SERIAL COMMUNICATION SYSTEM USING UART

RTL TO GDS II

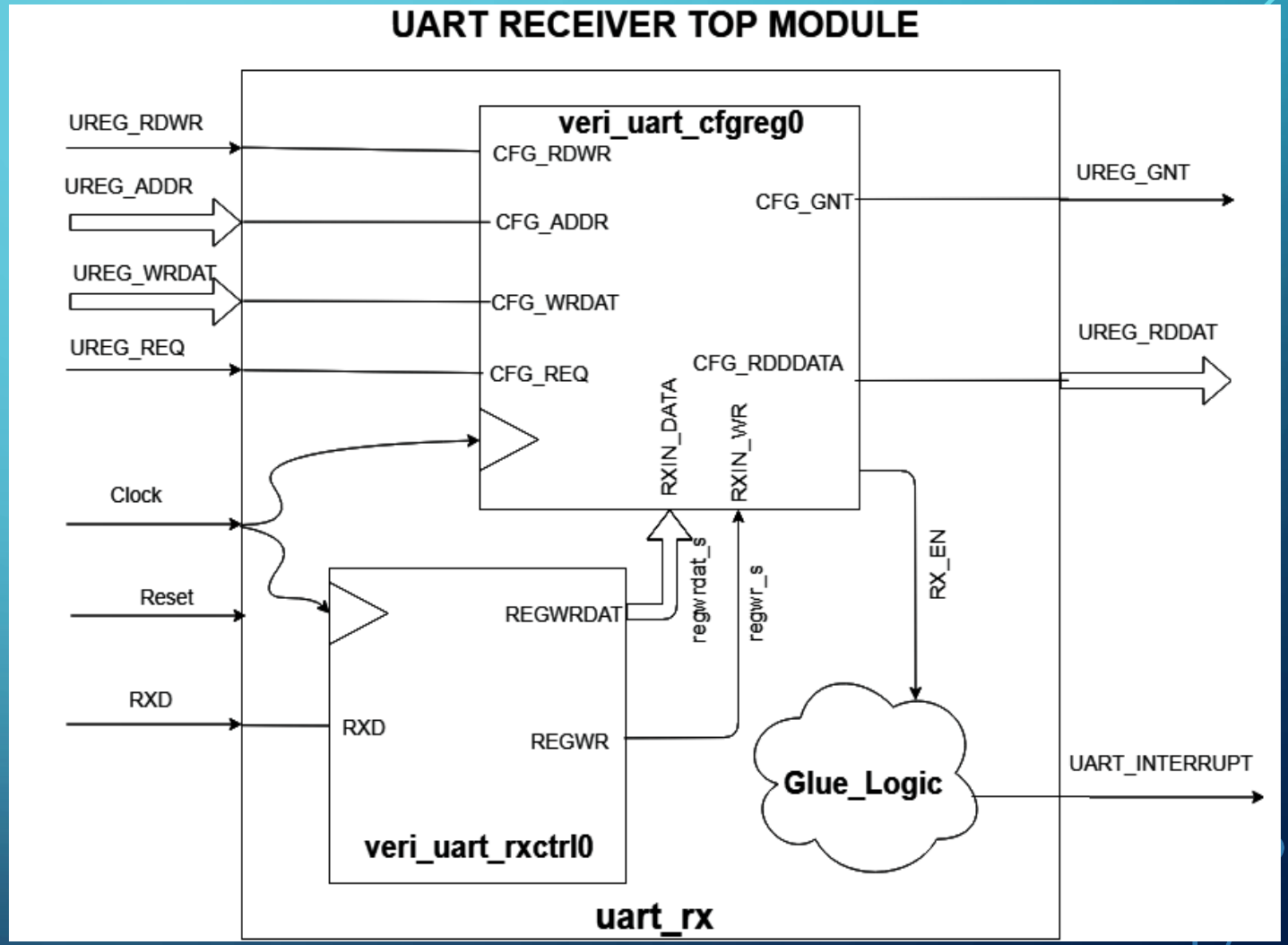
Presented By: Vipin Jasoria
Guided By: DR. George Engel
Department of Electrical Engineering
Southern Illinois University Edwardsville

TODAY'S AGENDA

- UART RECEIVER MODULE
- VERIFICATION PLAN
- TEST BENCH COMPONENTS
- SIMULATION RESULTS
- UART TX-RX MODULE
- COMMUNICATION SIMULATION
- SYNTHESIS
- PLACEMENT & ROUTING

RECEIVER MODULE

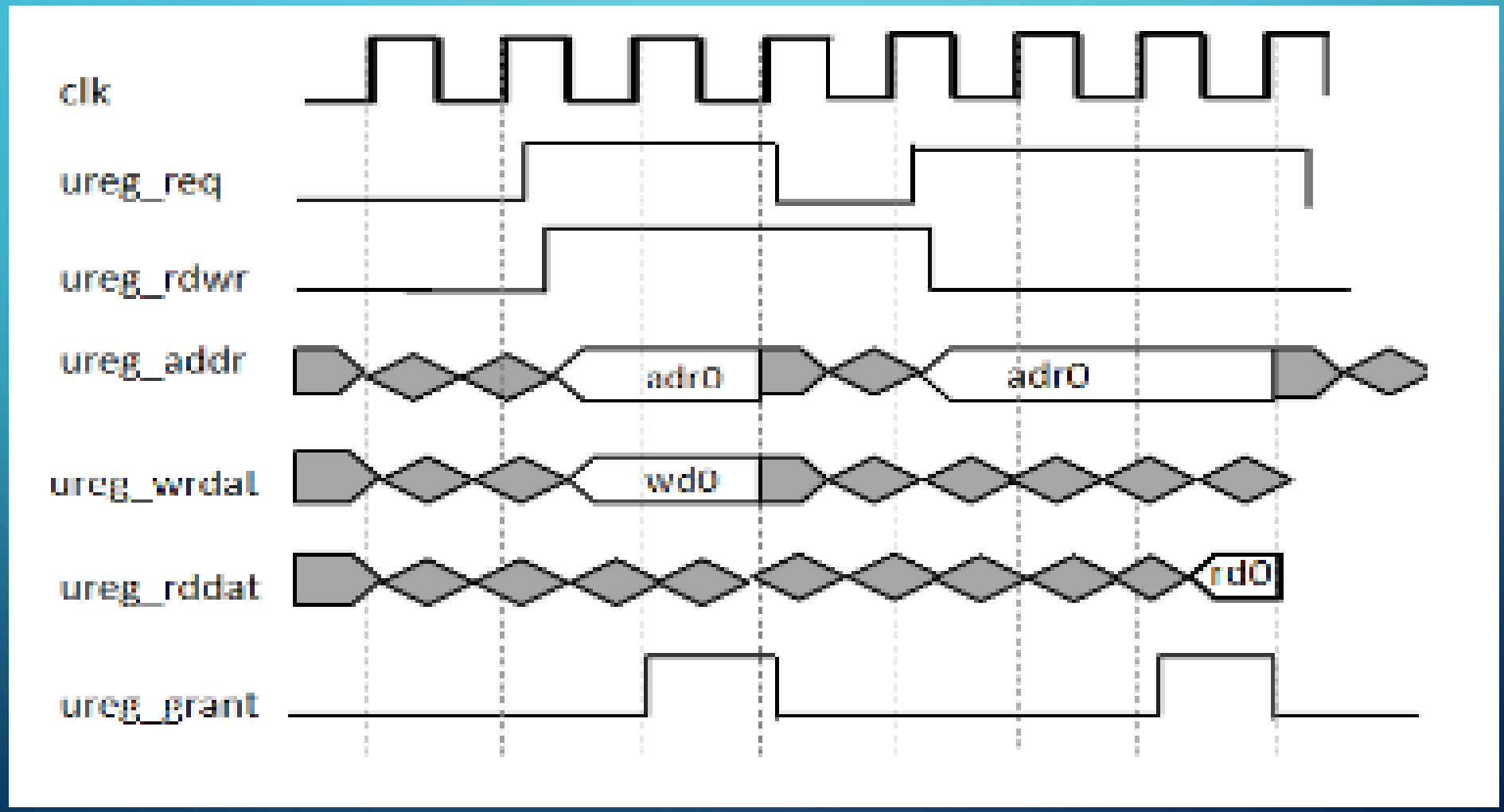
- UART Controller DUT receives serial data, does serial to parallel conversion and sends it to the host.
- DUT has two interfaces
  - Interface to receive the serial data.
  - Interface to interact with the host.
- Configuration Registers in DUT
  - Two Configuration registers
  - Core Control Register at Addr :00.
    - Recevie Enable – 1 bit RW [1]
  - UART receive data register at Addr:01.
    - RX Received Data – 8 bits Read Only.



DUT OPERATION DETAILS

- DUT RECEIVES DATA SERIALLY ACCORDING TO UART PROTOCOL.
- ONCE THE DUT RECEIVES A BYTE OF DATA, DUT STORES THE BYTE IN A REGISTER AND RAISES AN INTERRUPT TO THE HOST THROUGH THE INTERRUPT OUTPUT PIN.
- HOST READS UART DATA REGISTERED THROUGH THE HOST INTERFACE.
- INTERRUPT SIGNAL DEASSERTED ON THE REGISTER READ.
- DUT SERIAL INTERFACE
  - DUT receive data through our RXD pin serially one bit per cycle.
  - The first low cycle indicates a start bit followed by 8 bit data and 2 stop bits.
- DUT HOST INTERFACE
  - When host wants to read or write a register, host asserts request and gives the address of the register.
  - If it's a write request, Host gives Write data along with address and DUT asserts Grant, for read request by Host, DUT gives read data along with grant.

