

Abdalrahman Taha Mahmed

UART Using UVM

1.design

```
uart.sv
1 module uart( uart_if.DUT uartif);
2 parameter IDLE = 0, START = 1, DATA = 2, PARITY = 3, STOP = 4;
3 reg [3:0] state = IDLE;
4 reg [3:0] counter = 0;
5 reg [7:0] data_reg;
6 reg parity_bit;
7
8 always @(posedge uartif.clk or negedge uartif.rst_n) begin
9     if(!uartif.rst_n) begin
10         state <= IDLE;
11         uartif.TX_OUT <= 1'b1;
12         uartif.busy <= 1'b0;
13         counter <= 0;
14     end
15     else begin
16         case(state)
17             IDLE: begin
18                 uartif.TX_OUT <= 1'b1;
19                 uartif.busy <= 1'b0;
20                 counter <= 0;
21                 if (uartif.DATA_valid) begin
22                     data_reg <= uartif.P_DATA;
23                     parity_bit <= (uartif.PAR_TYP == 0) ? ~^uartif.P_DATA : ^uartif.P_DATA;
24                     state <= START;
25                     uartif.busy <= 1'b1;
26                 end
27             end
28             START: begin
29                 uartif.TX_OUT <= 1'b0;
30                 state <= DATA;
31                 counter <= 0;
32             end
33             DATA: begin
34                 uartif.TX_OUT <= data_reg[counter];
35                 counter <= counter + 1;
36                 if (counter == 7)
37                     state <= (uartif.PAR_EN) ? PARITY : STOP;
38             end
39             PARITY: begin
40                 uartif.TX_OUT <= parity_bit;
41                 state <= STOP;
42             end
43             STOP: begin
44                 uartif.TX_OUT <= 1'b1;
45                 state <= IDLE;
46             end
47         endcase
48     end
end
```

2. golden model

```
uart_golden_model.sv
1
2 module uart_golden_model ( uart_if.GOLDEN_MODEL uartif);
3
4 parameter IDLE = 0,
5           START = 1,
6           DATA = 2,
7           PARITY = 3,
8           STOP = 4;
9
10 reg [3:0] state = IDLE;
11 reg [3:0] counter = 0;
12 reg [7:0] data_reg;
13 reg parity_bit;
14
15 always @(posedge uartif.clk or negedge uartif.rst_n) begin
16     if(!uartif.rst_n) begin
17         state <= IDLE;
18         uartif.TX_OUT_ex <= 1'b1;
19         uartif.busy_ex <= 1'b0;
20         counter <= 4'b0000;
21     end
22     else begin
23         case(state)
24             IDLE: begin
25                 uartif.TX_OUT_ex <= 1'b1;
26                 uartif.busy_ex <= 1'b0;
27                 counter <= 4'b0000;
28                 if (uartif.DATA_valid) begin
29                     data_reg <= uartif.P_DATA;
30                     parity_bit <= (uartif.PAR_TYP) ? ^uartif.P_DATA : ~^uartif.P_DATA;
31                     state <= START;
32                     uartif.busy_ex <= 1'b1;
33                 end
34             end
35             START: begin
36                 uartif.TX_OUT_ex <= 1'b0;
37                 state <= DATA;
38                 counter <= 4'b0000;
39             end
40             DATA: begin
41                 uartif.TX_OUT_ex <= data_reg[counter];
42                 counter <= counter + 1;
43                 if (counter == 7)
44                     state <= (uartif.PAR_EN) ? PARITY : STOP;
45             end
46             PARITY: begin
47                 uartif.TX_OUT_ex <= parity_bit;
48                 state <= STOP;
49             end
50             STOP: begin
51                 uartif.TX_OUT_ex <= 1'b1;
52                 state <= IDLE;
53             end
54         endcase
55     end
56 end
57 endmodule
58
59
```

