0b " ,
              u_if.TX_OUT,u_if.TX_OUT_ex,u_if.Busy,u_if.Busy_ex);
  end
endmodule
```

UART_TX COVERAGE_PKG :

UART_pkg.sv

```
package UART_TX_pkg;

    class uart_tx_pkg;

// FSM state encoding (for future use or ref)
  parameter IDLE   = 0;
  parameter START  = 1;
  parameter DATA   = 2;
  parameter PARITY = 3;
  parameter STOP   = 4;



  // Control signals
  rand logic reset;                    // Active-low reset
  rand logic PAR_EN;                   // 0: disable parity, 1: enable parity
  rand logic PAR_TYP;                  // 0: even, 1: odd
  rand logic DATA_VALID;
  logic clk;

  // Data
  rand logic [7:0] P_DATA;

  // DUT outputs
  logic TX_OUT;
  logic Busy;

  ////////////////////////////////////////////////////////////////////////////////////////
  // Constraints - Functional Coverage Related
  ////////////////////////////////////////////////////////////////////////////////////////

  // I- Reset active for ~3% of simulation time
  constraint c_reset_active {
    reset dist {0 := 3, 1 := 97}; // Active-low reset
  }

  // II- PAR_TYP toggle: 50% even, 50% odd
  constraint c_par_typ_toggle {
    PAR_TYP dist {0 := 50, 1 := 50};
  }

  // III- PAR_EN enabled 25% of time
  constraint c_par_en {
    PAR_EN dist {1 := 25, 0 := 75};
  }

  // IV- DATA_VALID active most of simulation time
  constraint c_data_valid {
    DATA_VALID dist {1 := 85, 0 := 15};
  }

  // V- P_DATA specs:
  // a. LSB = 1 in 80% of cases
  constraint c_lsb_one {
    (P_DATA[0] == 1) dist {1 := 80, 0 := 20};
  }

  // b. Specific patterns (11111111, 00000000, 10101010) appear 4% of time
  constraint c_specific_patterns {
    P_DATA dist {
      8'b11111111 := 1,
      8'b00000000 := 1,
      8'b10101010 := 1,
      [8'b00000001:8'b11111110] := 96
    };
  }
```

```
  }
  covergroup uart_cg ;

  RST_cp : coverpoint reset {
    bins zero = {0} ;
    bins one = {1} ;
  }
  PAR_TYP_cp : coverpoint PAR_TYP {
    bins zero = {0} ;
    bins one = {1} ;
  }
  PAR_EN_cp : coverpoint PAR_EN {
    bins zero = {0} ;
    bins one = {1} ;
  }
  P_DATA_cp : coverpoint P_DATA {
    bins all_values = {[0:255]};
  }
  DATA_VALID_cp : coverpoint DATA_VALID {
    bins zero = {0} ;
    bins one = {1} ;
  }
  TX_OUT_cp : coverpoint TX_OUT {
    bins zero = {0} ;
    bins one = {1} ;
  }
  Busy_cp : coverpoint Busy {
    bins zero = {0} ;
    bins one = {1} ;
  }

  PAR_EN_cross_TYPE : cross PAR_EN_cp , PAR_TYP_cp ;
  P_DATA_cross_DATA_VALID : cross P_DATA_cp , DATA_VALID_cp ;
  TX_OUT_cross_Busy : cross TX_OUT_cp , Busy_cp ;

  endgroup

function new();
    uart_cg = new();

endfunction



    endclass
endpackage
```

UART_TX ASSERTIONS:

D: > IEEE Verification > Final Project > UART_TX_UVM > UART_TX_files_UVM > UART_TX_SVA.sv

```
module uart_tx_assertions (uart_tx_if.DUT vif);

// Final reset checks after reset is deasserted
always_comb begin
  if (!vif.reset) begin
    a_assert: assert final(uart_tx.state == vif.IDLE);
    b_assert: assert final(vif.TX_OUT == 1'b1);
    c_assert: assert final(vif.Busy == 1'b0);
    d_assert: assert final(uart_tx.counter == 0);
  end
end

// 1. TX_OUT should be high and Busy low in IDLE state
property idle_behavior;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1'b0) |->
  (vif.TX_OUT == 1'b1 && vif.Busy == 1'b0 && uart_tx.counter == 0);
endproperty
p1: assert property(idle_behavior) else $error("TX_OUT should be 1 and Busy 0 in IDLE state.");
c1: cover property(idle_behavior);

// 2. FSM should remain in IDLE when DATA_VALID = 0
property idle_2;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1'b0) |-> (uart_tx.state == vif.IDLE);
endproperty
p2: assert property(idle_2) else $error("State should remain IDLE when DATA_VALID is 0.");
c2: cover property(idle_2);

// 3. START state: TX_OUT should be 0 and counter = 0
property start_bit_low;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.START) |-> (vif.TX_OUT == 1'b0 && uart_tx.counter == 0);
endproperty
p3: assert property(start_bit_low) else $error("TX_OUT should be 0 and counter 0 in START state.");
c3: cover property(start_bit_low);

// 4. DATA state continues when counter != 7
property data_bit_1;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.DATA && uart_tx.counter != 7) |-> (uart_tx.state == vif.DATA);
endproperty
p4: assert property(data_bit_1) else $error("FSM should stay in DATA state while counter != 7.");
c4: cover property(data_bit_1);

// 5. TX_OUT should match data_reg[counter] and counter must increment
property data_bit_match;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.DATA && uart_tx.counter != 7) |->
  (vif.TX_OUT == uart_tx.data_reg[$past(uart_tx.counter)] &&
   uart_tx.counter == $past(uart_tx.counter) + 1);
endproperty
p5: assert property(data_bit_match) else $error("TX_OUT mismatch or counter didn't increment in DATA state.");
c5: cover property(data_bit_match);

// 6. PARITY state: TX_OUT should match calculated parity bit
property parity_correct;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.PARITY) |-> (vif.TX_OUT == uart_tx.parity_bit);
endproperty
p6: assert property(parity_correct) else $error("TX_OUT does not match expected parity bit.");
c6: cover property(parity_correct);

// 7. STOP state: TX_OUT should be high
property stop_bit_high;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.STOP) |-> (vif.TX_OUT == 1'b1);
endproperty
p7: assert property(stop_bit_high) else $error("TX_OUT must be high in STOP state.");
c7: cover property(stop_bit_high);
```

```
// 8. No transition from IDLE if DATA_VALID is not asserted
property no_start_without_valid;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 0) |=> (uart_tx.state == vif.IDLE);
endproperty
p8: assert property(no_start_without_valid) else $error("FSM left IDLE without DATA_VALID = 1.");
c8: cover property(no_start_without_valid);

// 9. Even parity logic from IDLE
property idle_even_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1 && vif.PAR_TYP == 0) |->
  (uart_tx.parity_bit == ~^$past(vif.P_DATA) &&
   vif.Busy == 1'b1 &&
   uart_tx.data_reg == $past(vif.P_DATA));
endproperty
p9: assert property(idle_even_parity) else $error("Even parity incorrect after IDLE.");
c9: cover property(idle_even_parity);

// 10. Odd parity logic from IDLE
property idle_odd_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1 && vif.PAR_TYP == 1) |->
  (uart_tx.parity_bit == ^$past(vif.P_DATA) &&
   vif.Busy == 1'b1 &&
   uart_tx.data_reg == $past(vif.P_DATA));
endproperty
p10: assert property(idle_odd_parity) else $error("Odd parity incorrect after IDLE.");
c10: cover property(idle_odd_parity);

// 11. Transition from IDLE to START when DATA_VALID = 1
property idle_to_start;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1) |=> (uart_tx.state == vif.START);
endproperty
p11: assert property(idle_to_start) else $error("Should move from IDLE to START when DATA_VALID = 1.");
c11: cover property(idle_to_start);

// 12. DATA to PARITY transition when counter==7 and PAR_EN==1
property data_to_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.DATA && uart_tx.counter == 7 && vif.PAR_EN == 1) |->
  (uart_tx.state == vif.PARITY);
endproperty
p12: assert property(data_to_parity) else $error("Should go to PARITY after DATA if PAR_EN == 1.");
c12: cover property(data_to_parity);

// 13. DATA to STOP transition when counter==7 and PAR_EN==0
property data_to_stop;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.DATA && uart_tx.counter == 7 && vif.PAR_EN == 0) |->
  (uart_tx.state == vif.STOP);
endproperty
p13: assert property(data_to_stop) else $error("Should go to STOP after DATA if PAR_EN == 0.");
c13: cover property(data_to_stop);

// 14. PARITY to STOP transition
property parity_to_stop;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.PARITY) |->
  (vif.TX_OUT == $past(uart_tx.parity_bit) && uart_tx.state == vif.STOP);
endproperty
p14: assert property(parity_to_stop) else $error("Should go to STOP after PARITY.");
c14: cover property(parity_to_stop);