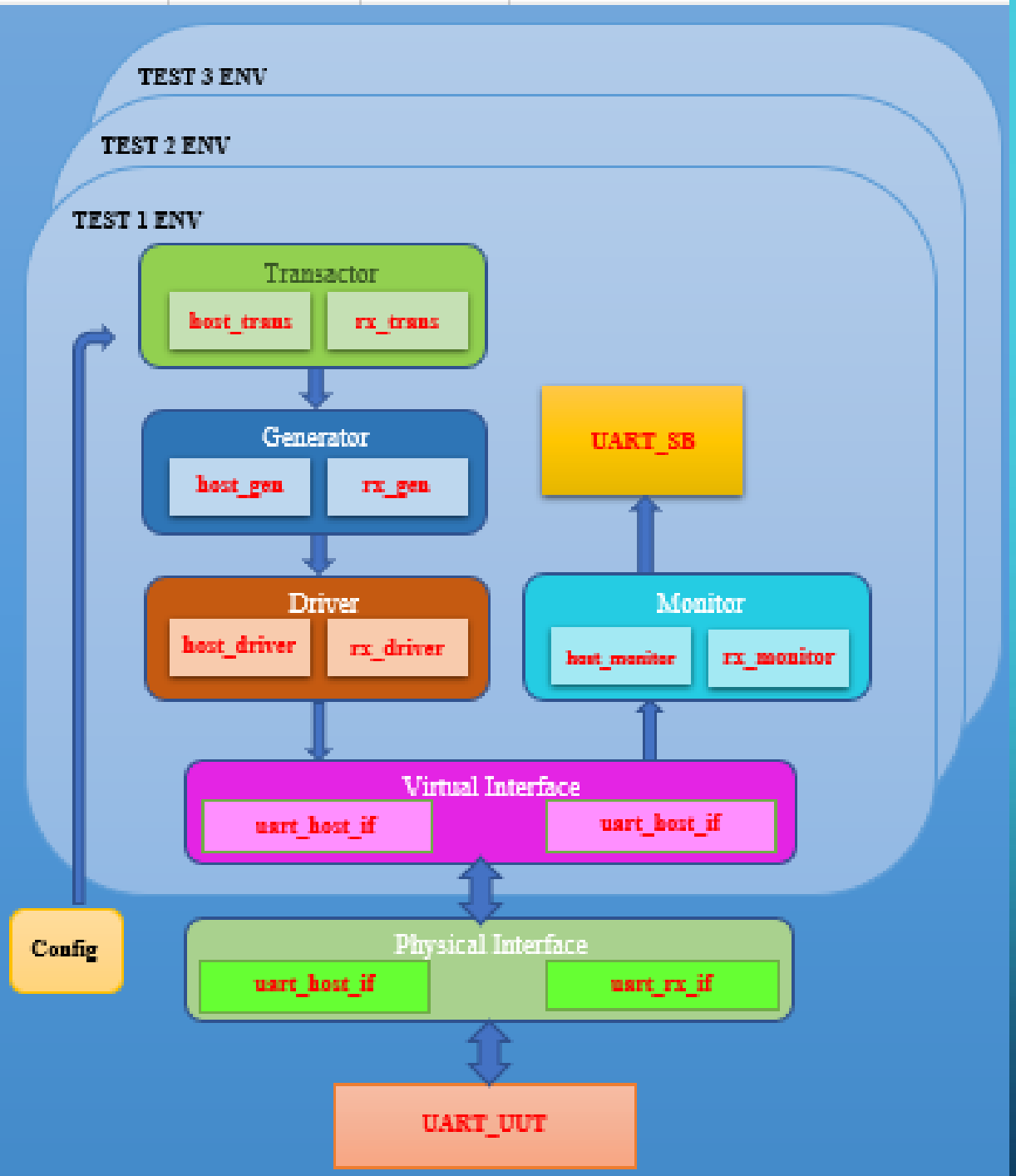
DUT HOST INTERFACE SIGNALS



Source: VerfastTech

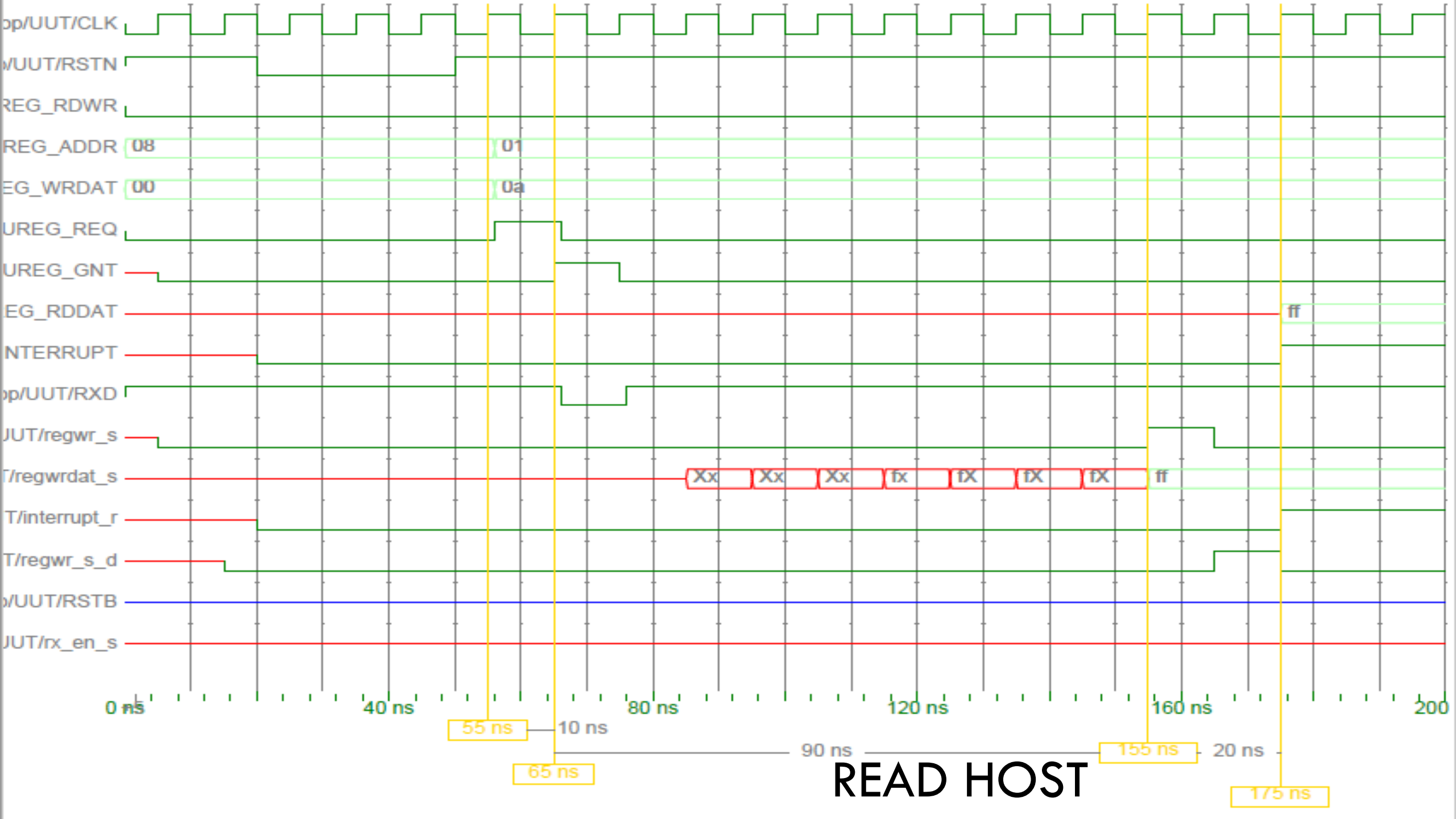
UART Verification Plan

S No.	Test Name	Directed/Random	Iterations	Description
1	RESET_TEST	Directed	1	Verifying if both the registers getting reset by RSTN = 0.
2	HOST_WITE_TEST	Directed	2	Checking the Write Functionality of Host by Wrtting into it. UREG_RWR=1.
3	HOST_READ_TEST	Directed	2	Checking the Read Functionality of Host by Reading from Host. UREG_RWR=0.
4	ADRESS_TEST	Random	4	Checking if both the Regiters are accessble. UREG_ADDR = RANDOM.
5	REQ_TEST	Directed	1	Check for Request Functionality UREG_REQ=0.
6	RANDOM_TEST	Random	10	Checking for Random Read/Write and other features [All Inputs Contraint Random]
7	RXD_TEST	Directed	1	Giving serial data and checking the registers getting written. RXD = 12.
8	RXD_TEST_RNDM	Random	2	Writing serial data back to back. RXD = random
9	ALL_ZERO	Directed	1	ALL Inputs 0.
10	ALL_ONCES	Directed	1	ALL Inputs 1.
11	ALTERNATE	Directed	2	Alternate 0s and 1s