3.sva

```
uart_sva.sv
1
2 module uart_sva ( uart_if.DUT uartif );
3
4     parameter IDLE = 0, START = 1, DATA = 2, PARITY = 3, STOP = 4;
5
6     property reset_check;
7         @(posedge uartif.clk) (!uartif.rst_n) | =>
8             (uart.state==IDLE && uartif.TX_OUT==1'b1 && uartif.busy==1'b0 && uart.counter==8'b00000000);
9     endproperty
10    assert property (reset_check) else $error(" error in rst ");
11    cover property (reset_check);
12
13    property check1;
14        @(posedge uartif.clk) disable iff (!uartif.rst_n)
15            (uart.state==IDLE)
16            | => (uartif.TX_OUT==1 && uartif.busy==0 && uart.counter==0);
17    endproperty
18    assert property (check1) else $error("error in check1");
19    cover property (check1);
20
21    property check2;
22        @(posedge uartif.clk) disable iff (!uartif.rst_n)
23            (uart.state==IDLE && uartif.DATA_valid)
24            | => (uart.data_reg==$past(uartif.P_DATA));
25    endproperty
26    assert property (check2) else $error("error in check2");
27    cover property (check2);
28
29    property check3;
30        @(posedge uartif.clk) disable iff (!uartif.rst_n)
31            (uart.state==IDLE && uartif.DATA_valid)
32            | => (uart.state==START && uartif.busy==1'b1);
33    endproperty
34    assert property (check3) else $error("error in check3");
35    cover property (check3);
36
37    property check4;
38        @(posedge uartif.clk) disable iff (!uartif.rst_n)
39            (uart.state==IDLE && uartif.DATA_valid && uartif.PAR_TYP)
40            | => (uart.parity_bit == ^$past(uartif.P_DATA));
41    endproperty
42    assert property (check4) else $error("error in check4");
43    cover property (check4);
44
```

```

45     property check5;
46         @(posedge uartif.clk) disable iff (!uartif.rst_n)
47         (uart.state==IDLE && uartif.DATA_valid && !uartif.PAR_TYP)
48         |>= (uart.parity_bit == ~^$past(uartif.P_DATA));
49     endproperty
50     assert property (check5) else $error("error in check5");
51     cover property (check5);
52
53     property check6;
54         @(posedge uartif.clk) disable iff (!uartif.rst_n)
55         (uart.state==START)
56         |>= (uartif.TX_OUT==1'b0 && uart.state==DATA && uart.counter==8'b00000000);
57     endproperty
58     assert property (check6) else $error("error in check6");
59     cover property (check6);
60
61     property check7;
62         @(posedge uartif.clk) disable iff (!uartif.rst_n)
63         (uart.state==DATA)
64         |>= (uartif.TX_OUT==$past(uart.data_reg[uart.counter]));
65     endproperty
66     assert property (check7) else $error("error in check7");
67     cover property (check7);
68
69     property check8;
70         @(posedge uartif.clk) disable iff (!uartif.rst_n)
71         (uart.state==DATA)
72         |>= (uart.counter==$past(uart.counter) + 1'b1);
73     endproperty
74     assert property (check8) else $error("error in check8");
75     cover property (check8);
76
77     property check9;
78         @(posedge uartif.clk) disable iff (!uartif.rst_n)
79         (uart.state==DATA && uart.counter == 7)
80         |>= (uart.state== (uartif.PAR_EN) ? PARITY : STOP);
81     endproperty
82     assert property (check9) else $error("error in check9");
83     cover property (check9);
84