// 15. STOP to IDLE transition
property stop_to_idle;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.STOP) |->
  (vif.TX_OUT == 1 && uart_tx.state == vif.IDLE);
endproperty
p15: assert property(stop_to_idle) else $error("Should go to IDLE after STOP.");
c15: cover property(stop_to_idle);

endmodule
```

DO_File :

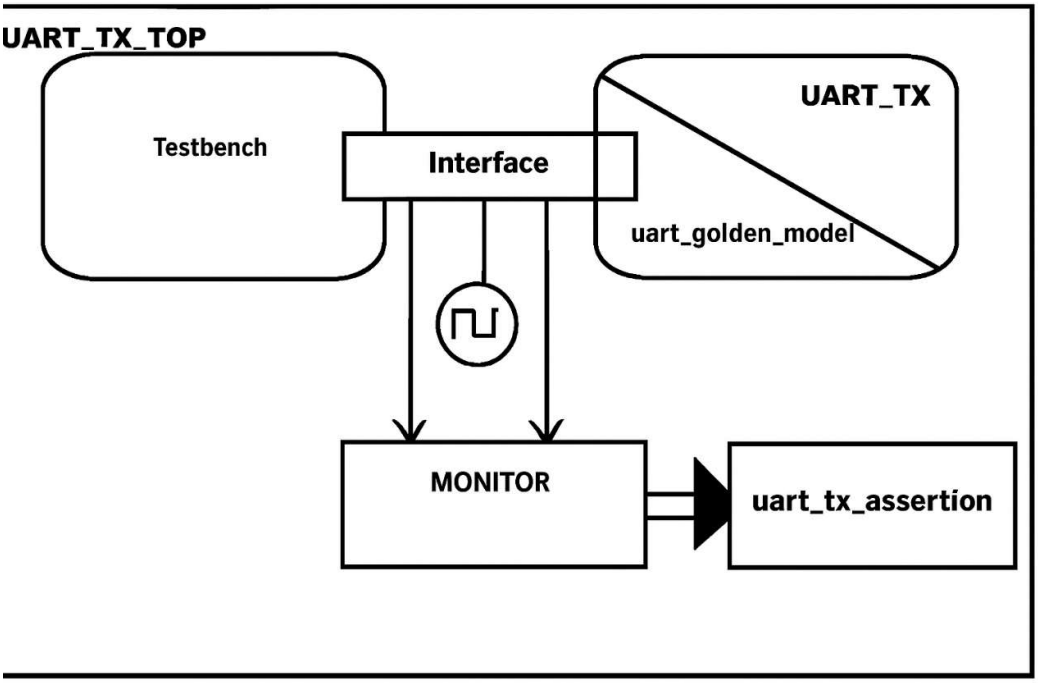
run.do

```
vlib work

vlog -sv +cover +acc \
UART_TX_IF.sv \
UART_TX.sv \
UART_TX_GOLDEN.sv \
UART_TX_MON.sv \
UART_TX_SVA.sv \
UART_TX_TEST.sv \
UART_TX_TOP.sv \
UART_pkg.sv

vsim -voptargs=+acc work.UART_TX_TOP -classdebug -uvmcontrol=all -coverage

add wave /UART_TX_TOP/clk
add wave -hex /UART_TX_TOP/R_if/*


run -all

coverage save uart_tx.ucdb -onexit


vcover report uart_tx.ucdb -details -all -annotate -output coverage.txt

# quit -sim

```

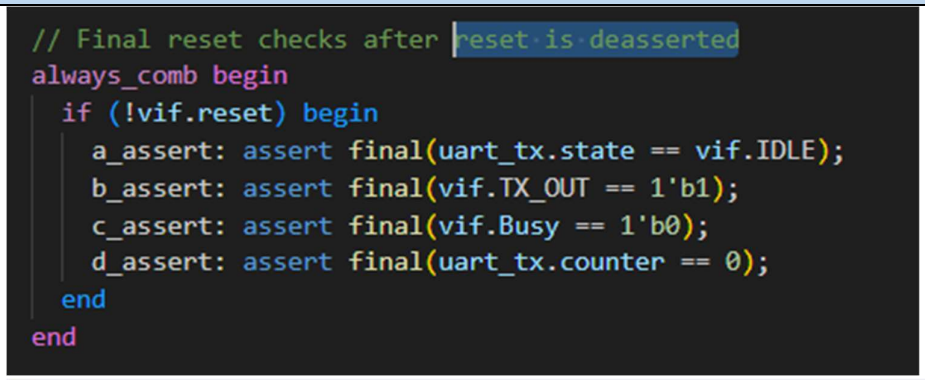
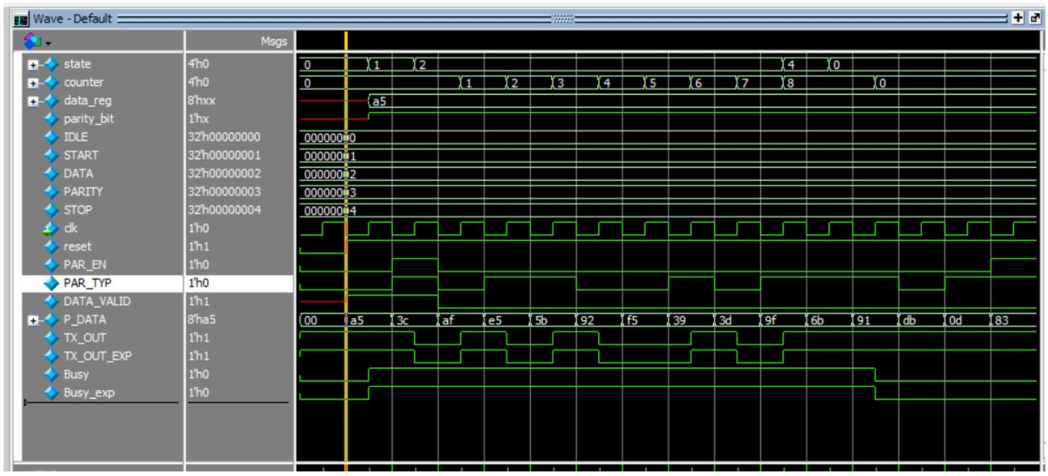
List:

```
UART_TX.sv
UART_TX_GOLDEN.sv
UART_TX_IF.sv
UART_TX_MON.sv
UART_TX_SVA.sv
UART_TX_TEST.sv
UART_TX_TOP.sv
UART_pkg.sv
```

PART1 :- SVA testbench structure:



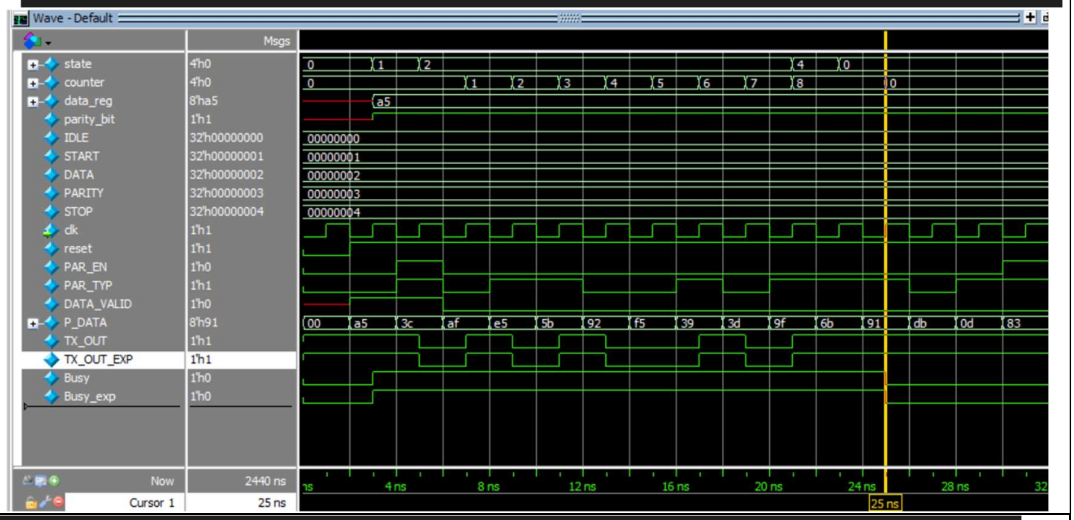
2 Assertions table:

Feature	Assertion
When reset is deasserted : State at IDLE ,TX_OUT = 1'B1,Busy = 1'b0,counter=0	 