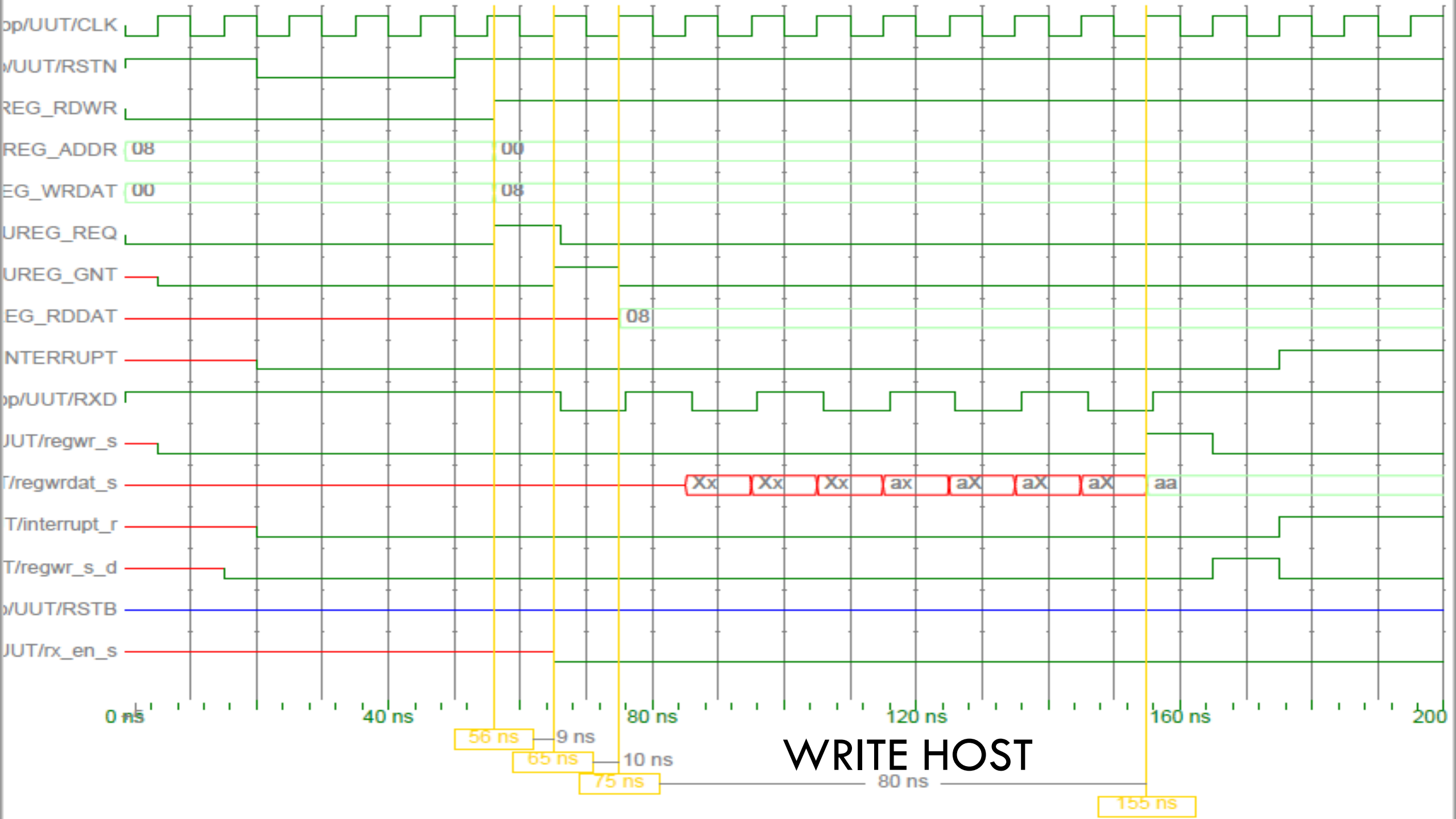


TEST BENCH COMPONENTS

- **Interface**
  - Group of Signals along with directions and timings.
- **Virtual Interface**
  - Connection between Static and Virtual Components.
- **Configuration**
  - Providing configurability to the test bench.
- **Transactor**
  - Prints & deep copies the signals of Interfaces.
- **Generator**
  - Generates various test cases to be tested on DUT.
- **Driver**
  - Drives or gives inputs as per the protocol to DUT.
- **Monitor**
  - Collects signals from DUT to send to Scoreboard
- **ScoreBoard**
  - Compares DUT and Reference model outputs.
- **DUT**
  - The Hardware unit we want to perform verification at.

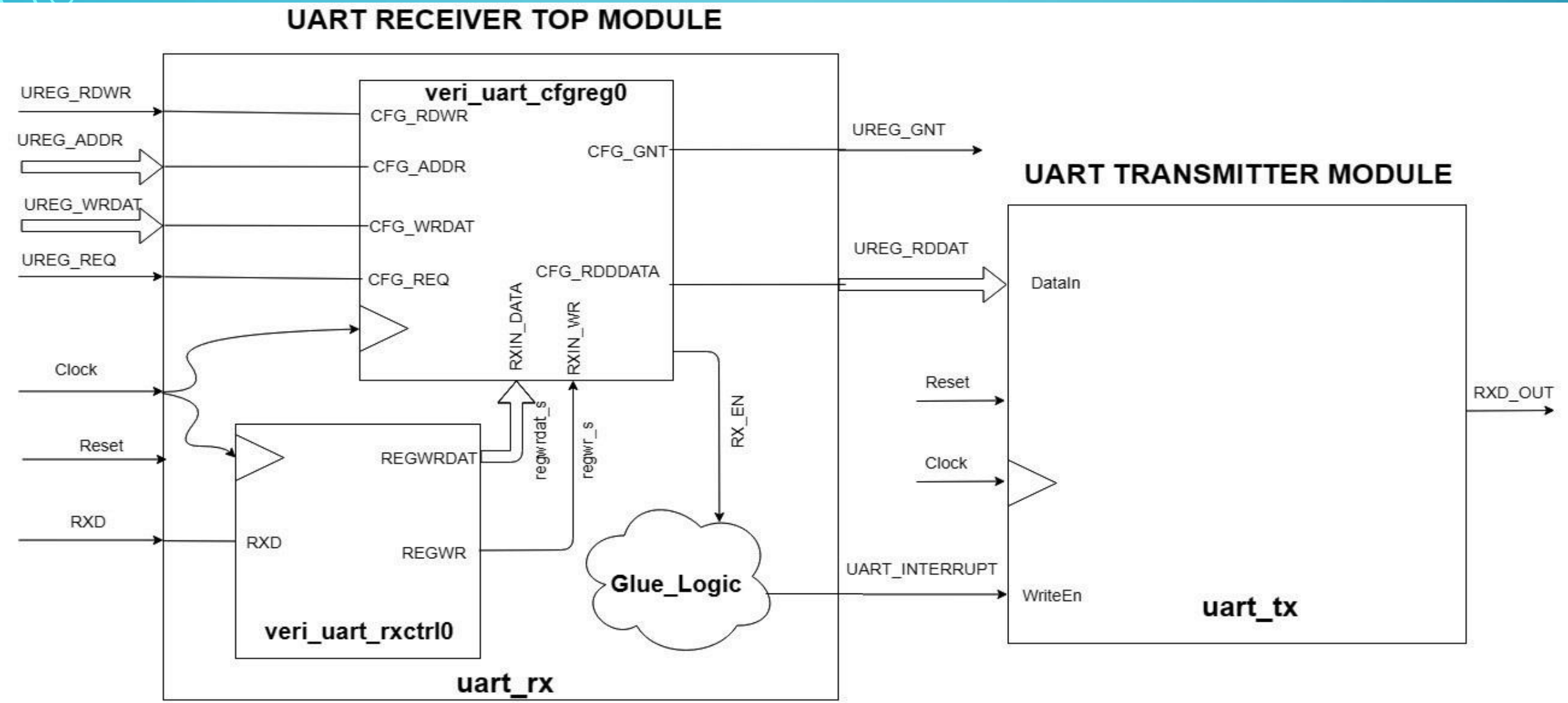

op/UUT/CLK
/UUT/RSTN
REG_RDWR
REG_ADDR
08
01
EG_WRDAT
00
0a
UREG_REQ
UREG_GNT
EG_RDDAT
ff
NTERRUPT
op/UUT/RXD
JUT/regwr_s
T/regwrdat_s
Xx
Xx
Xx
fx
fX
fX
fX
ff
T/interrupt_r
T/regwr_s_d
/UUT/RSTB
JUT/rx_en_s
0 ns
40 ns
80 ns
120 ns
160 ns
200
55 ns
10 ns
90 ns
155 ns
20 ns
65 ns
175 ns