```

```

84
85     property check10;
86         @(posedge uartif.clk) disable iff (!uartif.rst_n)
87         (uart.state==PARITY)
88         |>= (uartif.TX_OUT== $past(uart.parity_bit));
89     endproperty
90     assert property (check10) else $error("error in check10");
91     cover property (check10);
92
93     property check11;
94         @(posedge uartif.clk) disable iff (!uartif.rst_n)
95         (uart.state==PARITY)
96         |>= (uart.state==STOP);
97     endproperty
98     assert property (check11) else $error("error in check11");
99     cover property (check11);
100
101     property check12;
102         @(posedge uartif.clk) disable iff (!uartif.rst_n)
103         (uart.state==STOP)
104         |>= (uartif.TX_OUT==1'b1 && uart.state==IDLE);
105     endproperty
106     assert property (check12) else $error("error in check12");
107     cover property (check12);
108
109     property check13;
110         @(posedge uartif.clk) disable iff (!uartif.rst_n)
111         (uart.state==IDLE || uart.state==STOP)
112         |>= (uartif.TX_OUT);
113     endproperty
114     assert property (check13) else $error("error in check13");
115     cover property (check13);
116
117 endmodule
118
119

```

4.if

```
uart_if.sv
1  interface uart_if (clk);
2      input bit clk;
3      logic rst_n, PAR_EN, PAR_TYP, DATA_valid, TX_OUT, TX_OUT_ex, busy, busy_ex;
4      logic [7:0] P_DATA;
5
6      modport DUT (
7          input clk, rst_n, PAR_EN, PAR_TYP, DATA_valid, P_DATA,
8          output TX_OUT, busy
9      );
10
11     modport GOLDEN_MODEL (
12         input clk, rst_n, PAR_EN, PAR_TYP, DATA_valid, P_DATA,
13         output TX_OUT_ex, busy_ex
14     );
15
16
17  endinterface
```

5.seq_items

```
uart_seq_item.sv
1  package uart_seq_item_pkg;
2  import uvm_pkg::*;
3  `include "uvm_macros.svh"
4  parameter IDLE = 0, START = 1, DATA = 2, PARITY = 3, STOP = 4;
5  class uart_seq_item extends uvm_sequence_item;
6      `uvm_object_utils(uart_seq_item)
7      rand logic rst_n;
8      rand logic [7:0] P_DATA;
9      rand logic PAR_EN;
10     rand logic PAR_TYP;
11     rand logic DATA_valid;
12     rand logic direction, cin;
13     logic TX_OUT;
14     logic TX_OUT_ex;
15     logic busy;
16     logic busy_ex;
17     bit clk;
18     bit [7:0] vals [] = '{8'b11111111, 8'b00000000, 8'b10101010};
19     bit [7:0] P_DATA_vals;
20
21     function new (string name = "uart_seq_item");
22         super.new (name);
23     endfunction
24     function string convert2string();
25         return $sformatf("%s rst_n=%b P_DATA=%b TX_OUT=%b busy=%b", super.convert2string(), rst_n, P_DATA, TX_OUT, busy);
26     endfunction
27     function string convert2string_stimulus();
28         return $sformatf(" rst_n=%b P_DATA=%b TX_OUT=%b busy=%b", rst_n, P_DATA, TX_OUT, busy);
29     endfunction
30
31     constraint rst_cns{
32         rst_n dist{0:= 3, 1:= 97};
33     }
34     constraint PAR_TYP_cns{
35         PAR_TYP dist{0:= 50, 1:= 50};
36     }
37     constraint PAR_EN_cns{
38         PAR_EN dist{0:= 75, 1:= 25};
39     }
40     constraint DATA_valid_cns{
41         DATA_valid dist{0:= 5, 1:= 95};
42     }
43     constraint P_DATA_cns {
44         P_DATA_vals inside {vals};
45         P_DATA dist {
46             8'b00001111 := 50,
47             P_DATA_vals := 4,
48             [1:255] := 50;
49         }
50     }
```