```
// Final reset checks after reset is deasserted
always_comb begin
  if (!vif.reset) begin
    a_assert: assert final(uart_tx.state == vif.IDLE);
    b_assert: assert final(vif.TX_OUT == 1'b1);
    c_assert: assert final(vif.Busy == 1'b0);
    d_assert: assert final(uart_tx.counter == 0);
  end
end
```

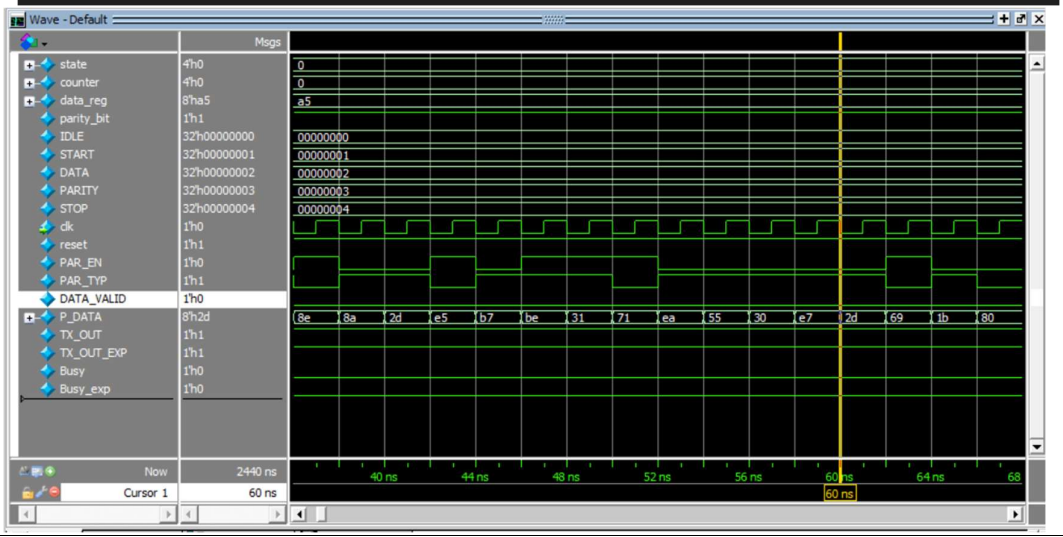
TX_OUT should be high and busy ;ow in IDLE STATE

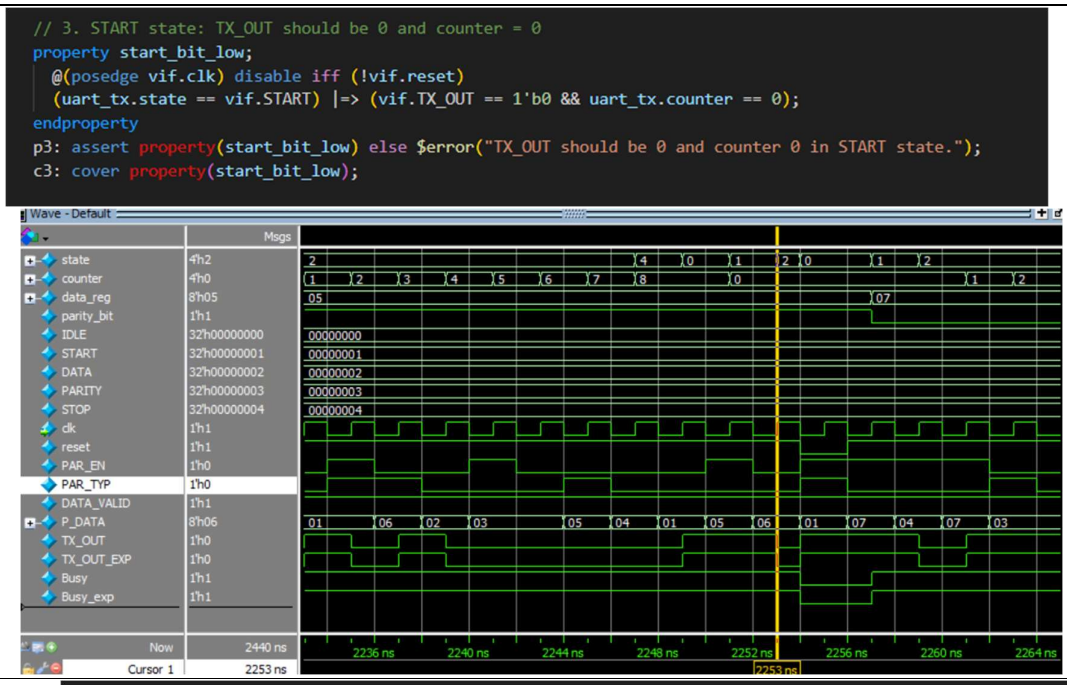
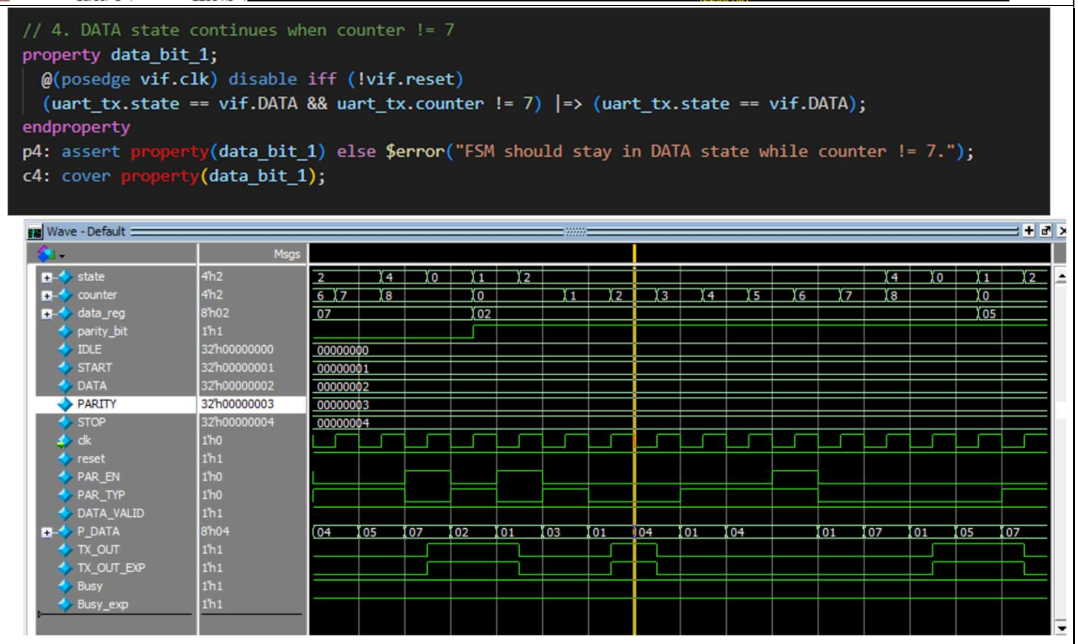
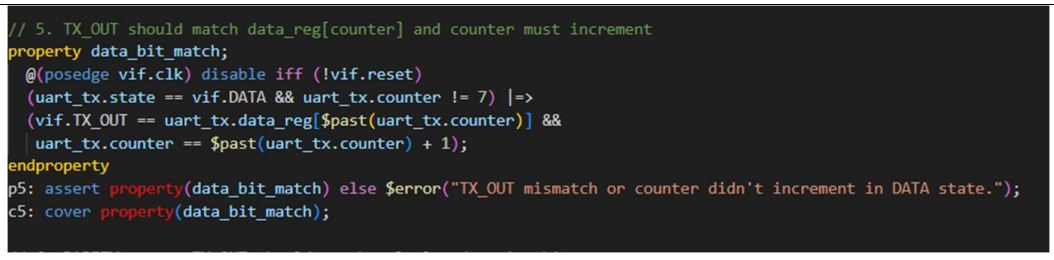
```
// 1. TX_OUT should be high and Busy low in IDLE state
property idle_behavior;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1'b0) |=>
  (vif.TX_OUT == 1'b1 && vif.Busy == 1'b0 && uart_tx.counter == 0);
endproperty
p1: assert property(idle_behavior) else $error("TX_OUT should be 1 and Busy 0 in IDLE state.");
c1: cover property(idle_behavior);
```

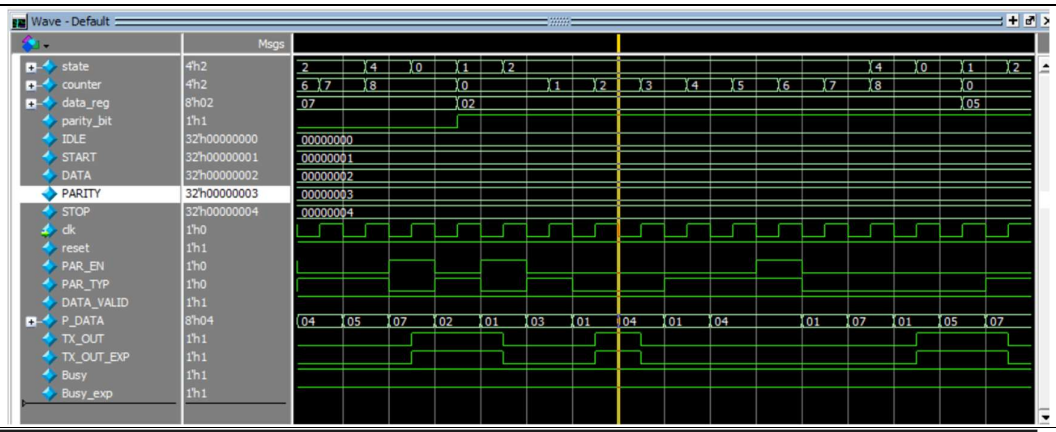
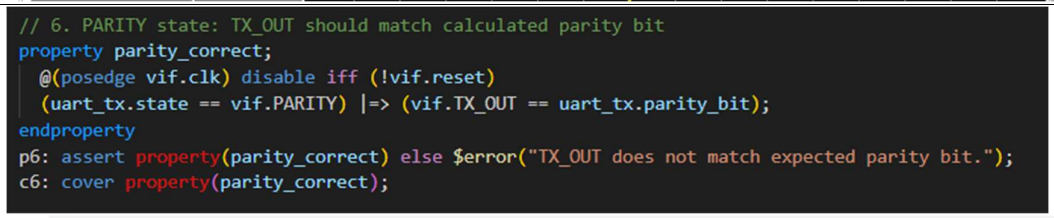
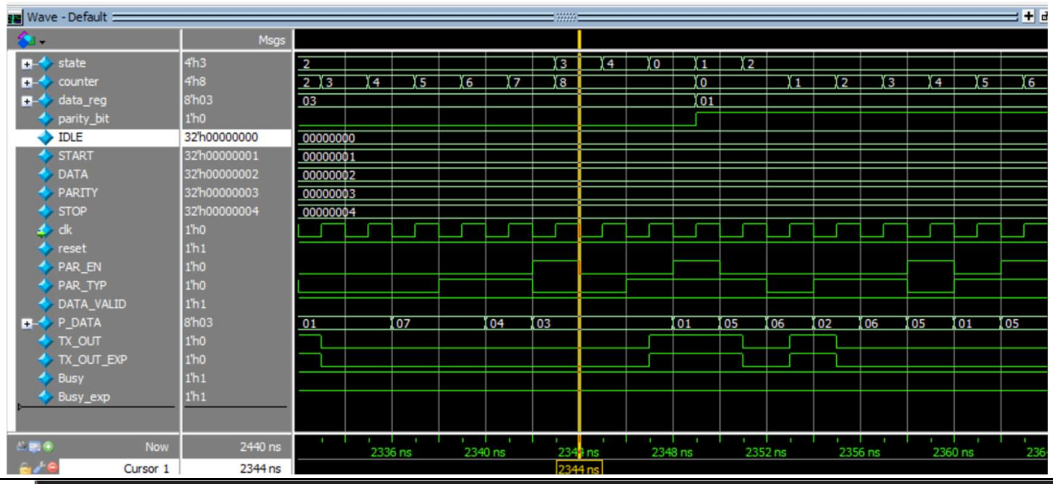
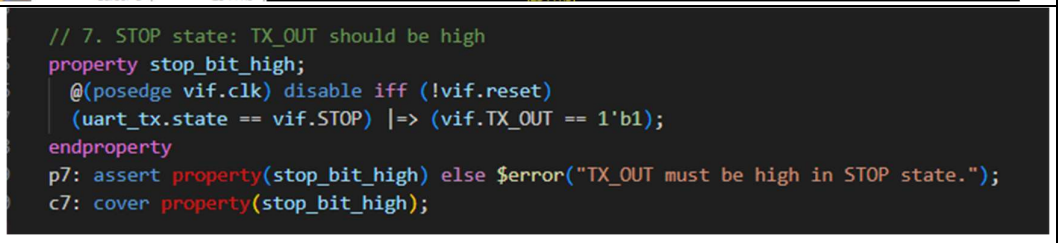


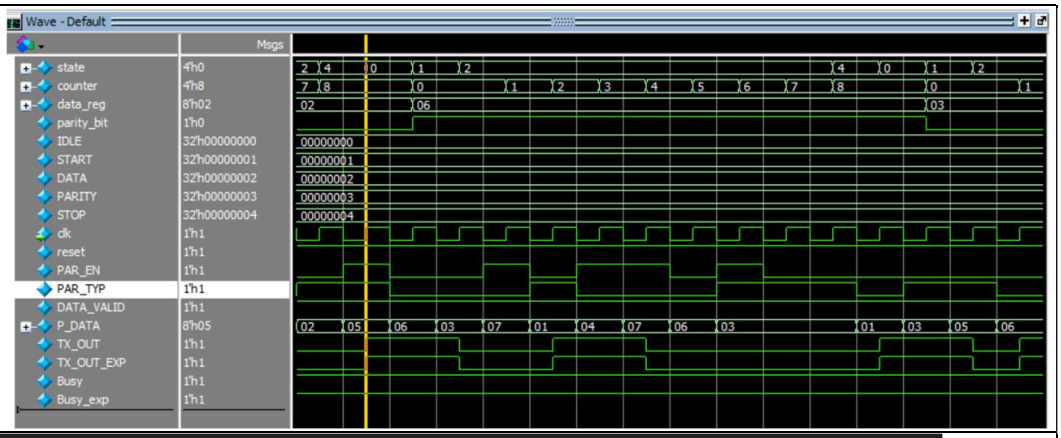
FSM should remain in IDLE when DATA_VALID =0