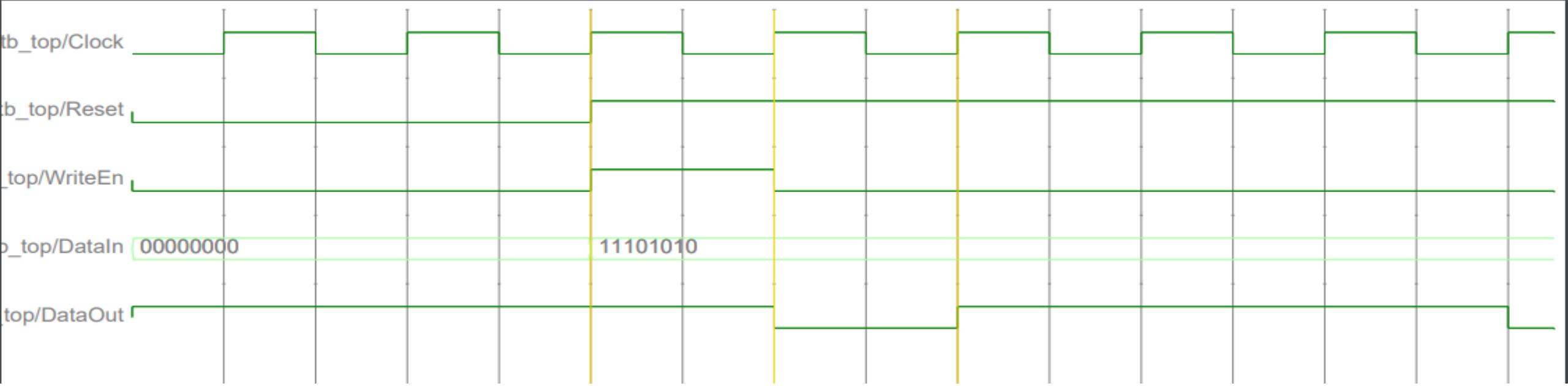
READ HOST

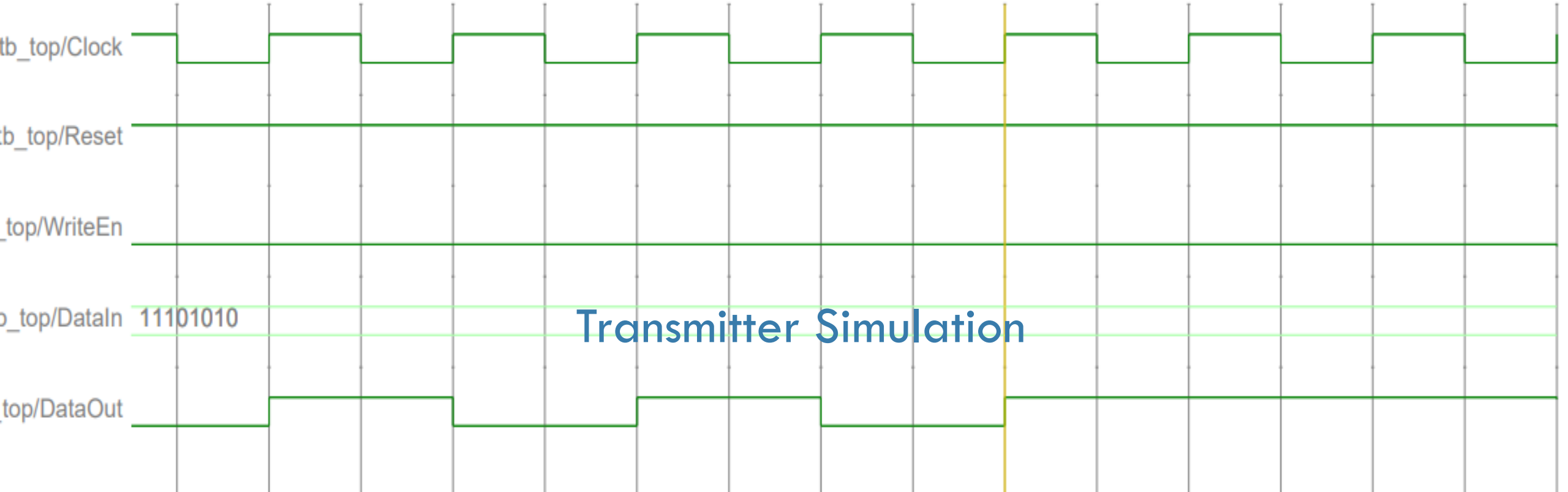

op/UUT/CLK
/UUT/RSTN
REG_RDWR
REG_ADDR
08
00
EG_WRDAT
00
08
UREG_REQ
UREG_GNT
EG_RDDAT
08
NTERRUPT
op/UUT/RXD
JUT/regwr_s
T/regwrdat_s
Xx
Xx
Xx
ax
aX
aX
aX
aa
T/interrupt_r
T/regwr_s_d
/UUT/RSTB
JUT/rx_en_s
0 ns
40 ns
80 ns
120 ns
160 ns
200
56 ns
9 ns
65 ns
10 ns
75 ns
80 ns
155 ns
WRITE HOST


op/UUT/CLK
/UUT/RSTN
REG_RDWR
REG_ADDR
08
00
EG_WRDAT
00
2b
UREG_REQ
UREG_GNT
EG_RDDAT
NTERRUPT
op/UUT/RXD
UUT/regwr_s
T/regwrdat_s
Xx
Xx
Xx
5x
5X
5X
5X
55
T/interrupt_r
T/regwr_s_d
/UUT/RSTB
UUT/rx_en_s
0 ns
40 ns
80 ns
120 ns
160 ns
200
60 ns
20 ns
85 ns
70 ns
9 ns
65 ns
56 ns
155 ns
ADDRESS TEST

UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

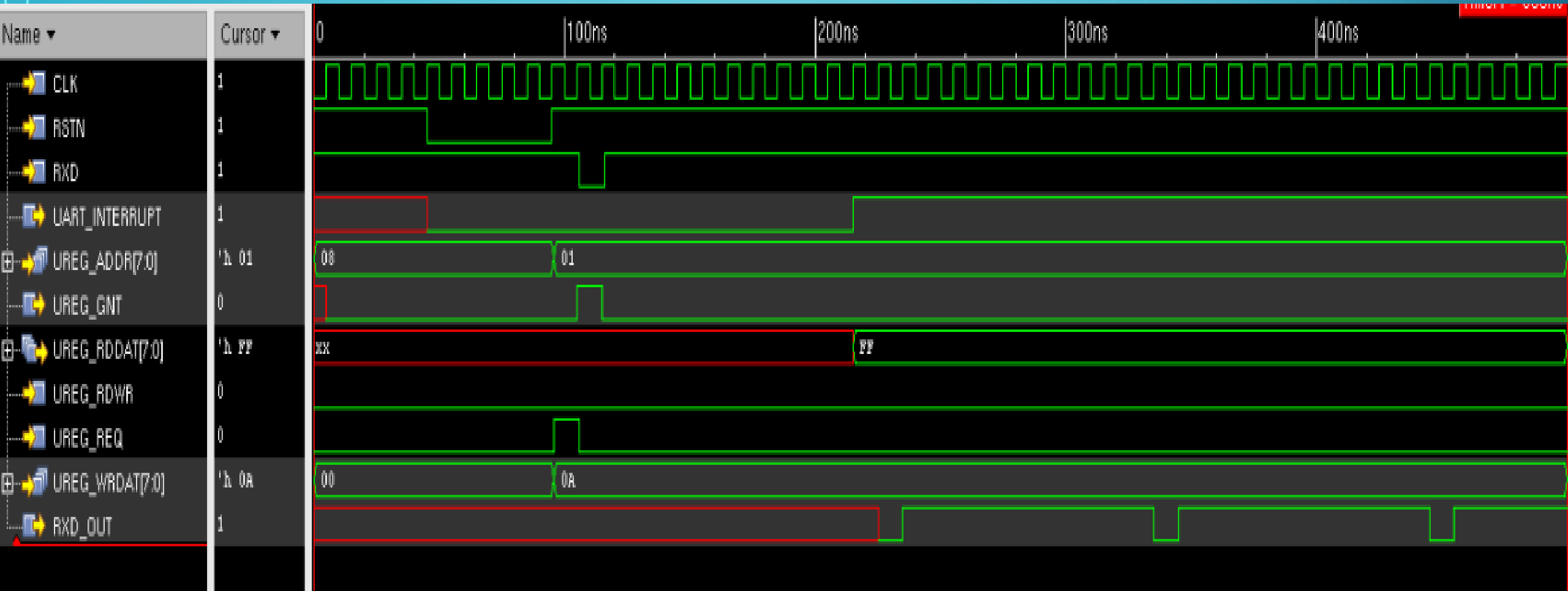




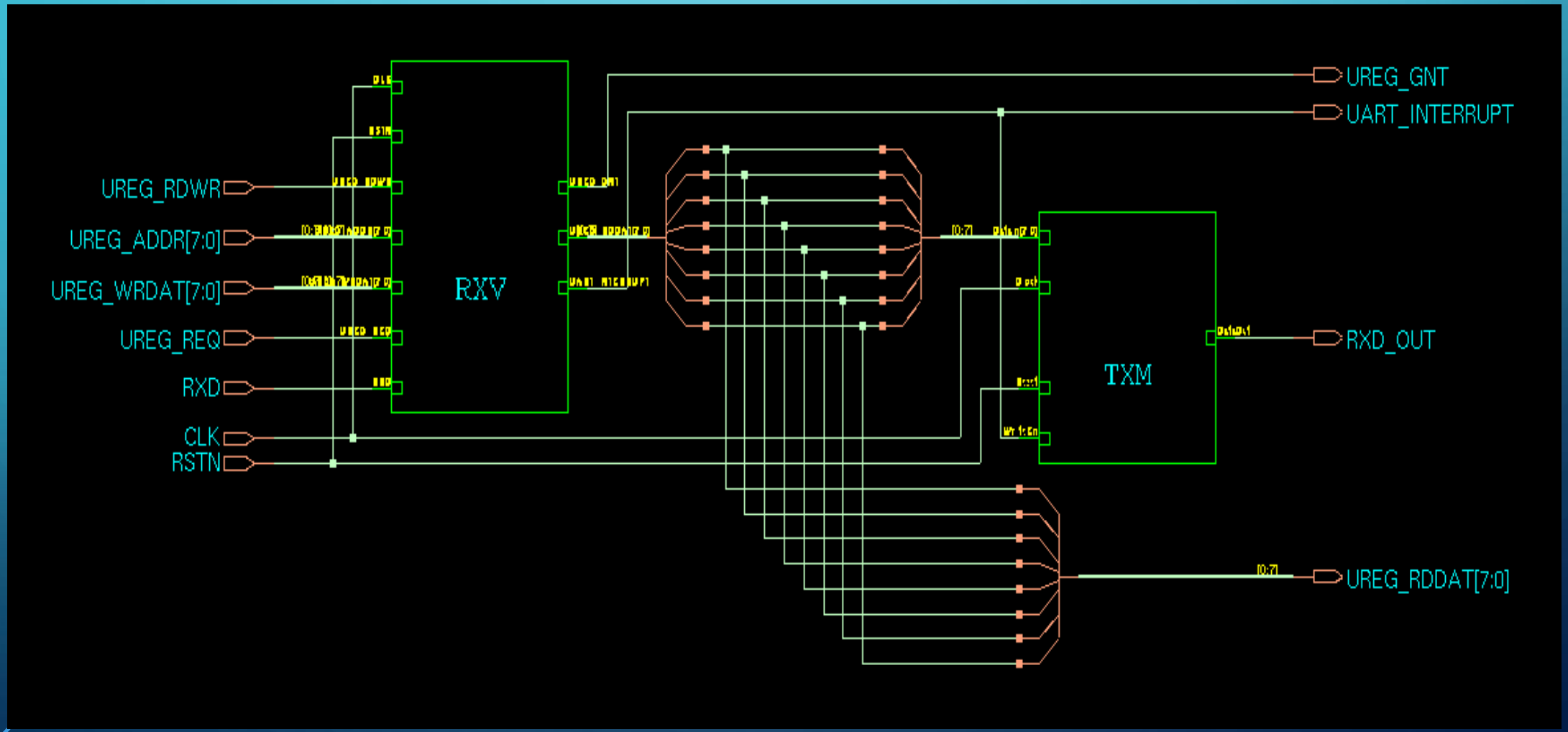


Transmitter Simulation

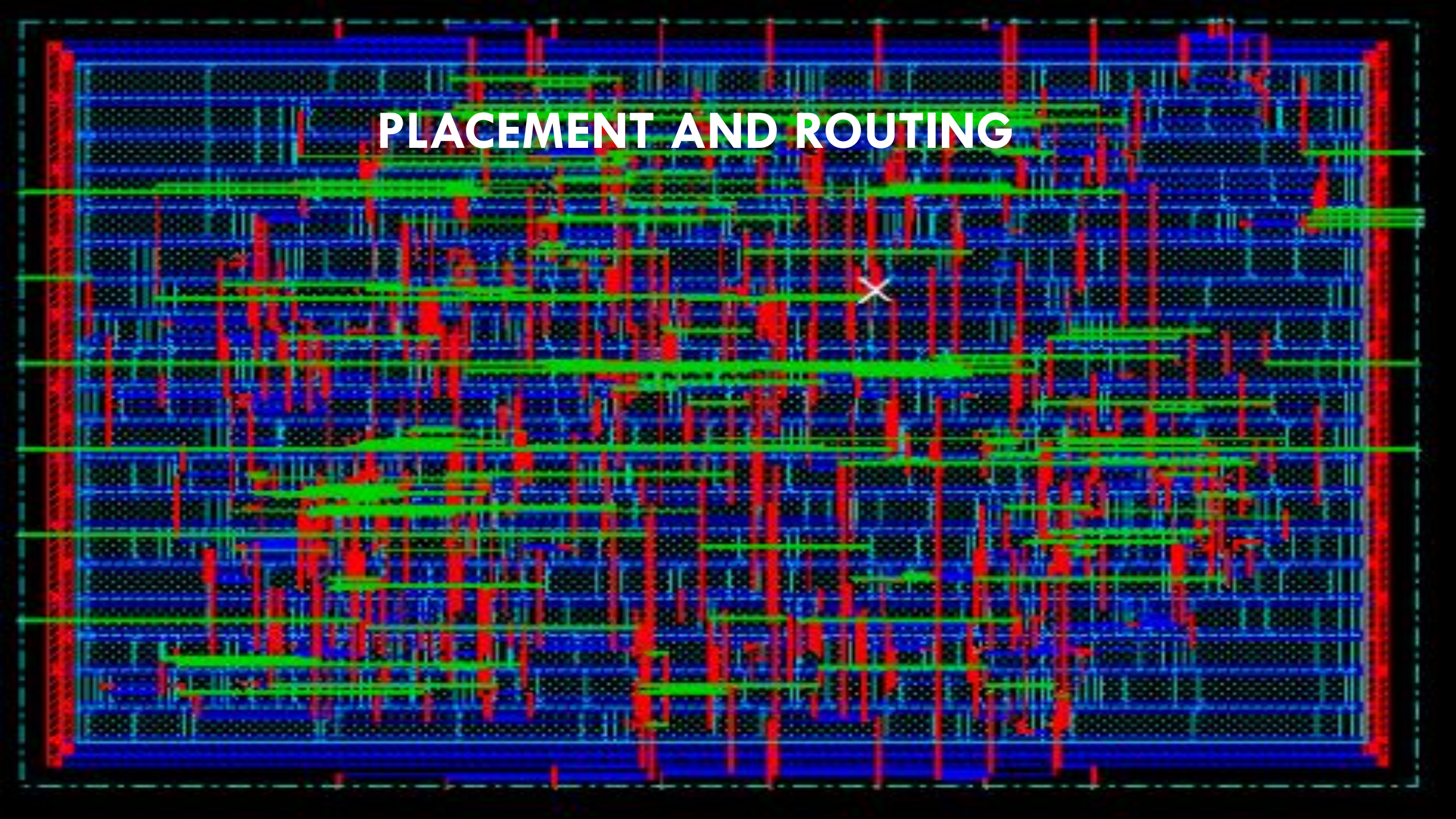
UART COMMUNICATION SYSTEM SIMULATION



SYNTHESIS RESULTS



PLACEMENT AND ROUTING



```
Reported timing to dir timingReports
Total CPU time: 0.46 sec
Total Real time: 1.0 sec
Total Memory Usage: 623.222656 Mbytes
 *** Starting Verify Geometry (MEM: 623.2) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 18000
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells           :  0 Viols.
  VERIFY GEOMETRY ..... SameNet         :  0 Viols.
  VERIFY GEOMETRY ..... Wiring          :  0 Viols.
  VERIFY GEOMETRY ..... Antenna         :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.6  MEM: 1.0M)
```

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Sun Aug  4 15:55:37 2019

Design Name: uart
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (398.5750, 277.6000)
Error Limit = 1000; Warning Limit = 50
Check all nets
Time Elapsed: 0:00:00.0

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Sun Aug  4 15:55:37 2019
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.1  MEM: 0.004M)
```

SUMMARY

THANKS !!

QUESTIONS ???

```
// Copyright Vipin Jasoria
// ONLY FOR REFERENCE
// UART CONFIG Module
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE

class uart_config;

        virtual         uart_rx_if              rx_vif;
        virtual         uart_host_if    host_vif;
        mailbox         #(rx_trans)             gen2dr_mbx1;
        mailbox         #(host_trans)   gen2dr_mbx2;

        static int trans_count = 10;
        static int driver_delay = 0;
                    event RST_DONE;
        string          test_name;
//-----------------------------------------------------
        function new();
        endfunction :new
//-----------------------------------------------------

        function void get_args_from_command_line();
                int no_of_trans;
                int delay;
                void'($value$plusargs("TESTNAME=%0s",test_name));
                void'($value$plusargs("NO_OF_TRANS=%d", no_of_trans));
                void'($value$plusargs("DELAY=%d", delay));
                $display("----------------------------");
                $display("Iterations :: %0d\nDelay :: %0d ", no_of_trans, delay);
                $display("TESTNAME is %s", test_name);
                $display("----------------------------");
                trans_count = no_of_trans;
                driver_delay = delay;

        endfunction : get_args_from_command_line

endclass : uart_config
```

```
//-------------------------------------------------------------
//  HOST GENERATOR MODULE
// Copyright 2019 VipinJasoria
// ONLY FOR REFERENCE
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE
//-------------------------------------------------------------
class host_gen;
        host_trans pkt;
        mailbox #(host_trans) gen2dr_mbx2;
        uart_config cfg;
//-------------------------------------------------------------
        function new(uart_config cfg);
                this.cfg = cfg;
                this.gen2dr_mbx2 = new();
        endfunction
//-------------------------------------------------------------
        function void connect();
                cfg.gen2dr_mbx2 = gen2dr_mbx2;
        endfunction: connect
//-------------------------------------------------------------
//              ALL TEST CASES GOES HERE
//-------------------------------------------------------------
//      virtual function void RESET_TEST;
//              pkt             =       new();
//
//      endfunction: RESET_TEST
//-------------------------------------------------------------
        virtual function void HOST_WRITE_TEST;
                pkt             =       new();
                pkt.UREG_RDWR   =       1;
                pkt.UREG_ADDR   =       0;
                pkt.UREG_WRDAT  =       8;
                pkt.UREG_REQ    =       1;
        endfunction: HOST_WRITE_TEST
//-------------------------------------------------------------
        virtual function void HOST_READ_TEST;
                pkt             =       new();
                pkt.UREG_RDWR   =       0;
                pkt.UREG_ADDR   =       1;
                pkt.UREG_WRDAT  =       10;
                pkt.UREG_REQ    =       1;
        endfunction: HOST_READ_TEST
//-------------------------------------------------------------
```