6.scoreboard

```
uart_scoreboard.sv
1 package uart_sco_pkg;
2 import uart_seq_item_pkg::*;
3 import uvm_pkg::*;
4 `include "uvm_macros.svh"
5
6 class uart_sco extends uvm_scoreboard;
7     `uvm_component_utils(uart_sco)
8
9     uvm_analysis_export #(uart_seq_item) sb_export;
10    uvm_tlm_analysis_fifo #(uart_seq_item) sb_fifo;
11    uart_seq_item seq_item_sb;
12    int error_count = 0;
13    int correct_count = 0;
14    function new (string name = "uart_sco" , uvm_component parent = null);
15        super.new (name,parent);
16    endfunction
17
18    function void build_phase (uvm_phase phase);
19        super.build_phase (phase);
20        sb_export = new("sb_export",this);
21        sb_fifo = new("sb_fifo",this);
22    endfunction
23
24    function void connect_phase(uvm_phase phase);
25        super.connect_phase(phase);
26        sb_export.connect(sb_fifo.analysis_export);
27    endfunction
28
29    task run_phase (uvm_phase phase);
30        super.run_phase(phase);
31        forever begin
32            sb_fifo.get(seq_item_sb);
33            if ((seq_item_sb.TX_OUT != seq_item_sb.TX_OUT_ex ) || (seq_item_sb.busy != seq_item_sb.busy_ex)) begin
34                `uvm_error("run_phase",$sformatf("compartion failed while ref = %b",seq_item_sb.TX_OUT_ex))
35                error_count++;
36            end
37            else
38                correct_count++;
39            end
40        endtask
41
42    function void report_phase (uvm_phase phase);
43        super.report_phase(phase);
44        `uvm_info("report_phase",$sformatf("corect = %d",correct_count) , UVM_MEDIUM)
45        `uvm_info("report_phase",$sformatf("error = %d",error_count) , UVM_MEDIUM)
46    endfunction
47 endclass
48 endpackage
```

7.config

```
uart_config.sv
1  package uart_conf_pkg;
2  import uvm_pkg::*;
3  `include "uvm_macros.svh"
4
5  class uart_config extends uvm_object;
6      `uvm_object_utils(uart_config)
7
8      virtual uart_if uart_vif;
9      uvm_active_passive_enum is_active;
10
11      function new (string name = "uart_config");
12          super.new(name);
13      endfunction
14
15  endclass
16
17 endpackage
```

8.seq

```
uart_seq.sv
1  package uart_seq_pkg;
2  import uvm_pkg::*;
3  import uart_seq_item_pkg::*;
4  `include "uvm_macros.svh"
5
6  class uart_reset_seq extends uvm_sequence #(uart_seq_item);
7      `uvm_object_utils(uart_reset_seq)
8      uart_seq_item seq_item;
9
10     function new(string name = "uart_reset_seq");
11         super.new(name);
12     endfunction
13
14     task body;
15         seq_item = uart_seq_item::type_id::create("seq_item");
16         start_item(seq_item);
17         seq_item.rst_n = 0;
18         finish_item (seq_item);
19     endtask
20 endclass
21
22 class uart_main_seq extends uvm_sequence #(uart_seq_item);
23     `uvm_object_utils(uart_main_seq)
24     uart_seq_item seq_item;
25
26     function new(string name = "uart_main_seq");
27         super.new(name);
28     endfunction
29
30     task body;
31         seq_item = uart_seq_item::type_id::create("seq_item");
32         repeat (9999) begin
33             start_item(seq_item);
34             assert (seq_item.randomize());
35             finish_item (seq_item);
36         end
37     endtask
38 endclass
39
40 endpackage
```


9.sequencer

```
uart_sequencer.sv
1  package uart_sequencer_pkg;
2  import uvm_pkg::*;
3  import uart_seq_item_pkg::*;
4  `include "uvm_macros.svh"
5
6  class uart_sequencer extends uvm_sequencer #(uart_seq_item);
7  `uvm_component_utils(uart_sequencer);
8
9  function new (string name = "uart_sequencer" , uvm_component parent = null);
10     super.new (name,parent);
11 endfunction
12 endclass
13
14 endpackage
```