```
// 2. FSM should remain in IDLE when DATA_VALID = 0
property idle_2;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1'b0) |=> (uart_tx.state == vif.IDLE);
endproperty
p2: assert property(idle_2) else $error("State should remain IDLE when DATA_VALID is 0.");
c2: cover property(idle_2);
```



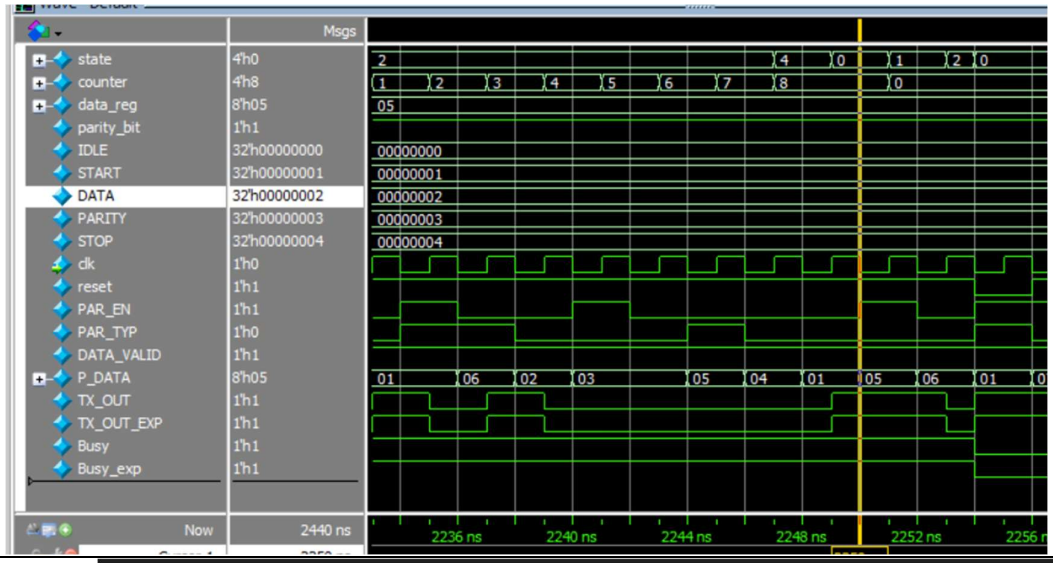
TX_OUT should be 0 and counter =0 at state START	
DATA state continues when counter not equal 7	
TX_OUT should match data_reg[counter] and counter must increment	

	
PARITY state: TX_OUT should match calculated parity bit	 
STOP state: TX_OUT should be high.	



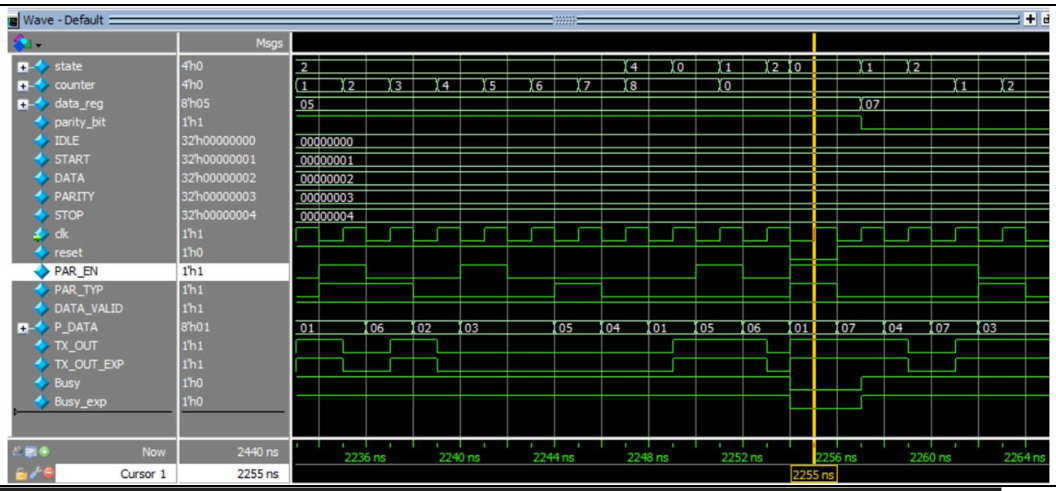
Even parity logic from IDLE.