```
        virtual function void ADDRESS_TEST;
                pkt             =       new();
                pkt.UREG_RDWR   =       $urandom_range(0,1); // Read or Write both
                pkt.UREG_ADDR   =       $urandom_range(0,1); // FOR both the address
                pkt.UREG_WRDAT  =       43;
                pkt.UREG_REQ    =       1;
        endfunction: ADDRESS_TEST

//-------------------------------------------------------------
        virtual function void REQ_TEST;
                pkt             =       new();
                pkt.UREG_RDWR   =       $urandom_range(0,1); // Read or Write both
                pkt.UREG_ADDR   =       $urandom_range(0,1); // FOR both the address
                pkt.UREG_WRDAT  =       43;
                pkt.UREG_REQ    =       0;
        endfunction: REQ_TEST
//-------------------------------------------------------------
        virtual function void RANDOM_TEST;
                pkt             =       new();
                void'(pkt.randomize());
        endfunction: RANDOM_TEST

//-------------------------------------------------------------
//              END TEST CASES
//-------------------------------------------------------------
        virtual task deliver_pkt_to_driver();
                gen2dr_mbx2.put(pkt);
        endtask : deliver_pkt_to_driver
//-------------------------------------------------------------

endclass: host_gen
```

```
//------------------------------------------------
// COPYRIGHT @ 2019 VIPIN JASORIA
// TEST BENCH TOP MODULE
//  UART_RX_TB.SV
// ****FOR REFERENCE ONLY****
//------------------------------------------------


//------------------------------------------------
//              INCLUDED FILES IN THE TB_TOP
//------------------------------------------------
//‘include "design.sv"
‘include "uart_host_if.sv"
‘include "uart_rx_if.sv"
‘include "rx_trans.sv"
‘include "host_trans.sv"
‘include "uart_config.sv"
‘include "rx_gen.sv"
‘include "host_gen.sv"
‘include "rx_driver.sv"
‘include "host_driver.sv"

//------------------------------------------------

//------------------------------------------------
//                      MACROS
//------------------------------------------------
‘define LOW 0
‘define HIGH 1
//------------------------------------------------

module uart_rx_tb;

        reg Clock;
        reg resetn;
        parameter CYCLE = 10;
        parameter RESET_PERIOD = 3;
//------------------------------------------------

        uart_rx_if      uart_rx(Clock, resetn);
        uart_host_if    uart_host(Clock, resetn);

        uart_config     cfg;
```

```
        rx_gen          rxgen;
        host_gen        hostgen;


        rx_driver       rxdriver;
        host_driver     hostdriver;
//-------------------------------------------------
//                      Design Under Test
//-------------------------------------------------
         uart_rx UUT (
                 .CLK           (Clock),
                 .RSTN          (resetn),
                 .UREG_RDWR     (uart_host.UREG_RDWR),
                 .UREG_ADDR     (uart_host.UREG_ADDR),
                 .UREG_WRDAT    (uart_host.UREG_WRDAT),
                 .UREG_REQ      (uart_host.UREG_REQ),
                 .UREG_GNT      (uart_host.UREG_GNT),
                 .UREG_RDDAT    (uart_host.UREG_RDDAT),
                 .UART_INTERRUPT(uart_host.UART_INTERRUPT),
                 .RXD           (uart_rx.RXD)
        );

//-------------------------------------------------
//              INITIAL BLOCK
//-------------------------------------------------

initial begin
        $display("------------------------------------");
        $display("Test Bench TOP Example with Interface");
        $display("------------------------------------");
        build();
        $display("Built Method Executed @ %gns", $time);
        connect();
        $display("Connect Method Executed @ %gns", $time);
        generate_clock();
        $display("Clock Generation Started %gns", $time);
        initialize_dut_input_signals();
        give_reset_to_dut();
        $display("Giving DUT a RESET PULSE @ %gns", $time);
        initiate_stimulus_to_dut();
        $display("initiating Stimulas @ %gns", $time);
        wait_for_sometime();
        $display("Waiting for sometime @ %gns", $time);
        $display("Finish !! OUT OF SIM @ %gns", $time );
```

```
        finish_test();


        #50us $finish;
end
//-------------------------------------------------------
//                      TASKS
//-------------------------------------------------------
        task build();
                build_config_and_do_assignment();
                rxgen           =        new(cfg);
                hostgen         =        new(cfg);
                rxdriver        =        new(cfg);
                hostdriver      =        new(cfg);
        endtask: build
//-------------------------------------------------------
        task build_config_and_do_assignment();
                cfg             =       new();
                cfg.rx_vif      =       uart_rx;
                cfg.host_vif    =       uart_host;
                cfg.get_args_from_command_line();
        endtask : build_config_and_do_assignment
//-------------------------------------------------------
        task connect();
                rxgen.connect();
                hostgen.connect();
                rxdriver.connect();
                hostdriver.connect();
        endtask: connect

//-------------------------------------------------------
        task generate_clock();
                fork
                forever begin
                        Clock = 'LOW;
                        #(CYCLE/2);
                        Clock = 'HIGH;
                        #(CYCLE/2);
                end
                join_none
        endtask : generate_clock
//-------------------------------------------------
        task initialize_dut_input_signals();
                uart_rx.RXD             =       1'b1;
```

```
                uart_host.UREG_RDWR     =       1'b0;
                uart_host.UREG_ADDR     =       8'b0000_1000;
                uart_host.UREG_WRDAT    =       0;
                uart_host.UREG_REQ      =       1'b0;

        endtask : initialize_dut_input_signals
//-------------------------------------------------
        task give_reset_to_dut();
                resetn = 'HIGH;
                repeat(RESET_PERIOD) @(negedge Clock);
                do_reset();
                -> cfg.RST_DONE;
        endtask : give_reset_to_dut
//-------------------------------------------------
        task do_reset();
        resetn = 'LOW;
        repeat(RESET_PERIOD) @(negedge Clock);
        resetn = 'HIGH;
        endtask : do_reset
//-------------------------------------------------
        task initiate_stimulus_to_dut();
                initiate_driver();
                $display("Initiate driver task executed");
                built_test_given_in_command_line();
                $display("Built test cases executed with test_name is %s", cfg.test_name);
        endtask : initiate_stimulus_to_dut
//-------------------------------------------------
        task initiate_driver();
            fork
              rxdriver.run_driver();
              hostdriver.run_driver();
            join_none
        endtask : initiate_driver
//-------------------------------------------------
//                      NAME OF THE TEST CASES
//-------------------------------------------------



//-------------------------------------------------
//                              HOST SIDE
//-------------------------------------------------
//RESET_TEST    HOST_WRITE_TEST HOST_READ_TEST  ADDRESS_TEST
//REQ_TEST      RANDOM_TEST
```

```
//-------------------------------------------------




//-------------------------------------------------
//                      Receiver Side
//-------------------------------------------------
//RXD_TEST      RXD_TEST_RNDM   ALL_ZERO        ALL_ONCES
//ALTERNATE1    ALTERNATE2
//-------------------------------------------------



//-------------------------------------------------
// CONFIGURABILITY Stuff
//-------------------------------------------------
        task built_test_given_in_command_line();
                // void'($value$plusargs("TESTNAME=%0s",test_name));
                // $display(" The Test Case is %s", cfg.test_name);
                case (cfg.test_name)
                        "HOST_WRITE_TEST" : initiate_generator_HOST_WRITE_TEST();
                        "HOST_READ_TEST"  : initiate_generator_HOST_READ_TEST();
                        "ADDRESS_TEST"    : initiate_generator_ADDRESS_TEST();
                        "REQ_TEST"        : initiate_generator_REQ_TEST();
                        "RANDOM_TEST"  : initiate_generator_RANDOM_TEST();
                        default : $display("Please Provide Correct Test Case");
                endcase
        endfunction : built_test_given_in_command_line

//-------------------------------------------------
        task initiate_generator_HOST_WRITE_TEST();
                rxgen.ALL_ONCES();
                hostgen.HOST_WRITE_TEST();
                rxgen.deliver_pkt_to_driver();
                hostgen.deliver_pkt_to_driver();
        endtask : initiate_generator_HOST_WRITE_TEST
//-------------------------------------------------
        task initiate_generator_HOST_READ_TEST();
                rxgen.ALL_ZERO();
                hostgen.HOST_READ_TEST();
                rxgen.deliver_pkt_to_driver();
                hostgen.deliver_pkt_to_driver();
        endtask : initiate_generator_HOST_READ_TEST
```

```
//-------------------------------------------------
        task initiate_generator_ADDRESS_TEST();
                rxgen.ALTERNATE1();
                hostgen.ADDRESS_TEST();
                rxgen.deliver_pkt_to_driver();
                hostgen.deliver_pkt_to_driver();
        endtask : initiate_generator_ADDRESS_TEST
//-------------------------------------------------
        task initiate_generator_REQ_TEST();
                rxgen.ALTERNATE2();
                hostgen.REQ_TEST();
                rxgen.deliver_pkt_to_driver();
                hostgen.deliver_pkt_to_driver();
        endtask : initiate_generator_REQ_TEST
//-------------------------------------------------
        task initiate_generator_RANDOM_TEST();
                rxgen.ALTERNATE2();
                hostgen.RANDOM_TEST();
                rxgen.deliver_pkt_to_driver();
                hostgen.deliver_pkt_to_driver();
        endtask : initiate_generator_RANDOM_TEST
//-------------------------------------------------
// END CONFIGURABILITY STUFF
//-------------------------------------------------
//-------------------------------------------------
        task wait_for_sometime();
                #500ns;
        endtask : wait_for_sometime
//-------------------------------------------------
        task finish_test();
          $finish();
        endtask : finish_test
//-------------------------------------------------


endmodule
```

```
//------------------------------------------------------------------------
// UART INTERFACE  -- host side
// COPYRIGHT @ 2019 VIPINJASORIA
// ONLY FOR REFERENCE
//------------------------------------------------------------------------
interface uart_host_if(input bit CLK, reset);

        parameter i_skew = 2ns;
        parameter o_skew = 1ns;

        logic                   UREG_RDWR;
        logic   [7:0]   UREG_ADDR;
        logic   [7:0]   UREG_WRDAT;
        logic                   UREG_REQ;
        logic                   UREG_GNT;
        logic   [7:0]   UREG_RDDAT;
        logic                   UART_INTERRUPT;

//------------------------------------------------------------------------
        modport  host_module (input CLK, input reset, clocking dr_cb);
//------------------------------------------------------------------------

        clocking dr_cb @(posedge CLK);
                default input #i_skew   output #o_skew;
                output  UREG_RDWR, UREG_ADDR,
                        UREG_WRDAT,UREG_REQ;
                input UREG_GNT, UREG_RDDAT, UART_INTERRUPT;
        endclocking

        clocking mn_cb @(posedge CLK);
                default input #i_skew   output #o_skew;
                input   UREG_RDWR, UREG_ADDR,
                        UREG_WRDAT,UREG_REQ;
                output UREG_GNT, UREG_RDDAT, UART_INTERRUPT;
        endclocking

endinterface: uart_host_if
//------------------------------------------------------------------------
//------------------------------------------------------------------------
```

```
//-------------------------------------------------
//      CopyRight 2019 VIPIN JASORIA
//        Monitor Module -- Serial
// ONLY FOR REFERENCE
//      mon2sb_mbx1 -- Serial side MB
//      mon2sb_mbx2 -- Host Side MB
//-------------------------------------------------

class rx_monitor;
        virtual         uart_host_if    host_vif;
        mailbox         #(host_trans)   mon2sb_mbx2;
        host_trans      pkt;
        uart_config     cfg;
        int count;
        string mon_type;

//-------------------------------------------------
        function new (uart_config cfg)
                this.cfg        =       cfg;
                this.mon2sb_mbx2=       new();
                count           =       1;
        endfunction: new
//-------------------------------------------------
        function void connect();
                if(mon_type     == "INPUT MONITOR")
                        cfg.imon2sb_mbx2        =       mon2sb_mbx2;
                else if (mon_type == "OUTPUT MONITOR")
                        cfg.omon2sb_mbx2        =       mon2sb_mbx2;
        endfunction: connect
//-------------------------------------------------
        virtual task run_monitor();
                wait_for_reset_done();
                fork
                        collect_transfer();
                join_none
        endtask : run_monitor
//-------------------------------------------------
        task wait_for_reset_done();
                wait(cfg.RST_DONE.triggered);
        endtask: wait_for_reset_done
//-------------------------------------------------
        virtual task collect_transfer();
                forever begin
                        wait_for_a_valid_transaction_cycle();
```

```
                        create_and_populate_pkt();
                        put_pkt_into_mailbox();
                        @(host_vif.mr_cb);
                end
        endtask : collect_transfer
//-------------------------------------------------
        virtual function void create_and_populate_pkt();
                pkt     =       new();
                pkt.RXD =       rx_vif.mr_cb.RXD; // ALL HOST SIGNALS
        endfunction: create_and_populate_pkt
//-------------------------------------------------
        task put_pkt_into_mailbox();
                mon2sb_mbx2.put(pkt);
        endtask: put_pkt_into_mailbox
//-------------------------------------------------
endclass : rx_monitor
```

```
// Copyright Vipin Jasoria
// ONLY FOR REFERENCE
// UART TRANSCTOR Module  -- SERIAL
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE

class rx_trans;


        rand logic      [7:0]   RXD;

//------------------------------------------------------------------------
        function new();
        endfunction
//------------------------------------------------------------------------
        virtual function void print();
                $display("-------------------------");
                $display("RXD = %h", RXD);
                $display("-------------------------");
        endfunction: print
//------------------------------------------------------------------------
        virtual function rx_trans copy();
                copy = new();
                copy.RXD = this.RXD;
//              copy.expected_output = this.expected_output;
        endfunction: copy
//------------------------------------------------------------------------
endclass : rx_trans
```

```
// Copyright Vipin Jasoria
// ONLY FOR REFERENCE
// UART Driver Module  -- HOST
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE

//-------------------------------------------------------
class host_driver;
        host_trans pkt;
        virtual         uart_host_if    host_vif;
        mailbox         #(host_trans)   gen2dr_mbx2;
        uart_config     cfg;
        int count;


//-------------------------------------------------------
        function new (uart_config cfg);
                this.cfg = cfg;
                count = 1;
        endfunction

//-------------------------------------------------------
        function void connect();
                host_vif        =       cfg.host_vif;
                gen2dr_mbx2     =       cfg.gen2dr_mbx2;
        endfunction : connect

//-------------------------------------------------------
        virtual task run_driver();
                wait_for_reset_done();
                fork
                        forever begin
                                get_pkt_from_mailbox();
                                drive_transfer(pkt);
                        end
                join_none
        endtask : run_driver

//-------------------------------------------------------
        task wait_for_reset_done();
                wait(cfg.RST_DONE.triggered);
                @(host_vif.dr_cb);
        endtask : wait_for_reset_done
```

```
//-------------------------------------------------------
        task get_pkt_from_mailbox();
                gen2dr_mbx2.get(pkt);
                $display("Driver got Packet..");
                pkt.print();
        endtask: get_pkt_from_mailbox

//-------------------------------------------------------
        virtual task introduce_random_delay_between_transactions();
                int trans_delay;
                trans_delay = $urandom_range(cfg.driver_delay,0);
                repeat(trans_delay) @(host_vif.dr_cb);
        endtask: introduce_random_delay_between_transactions

//-------------------------------------------------------
        task set_dut_inputs();
                host_vif.dr_cb.UREG_RDWR        <=      pkt.UREG_RDWR;
                host_vif.dr_cb.UREG_ADDR        <=      pkt.UREG_ADDR;
                host_vif.dr_cb.UREG_WRDAT       <=      pkt.UREG_WRDAT;
                host_vif.dr_cb.UREG_REQ         <=      pkt.UREG_REQ;
                @(posedge host_vif.CLK)
                host_vif.dr_cb.UREG_REQ         <=      ~(pkt.UREG_REQ);

        endtask : set_dut_inputs

//-------------------------------------------------------
        virtual function void init_dut_inputs();
                host_vif.dr_cb.UREG_RDWR        <=      1'b0;
                host_vif.dr_cb.UREG_ADDR        <=      8'b0000_1000;
                host_vif.dr_cb.UREG_WRDAT       <=      0;
                host_vif.dr_cb.UREG_REQ         <=      1'b0;
        endfunction: init_dut_inputs

//-------------------------------------------------------
        virtual task drive_transfer(host_trans pkt);
                introduce_random_delay_between_transactions();
                init_dut_inputs();
                introduce_random_delay_between_transactions();
                introduce_random_delay_between_transactions();
                set_dut_inputs();
                introduce_random_delay_between_transactions();
//              wait_for_dut_to_take_input();
                count++;
        endtask : drive_transfer
```

```
//-------------------------------------------------------------

endclass : host_driver
```

```
//------------------------------------------------------------
//  UART GENERATOR MODULE
// Copyright 2019 VipinJasoria
// ONLY FOR REFERENCE
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE
//------------------------------------------------------------
class rx_gen;
        rx_trans pkt;
        mailbox #(rx_trans) gen2dr_mbx1;
        uart_config cfg;
//------------------------------------------------------------
        function new(uart_config cfg);
                this.cfg = cfg;
                this.gen2dr_mbx1 = new();
        endfunction
//------------------------------------------------------------
        function void connect();
                cfg.gen2dr_mbx1 = gen2dr_mbx1;
        endfunction: connect
//------------------------------------------------------------
//              ALL TEST CASES GOES HERE
//------------------------------------------------------------
        virtual function void RXD_TEST;
                pkt = new();
                pkt.RXD         =       12;
        endfunction: RXD_TEST
//------------------------------------------------------------
        virtual function void RXD_TEST_RNDM;
                pkt = new();
                pkt.RXD = $random;
        endfunction: RXD_TEST_RNDM
//------------------------------------------------------------
        virtual function void ALL_ZERO;
                pkt = new();
                pkt.RXD = 0;
        endfunction: ALL_ZERO
//------------------------------------------------------------
        virtual function void ALL_ONCES;
                pkt = new();
                pkt.RXD = 8'b1111_1111;
        endfunction: ALL_ONCES
//------------------------------------------------------------
        virtual function void ALTERNATE1;
```