```
// 9. Even parity logic from IDLE
property idle_even_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1 && vif.PAR_TYP == 0) |=>
  (uart_tx.parity_bit == ~^$past(vif.P_DATA) &&
   vif.Busy == 1'b1 &&
   uart_tx.data_reg == $past(vif.P_DATA));
endproperty
p9: assert property(idle_even_parity) else $error("Even parity incorrect after IDLE.");
c9: cover property(idle_even_parity);
```



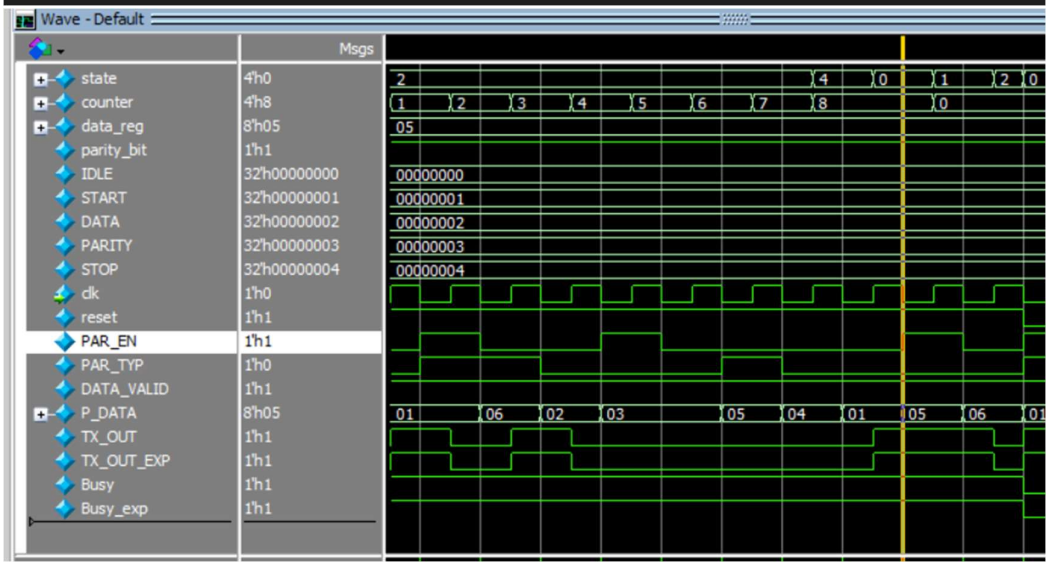
Odd parity logic from IDLE

```
// 10. Odd parity logic from IDLE
property idle_odd_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1 && vif.PAR_TYP == 1) |=>
  (uart_tx.parity_bit == ^$past(vif.P_DATA) &&
   vif.Busy == 1'b1 &&
   uart_tx.data_reg == $past(vif.P_DATA));
endproperty
p10: assert property(idle_odd_parity) else $error("Odd parity incorrect after IDLE.");
c10: cover property(idle_odd_parity);
```



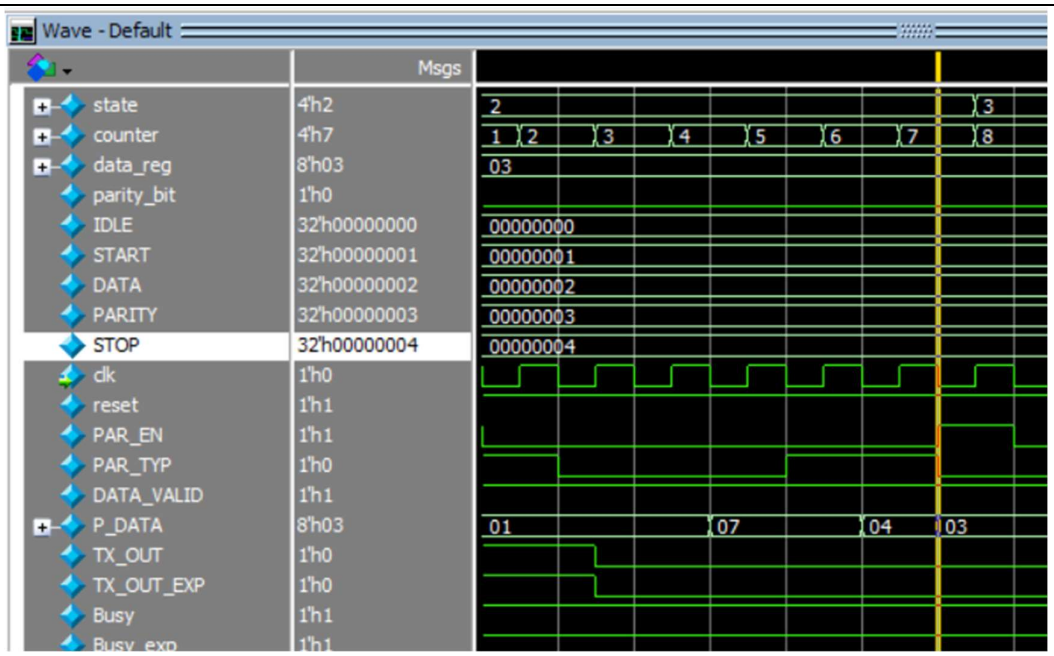
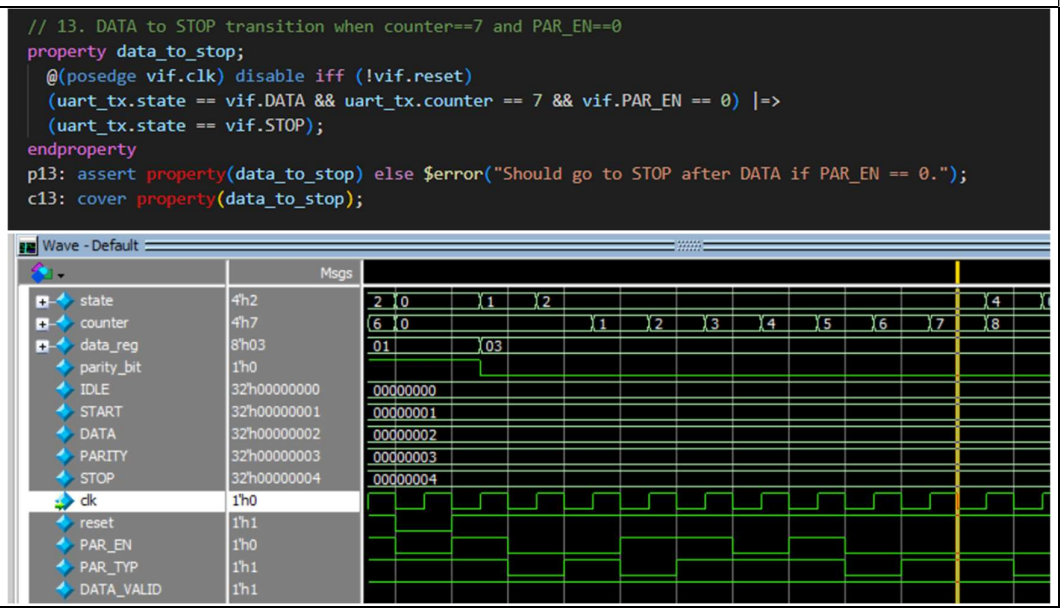
Transition from IDLE to START when DATA_VALID = 1.

```
// 11. Transition from IDLE to START when DATA_VALID = 1
property idle_to_start;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.IDLE && vif.DATA_VALID == 1) |=> (uart_tx.state == vif.START);
endproperty
p11: assert property(idle_to_start) else $error("Should move from IDLE to START when DATA_VALID = 1.");
c11: cover property(idle_to_start);
```



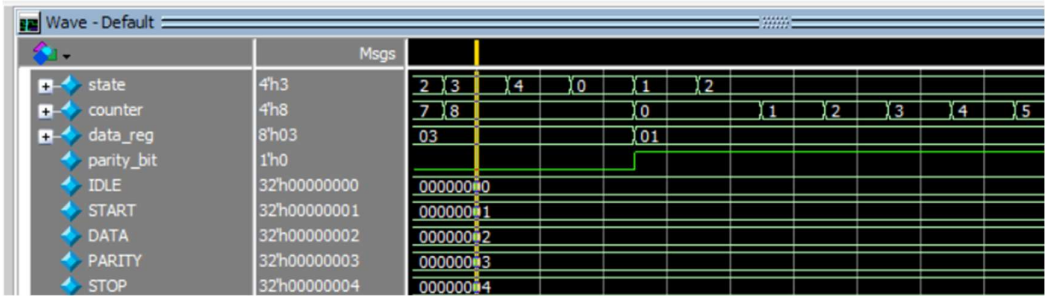
DATA to PARITY transition when counter==7 and PAR_EN==1

```
// 12. DATA to PARITY transition when counter==7 and PAR_EN==1
property data_to_parity;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.DATA && uart_tx.counter == 7 && vif.PAR_EN == 1) |=>
  (uart_tx.state == vif.PARITY);
endproperty
p12: assert property(data_to_parity) else $error("Should go to PARITY after DATA if PAR_EN == 1.");
c12: cover property(data_to_parity);
```

	
DATA to STOP transition when counter==7 and PAR_EN==0	

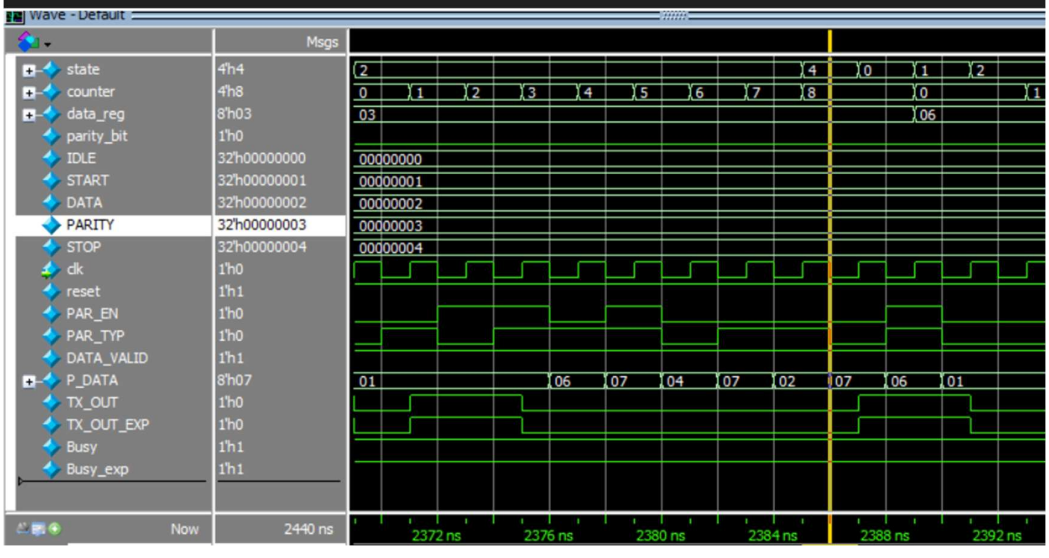
PARITY to STOP transition

```
// 14. PARITY to STOP transition
property parity_to_stop;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.PARITY) |=>
  (vif.TX_OUT == $past(uart_tx.parity_bit) && uart_tx.state == vif.STOP);
endproperty
p14: assert property(parity_to_stop) else $error("Should go to STOP after PARITY.");
c14: cover property(parity_to_stop);
```

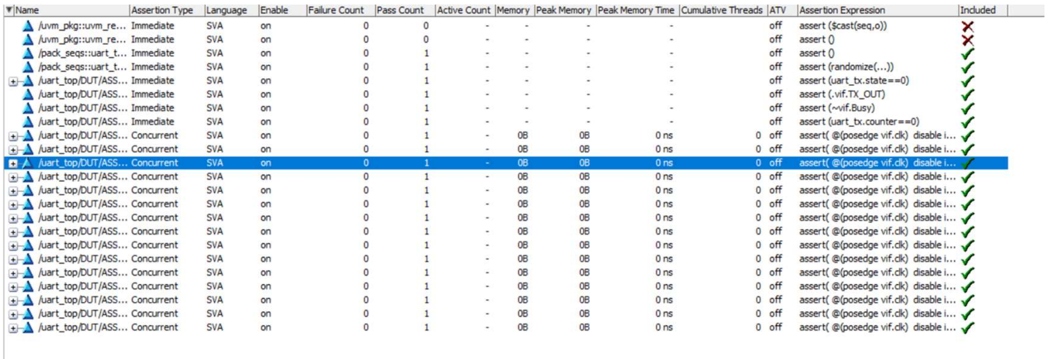


STOP to IDLE transition

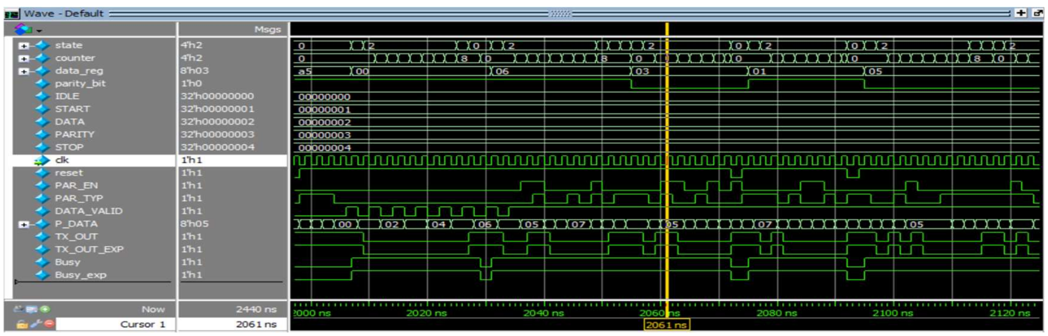
```
// 15. STOP to IDLE transition
property stop_to_idle;
  @(posedge vif.clk) disable iff (!vif.reset)
  (uart_tx.state == vif.STOP) |=>
  (vif.TX_OUT == 1 && uart_tx.state == vif.IDLE);
endproperty
p15: assert property(stop_to_idle) else $error("Should go to IDLE after STOP.");
c15: cover property(stop_to_idle);

endmodule
```

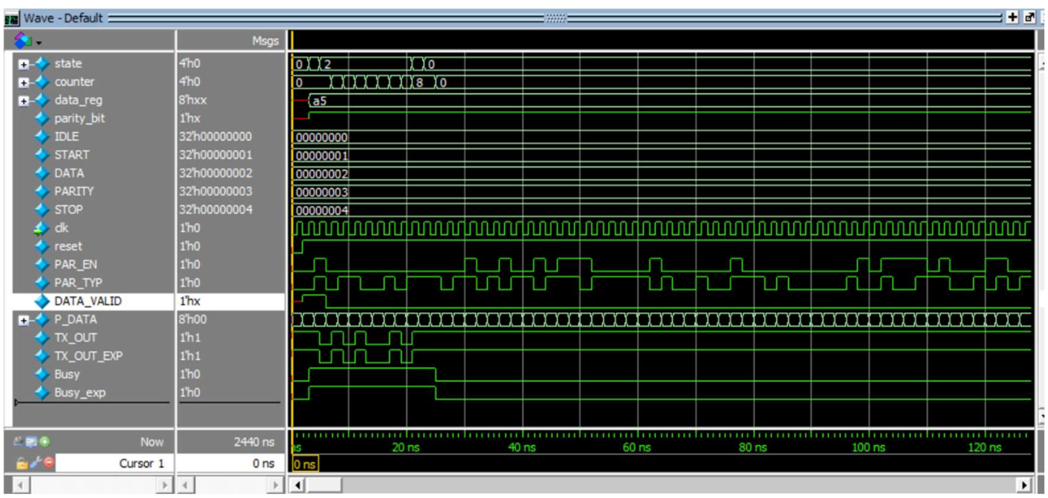


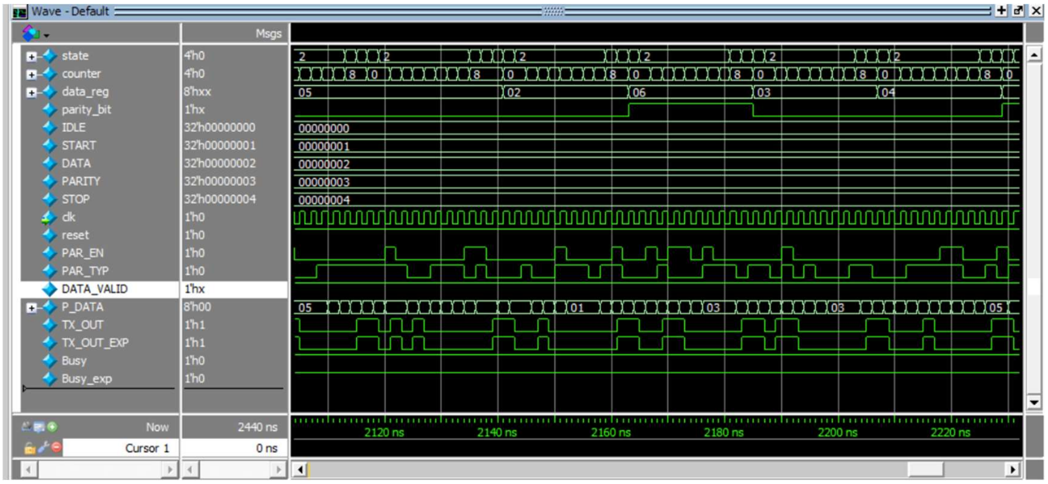
Simulation :

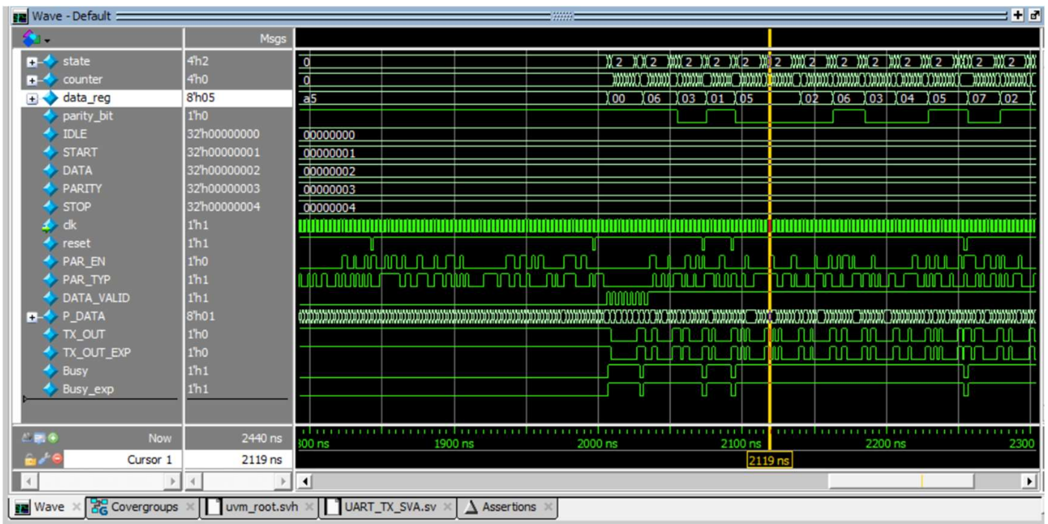


Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_re...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_re...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ()	✗
/pack_seqs::uart_t...	Immediate	SVA	on	0	1	-	-	-	-		off	assert ()	✓
/pack_seqs::uart_t...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/uart_top/DUT/ASS...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (uart_tx.state==0)	✓
/uart_top/DUT/ASS...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (.vif.TX_OUT)	✓
/uart_top/DUT/ASS...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~vif.Busy)	✓
/uart_top/DUT/ASS...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (uart_tx.counter==0)	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓
/uart_top/DUT/ASS...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge vif.clk) disable i...	✓







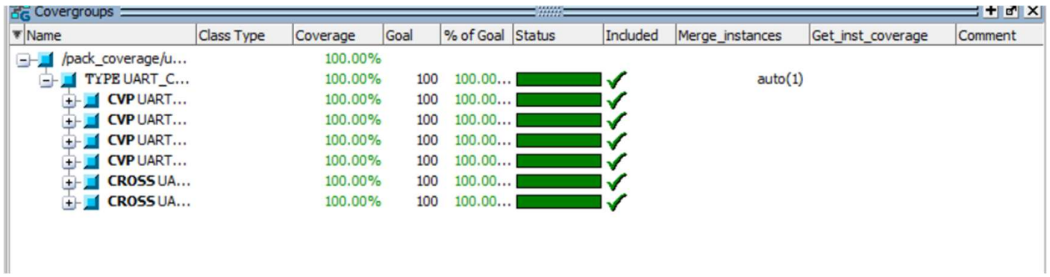