```
                pkt = new();
                pkt.RXD = 8'b0101_0101;
        endfunction: ALTERNATE1
//-------------------------------------------------------------
        virtual function void ALTERNATE2;
                pkt = new();
                pkt.RXD = 8'b1010_1010;
        endfunction: ALTERNATE2


//-------------------------------------------------------------
//              END TEST CASES
//-------------------------------------------------------------
        virtual task deliver_pkt_to_driver();
                gen2dr_mbx1.put(pkt);
        endtask : deliver_pkt_to_driver
//-------------------------------------------------------------

endclass: rx_gen
```

```
//------------------------------------------------
//      CopyRight 2019 VIPIN JASORIA
//        Monitor Module -- Serial
// ONLY FOR REFERENCE
//      mon2sb_mbx1 -- Serial side MB
//      mon2sb_mbx2 -- Host Side MB
//------------------------------------------------

class rx_monitor;
        virtual         uart_rx_if      rx_vif;
        mailbox         #(rx_trans)     mon2sb_mbx1;
        rx_trans        pkt;
        uart_config     cfg;
        int count;
        string mon_type;

//------------------------------------------------
        function new (uart_config cfg)
                this.cfg        =       cfg;
                this.mon2sb_mbx1=       new();
                count           =       1;
        endfunction: new
//------------------------------------------------
        function void connect();
                if(mon_type     == "INPUT MONITOR")
                        cfg.imon2sb_mbx1        =       mon2sb_mbx1;
                else if (mon_type == "OUTPUT MONITOR")
                        cfg.omon2sb_mbx1        =       mon2sb_mbx1;
        endfunction: connect
//------------------------------------------------
        virtual task run_monitor();
                wait_for_reset_done();
                randomize_and_drive_stall();
                fork
                        collect_transfer();
                join_none
        endtask : run_monitor
//------------------------------------------------
        task wait_for_reset_done();
                wait(cfg.RST_DONE.triggered);
        endtask: wait_for_reset_done
//------------------------------------------------
        virtual task collect_transfer();
                forever begin
```

```
                    wait_for_a_valid_transaction_cycle();
                    create_and_populate_pkt();
                    put_pkt_into_mailbox();
                    @(rx_vif.mr_cb);
                end
        endtask : collect_transfer
//-------------------------------------------------
        virtual function void create_and_populate_pkt();
                pkt     =       new();
                pkt.RXD =       rx_vif.mr_cb.RXD; // USE for loop here
        endfunction: create_and_populate_pkt
//-------------------------------------------------
        task put_pkt_into_mailbox();
                mon2sb_mbx1.put(pkt);
        endtask: put_pkt_into_mailbox
//-------------------------------------------------
endclass : rx_monitor
```

```
//------------------------------------------------------------------------
// UART INTERFACE  -- serial side
// ONLY FOR REFERENCE
// COPYRIGHT @ 2019 VIPINJASORIA
//------------------------------------------------------------------------


interface uart_rx_if(input bit CLK, reset);

        parameter i_skew = 2ns;
        parameter o_skew = 1ns;

        logic           RXD;

//------------------------------------------------------------------------
        modport  rx_module (input CLK,input reset, clocking dr_cb);
//------------------------------------------------------------------------

        clocking dr_cb @(posedge CLK);
                default input #i_skew   output #o_skew;
                output   RXD;
        endclocking


        clocking mn_cb @(posedge CLK);
                default input #i_skew   output #o_skew;
                input    RXD;
        endclocking


endinterface: uart_rx_if
//------------------------------------------------------------------------
//------------------------------------------------------------------------
```

```
// Copyright Vipin Jasoria
// ONLY FOR REFERENCE
// UART Driver Module  -- Serial Side
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE
//-------------------------------------------------------
class rx_driver;
        rx_trans pkt;
        virtual         uart_rx_if      rx_vif;
        mailbox         #(rx_trans)     gen2dr_mbx1;
        uart_config     cfg;
        int count;


//-------------------------------------------------------
        function new (uart_config cfg);
                this.cfg = cfg;
                count = 1;
        endfunction

//-------------------------------------------------------
        function void connect();
                rx_vif          =       cfg.rx_vif;
                gen2dr_mbx1     =       cfg.gen2dr_mbx1;
        endfunction : connect

//-------------------------------------------------------
        virtual task run_driver();
                wait_for_reset_done();
                fork
                        forever begin
                                get_pkt_from_mailbox();
                                drive_transfer(pkt);
                        end
                join_none
        endtask : run_driver

//-------------------------------------------------------
        task wait_for_reset_done();
                wait(cfg.RST_DONE.triggered);
                @(rx_vif.dr_cb);
        endtask : wait_for_reset_done

//-------------------------------------------------------
```

```
        task get_pkt_from_mailbox();
                gen2dr_mbx1.get(pkt);
                $display("Driver got Packet..");
                pkt.print();
        endtask: get_pkt_from_mailbox

//-------------------------------------------------------
        virtual task introduce_random_delay_between_transactions();
                int trans_delay;
                trans_delay = $urandom_range(cfg.driver_delay,0);
                repeat(trans_delay) @(rx_vif.dr_cb);
        endtask: introduce_random_delay_between_transactions

//-------------------------------------------------------
//              SENDING DATA SERIALLY
//-------------------------------------------------------
        task set_dut_inputs();
                @(posedge rx_vif.CLK)   rx_vif.dr_cb.RXD        <=      1'b0;
                for (int i=0; i <= 7; i++) begin
                        @(posedge rx_vif.CLK)
                        rx_vif.dr_cb.RXD        <=      pkt.RXD[i];
                end
                @(posedge rx_vif.CLK)   rx_vif.dr_cb.RXD        <=      1'b1;
                @(posedge rx_vif.CLK)   rx_vif.dr_cb.RXD        <=      1'b1;

        endtask : set_dut_inputs
//-------------------------------------------------------
        virtual function void init_dut_inputs();
                rx_vif.dr_cb.RXD <= 1'b1;
        endfunction: init_dut_inputs

//-------------------------------------------------------
        virtual task drive_transfer(rx_trans pkt);
                introduce_random_delay_between_transactions();
                init_dut_inputs();
                introduce_random_delay_between_transactions();
                set_dut_inputs();
                introduce_random_delay_between_transactions();
//              wait_for_dut_to_take_input();
                count++;
        endtask : drive_transfer

endclass : rx_driver
```

```
// Copyright Vipin Jasoria
// ONLY FOR REFERENCE
// UART TRANSCTOR Module  -- HOST
// gen2dr_mbx1 MAILBOX BELONGS TO SERIAL SIDE
// gen2dr_mbx2 MAILBOX BELONGS TO HOST SIDE


class host_trans;

        rand logic                      UREG_RDWR;
        rand logic      [7:0]   UREG_ADDR;
        rand logic      [7:0]   UREG_WRDAT;
        rand logic                      UREG_REQ;
             logic                      UREG_GNT;       // Output
             logic      [7:0]   UREG_RDDAT;     // Output
             logic                      UART_INTERRUPT; // Output
             logic      [7:0]   expected_output_host;

//------------------------------------------------------------------------
        function new();
        endfunction
//------------------------------------------------------------------------
        virtual function void print();
                $display("--------------------------");
                $display("UREG_RDWR = %h\t, UREG_ADDR = %h", UREG_RDWR,UREG_ADDR);
                $display("UREG_WRDAT = %h\t, UREG_REQ = %h\t", UREG_WRDAT, UREG_REQ);
                $display("UREG_GNT = %h\t,UREG_RDDAT = %h\t,UART_INTERRUPT=%0h ", UREG_GNT, UREG_RDDAT, UART_INTERRUPT);
                $display("--------------------------");
        endfunction: print
//------------------------------------------------------------------------
        virtual function host_trans copy();
                copy = new();
//              copy.RSTN = this.RSTN;
                copy.UREG_RDWR = this.UREG_RDWR;
                copy.UREG_ADDR = this.UREG_ADDR;
                copy.UREG_WRDAT = this.UREG_WRDAT;
                copy.UREG_REQ = this.UREG_REQ;
                copy.UREG_GNT = this.UREG_GNT;
                copy.UREG_RDDAT = this.UREG_RDDAT;
                copy.UART_INTERRUPT = this.UART_INTERRUPT;
                copy.expected_output_host = this.expected_output_host;
        endfunction: copy
//------------------------------------------------------------------------
endclass : host_trans
```

```
# COPYRIGHT @ 2019 VIPIN JASORIA
# UART Synopsys Design Contraints


# SOME PARAMETERS
set PERIOD   200
set DELAY    0.01
set  CAP        0.5

# CLOCK
create_clock -name clk -period $PERIOD -waveform [list [expr {$PERIOD/2.0}] $PERIOD ] [get_ports CLK]

# INPUTS
set_input_delay -clock clk $DELAY  [get_ports UREG_RDWR]
set_input_delay -clock clk $DELAY  [get_ports UREG_ADDR]
set_input_delay -clock clk $DELAY  [get_ports UREG_WRDAT]
set_input_delay -clock clk $DELAY  [get_ports UREG_REQ]
set_input_delay -clock clk $DELAY  [get_ports RXD]



# OUTPUTS
set_output_delay -clock clk $DELAY  [get_ports UREG_GNT]
set_output_delay -clock clk $DELAY  [get_ports UREG_RDDAT]
set_output_delay -clock clk $DELAY  [get_ports UART_INTERRUPT]
#set_load $CAP [get_ports UREG_RDDAT]

# FALSE PATH
set_false_path -fall_from RSTN
```