```
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test test...
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(51) @ 0: uvm_test_top [run_phase] Reset sequence started
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(53) @ 2: uvm_test_top [run_phase] Reset sequence finished
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(55) @ 2: uvm_test_top [run_phase] noparity_sequence  started
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(57) @ 4: uvm_test_top [run_phase] noparity_sequence  finished
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(59) @ 4: uvm_test_top [run_phase] tx_parity_sequence  started
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(61) @ 6: uvm_test_top [run_phase] tx_parity_sequence  finished
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(64) @ 6: uvm_test_top [run_phase] Random  sequence started
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(66) @ 2004: uvm_test_top [run_phase] Random  sequence finished
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(69) @ 2004: uvm_test_top [run_phase] Random  sequence started
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_TEST_pkg.sv(71) @ 2440: uvm_test_top [run_phase] Random  sequence finished
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 2440: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# ** Warning: (vsim-PLI-3070) $sformatf : Too many arguments.
#    Time: 2440 ns  Iteration: 43  Process: /uvm_pkg::uvm_phase::m_run_phases/#FORK#1847_7ff4f78e580 File: D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_SCOREBOARD_pkg.sv Line: 53
# UVM_INFO D:/IEEE Verifiction/Final Project/UART_TX_UVM/UART_TX_files_UVM/UART_TX_SCOREBOARD_pkg.sv(53) @ 2440: uvm_test_top.env.sb [SB_REPORT]
# --- UA[illegible]N[illegible][p\$ [illegible]%$§¿ÒÝÜßá¤»"K         0       1220
#
```

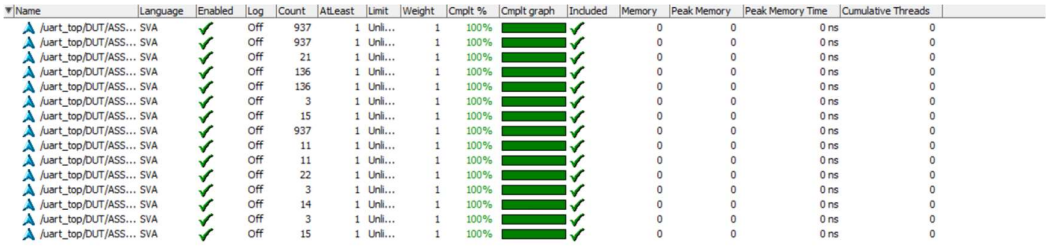
transcript snippets:

```
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   15
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SB_REPORT]     1
# [TEST_DONE]     1
# [run_phase]    10
# ** Note: $finish    : C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 2440 ns  Iteration: 61  Instance: /uart_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
# Causality operation skipped due to absence of debug database file

VSIM 5>
```

Coverage directives:

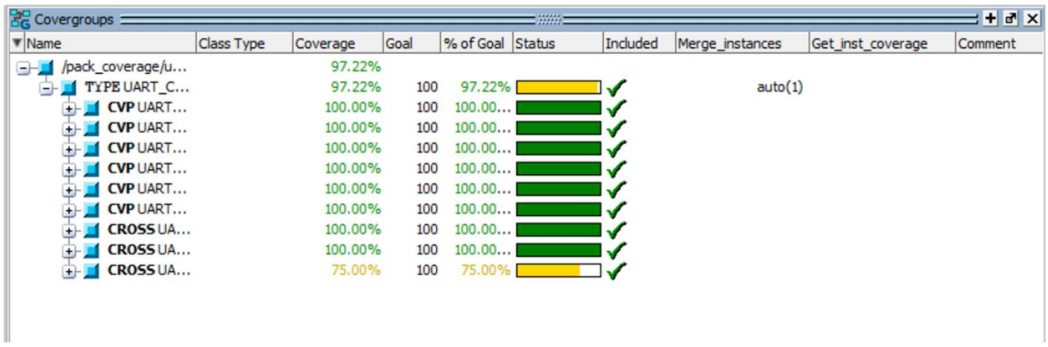


Code_coverage :

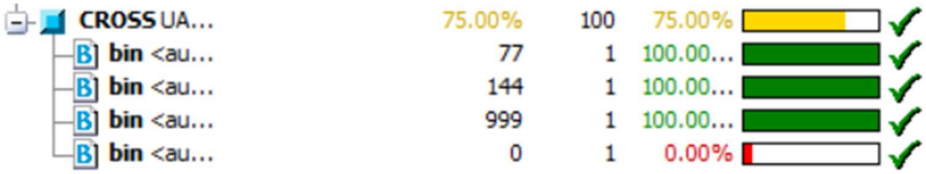


note when we add :

```
cross_TX_OUT_BUSY     : cross seq_item_cov.TX_OUT, seq_item_cov.Busy;
```



All cases of two autputs is not valid :




```
================================================================================
=== Instance: /uart_top/DUT/ASSERTION
=== Design Unit: work.uart_tx_assertions
================================================================================

Directive Coverage:
    Directives                      15        15         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------
/uart_top/DUT/ASSERTION/c1               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(20)
                                                                           937 Covered
/uart_top/DUT/ASSERTION/c2               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(28)
                                                                           937 Covered
/uart_top/DUT/ASSERTION/c3               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(36)
                                                                            21 Covered
/uart_top/DUT/ASSERTION/c4               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(44)
                                                                           136 Covered
/uart_top/DUT/ASSERTION/c5               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(54)
                                                                           136 Covered
/uart_top/DUT/ASSERTION/c6               uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(62)
```

```
--------------------------------------------------------------------------------

Covergroup Coverage:
    Covergroups                     1        na        na    97.22%
        Coverpoints/Crosses         9        na        na        na
            Covergroup Bins        41        40         1    97.56%
----------------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------------------------------
 TYPE /pack_coverage/uart_coverage/UART_CG         97.22%        100          -    Uncovered
    covered/total bins:                               40         41          -
    missing/total bins:                                1         41          -
    % Hit:                                        97.56%        100          -
    Coverpoint cp_PAR_EN                         100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  947          1          -    Covered
        bin auto[1]                                  273          1          -    Covered
    Coverpoint cp_PAR_TYPE                       100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  608          1          -    Covered
        bin auto[1]                                  612          1          -    Covered
    Coverpoint cp_P_VALID                        100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                 1007          1          -    Covered
        bin auto[1]                                  212          1          -    Covered
    Coverpoint cp_P_DATA                         100.00%        100          -    Covered
        covered/total bins:                            8          8          -
        missing/total bins:                            0          8          -
        % Hit:                                   100.00%        100          -
```

```
TOTAL COVERGROUP COVERAGE: 97.22%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/uart_top/DUT/ASSERTION/c1              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(20)
                                                                           937 Covered
/uart_top/DUT/ASSERTION/c2              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(28)
                                                                           937 Covered
/uart_top/DUT/ASSERTION/c3              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(36)
                                                                            21 Covered
/uart_top/DUT/ASSERTION/c4              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(44)
                                                                           136 Covered
/uart_top/DUT/ASSERTION/c5              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(54)
                                                                           136 Covered
/uart_top/DUT/ASSERTION/c6              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(62)
                                                                             3 Covered
/uart_top/DUT/ASSERTION/c7              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(70)
                                                                            15 Covered
/uart_top/DUT/ASSERTION/c8              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(78)
                                                                           937 Covered
/uart_top/DUT/ASSERTION/c9              uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(89)
                                                                            11 Covered
/uart_top/DUT/ASSERTION/c10             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(100)
                                                                            11 Covered
/uart_top/DUT/ASSERTION/c11             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(108)
                                                                            22 Covered
/uart_top/DUT/ASSERTION/c12             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(117)
                                                                             3 Covered
/uart_top/DUT/ASSERTION/c13             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(126)
                                                                            14 Covered
/uart_top/DUT/ASSERTION/c14             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(135)
                                                                             3 Covered
/uart_top/DUT/ASSERTION/c15             uart_tx_assertions Verilog  SVA  UART_TX_SVA.sv(144)
                                                                            15 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 15

Total Coverage By Instance (filtered view): 98.61%
```

```
Coverage Report by instance with details

================================================================================
=== Instance: /uart_top/R_if
=== Design Unit: work.uart_tx_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         34        34         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /uart_top/R_if --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                          Busy           1           1      100.00
                                      Busy_exp           1           1      100.00
                                    DATA_VALID           1           1      100.00
                                        PAR_EN           1           1      100.00
                                       PAR_TYP           1           1      100.00
                                    P_DATA[0-7]          1           1      100.00
                                        TX_OUT           1           1      100.00
                                    TX_OUT_EXP           1           1      100.00
                                           clk           1           1      100.00
                                         reset           1           1      100.00

Total Node Count     =         17
Toggled Node Count   =         17
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (34 of 34 bins)
```

```
================================================================================

Assertion Coverage:
    Assertions                      19        19         0   100.00%
--------------------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------------------
/uart_top/DUT/ASSERTION/a_assert
                     UART_TX_SVA.sv(6)                   0          1
/uart_top/DUT/ASSERTION/b_assert
                     UART_TX_SVA.sv(7)                   0          1
/uart_top/DUT/ASSERTION/c_assert
                     UART_TX_SVA.sv(8)                   0          1
/uart_top/DUT/ASSERTION/d_assert
                     UART_TX_SVA.sv(9)                   0          1
```

```
UART_TX_SVA.sv(143)                                0          1
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         2         2         0   100.00%

================================Branch Details================================

Branch Coverage for instance /uart_top/DUT/ASSERTION

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File UART_TX_SVA.sv
-----------------------------------IF Branch-----------------------------------
    5                                        299     Count coming in to IF
    5              1                          41       if (!vif.reset) begin
                                             258     All False Count
Branch totals: 2 hits of 2 branches = 100.00%



Directive Coverage:
    Directives                      15        15         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                   Design Design   Lang File(Line)      Hits Status
```

```
                                                                                          15 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       1         1         0   100.00%

================================Statement Details================================

Statement Coverage for instance /uart_top/DUT/ASSERTION --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File UART_TX_SVA.sv
    1                                                module uart_tx_assertions (uart_tx_if.DUT vi
    2
    3                                                // Final reset checks after reset is deasser
    4              1                           299     always_comb begin


================================================================================
=== Instance: /uart_top/DUT
=== Design Unit: work.uart_tx
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        16        15         1    93.75%

================================Branch Details================================

Branch Coverage for instance /uart_top/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File UART_TX.sv
```

```
Expression Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Expressions                      3         3         0   100.00%

================================Expression Details================================

Expression Coverage for instance /uart_top/DUT --

  File UART_TX.sv
-----------Focused Expression View (Bimodal)------------
Line       23 Item    1  (~R_if.PAR_TYP? ~^R_if.P_DATA: ^R_if.P_DATA)
Expression totals: 3 of 3 input terms covered = 100.00%

       Input Term   Covered  Reason for no coverage                 Hint
      -----------  --------  -------------------------------------  --------------
     R_if.PAR_TYP         Y
    ~^R_if.P_DATA         Y
     ^R_if.P_DATA         Y

    Rows:   Hits(->0)   Hits(->1)  FEC Target            Non-masking condition(s)

---------  ----------  ----------  --------------------  -------------------------
 Row   1:           0           1  R_if.PAR_TYP_0        -
 Row   2:           1           0  R_if.PAR_TYP_1        -
 Row   3:           1           0  ~^R_if.P_DATA_0       ~R_if.PAR_TYP
 Row   4:           0           1  ~^R_if.P_DATA_1       ~R_if.PAR_TYP
 Row   5:           1           0  ^R_if.P_DATA_0        R_if.PAR_TYP
 Row   6:           0           1  ^R_if.P_DATA_1        R_if.PAR_TYP
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      22        22         0   100.00%

================================Statement Details================================

Statement Coverage for instance /uart_top/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File UART_TX.sv
    1                                                module uart_tx (uart_tx_if.DUT R_if);
    2
    3
```

```
Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Conditions                       1          1         0   100.00%

================================Condition Details================================

Condition Coverage for instance /uart_top/GO --

  File UART_TX_GOLDEN.sv
----------------Focused Condition View-------------------
Line       37 Item    1  (counter_g == 7)
Condition totals: 1 of 1 input term covered = 100.00%

        Input Term   Covered  Reason for no coverage   Hint
       -----------  --------  ----------------------  --------------
  (counter_g == 7)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (counter_g == 7)_0    -
  Row   2:          1  (counter_g == 7)_1    -


Expression Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Expressions                      3          3         0   100.00%
```

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     22        22         0   100.00%

================================Statement Details================================

Statement Coverage for instance /uart_top/GO --

    Line         Item                    Count     Source
    ----         ----                    -----     ------
  File UART_TX_GOLDEN.sv
    1                                              module uart_tx_golden (uart_tx_if.GOLDEN R_if);
    2
    3
    4                                                reg [3:0] state_g = R_if.IDLE;
    5                                                reg [3:0] counter_g = 0;
    6                                                reg [7:0] data_reg_g;
    7                                                reg parity_bit_g;
    8
    9              1                    1254        always @(posedge R_if.clk or negedge R_if.reset) begin
    10                                                 if (!R_if.reset) begin
    11             1                      69            state_g <= R_if.IDLE;
    12             1                      69            R_if.TX_OUT_EXP <= 1'b1;
    13             1                      69            R_if.Busy_exp <= 1'b0;
    14             1                      69            counter_g <= 0;
    15                                                 end else begin
    16                                                   case (state_g)
    17                                                     R_if.IDLE: begin
    18             1                     986               R_if.TX_OUT_EXP <= 1'b1;
    19             1                     986               R_if.Busy_exp <= 1'b0;
    20             1                     986             counter_g <=0 ;
    21                                                     if (R_if.DATA_VALID) begin
    22             1                      22                 data_reg_g <= R_if.P_DATA;
    23             1                      22                 parity_bit_g <= (R_if.PAR_TYP == 0 ) ? ~^R_if.P_DATA : ^R_if.P_DATA ;
    24             1                      22                 state_g <= R_if.START;
    25             1                      22                 R_if.Busy_exp <= 1'b1;
    26                                                     end
    27                                                       end
    28                                                     R_if.START: begin
    29             1                      22               R_if.TX_OUT_EXP <= 1'b0;
    30             1                      22               state_g <= R_if.DATA;
    31             1                      22               counter_g <= 0 ;
    32                                                     end
```

تم بحمد الله

﴿ وَأَن لَّيْسَ لِلْإِنسَانِ إِلَّا مَا سَعَىٰ﴾

[النجم: 39]

صدق الله العظيم