10.driver

```
uart_driver.sv
1  package uart_driver_pkg;
2  import uvm_pkg::*;
3  import uart_seq_item_pkg::*;
4  `include "uvm_macros.svh"
5
6  class uart_driver extends uvm_driver #(uart_seq_item);
7  `uvm_component_utils (uart_driver)
8
9  virtual uart_if uart_vif;
10  uart_seq_item stim_seq_item;
11
12  function new (string name = "uart_driver" , uvm_component parent = null);
13     super.new (name,parent);
14 endfunction
15
16  task run_phase (uvm_phase phase);
17     super.run_phase(phase);
18     forever begin
19         stim_seq_item = uart_seq_item::type_id::create("stim_seq_item");
20         seq_item_port.get_next_item(stim_seq_item);
21
22         @(negedge uart_vif.clk);
23         uart_vif.P_DATA      = stim_seq_item.P_DATA;
24         uart_vif.rst_n       = stim_seq_item.rst_n;
25         uart_vif.PAR_EN      = stim_seq_item.PAR_EN;
26         uart_vif.PAR_TYP     = stim_seq_item.PAR_TYP;
27         uart_vif.DATA_valid  = 1'b1;
28         @(posedge uart_vif.clk);
29         uart_vif.DATA_valid  = 1'b0;
30         repeat (8) @(posedge uart_vif.clk);
31         seq_item_port.item_done();
32
33         `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
34     end
35 endtask
36
37 endclass
38 endpackage
```

11.monitor

```
uart_monitor.sv
1 package uart_monitor_pkg;
2 import uvm_pkg::*;
3 import uart_seq_item_pkg::*;
4 `include "uvm_macros.svh"
5
6 class uart_monitor extends uvm_monitor;
7   `uvm_component_utils (uart_monitor)
8
9   virtual uart_if uart_vif;
10   uart_seq_item rsp_seq_item;
11   uvm_analysis_port #(uart_seq_item) mon_ap;
12
13   function new (string name = "uart_monitor" , uvm_component parent = null);
14     super.new (name,parent);
15   endfunction
16
17   function void build_phase (uvm_phase phase);
18     super.build_phase(phase);
19     mon_ap = new("mon_ap",this);
20   endfunction
21
22   task run_phase (uvm_phase phase);
23     super.run_phase(phase);
24     forever begin
25       rsp_seq_item = uart_seq_item::type_id::create ("rsp_seq_item");
26       @(negedge uart_vif.clk);
27       rsp_seq_item.P_DATA = uart_vif.P_DATA;
28       rsp_seq_item.rst_n = uart_vif.rst_n;
29       rsp_seq_item.DATA_valid = uart_vif.DATA_valid;
30       rsp_seq_item.PAR_EN = uart_vif.PAR_EN;
31       rsp_seq_item.PAR_TYP = uart_vif.PAR_TYP;
32       rsp_seq_item.TX_OUT = uart_vif.TX_OUT;
33       rsp_seq_item.busy = uart_vif.busy;
34       mon_ap.write(rsp_seq_item);
35       `uvm_info("run_phase" , rsp_seq_item.convert2string_stimulus(),UVM_HIGH)
36     end
37   endtask
38 endclass
39 endpackage
```

12.coverage

```
uart_coverage.sv
1  package uart_coverage_pkg;
2  import uart_seq_item_pkg::*;
3  import uvm_pkg::*;
4  `include "uvm_macros.svh"
5
6  class uart_coverage extends uvm_component;
7      `uvm_component_utils(uart_coverage)
8
9      uvm_analysis_export #(uart_seq_item) cov_export;
10     uvm_tlm_analysis_fifo #(uart_seq_item) cov_fifo;
11     uart_seq_item seq_item_cov;
12
13     covergroup cvr_gp ;
14         label_P_DATA : coverpoint seq_item_cov.P_DATA {
15             bins data1 = {8'b00000000};
16             bins data2[] = {8'b11111111 , 8'b00000000 , 8'b10101010};}
17
18         label_PAR_TYP : coverpoint seq_item_cov.PAR_TYP;
19
20         label_PAR_EN : coverpoint seq_item_cov.PAR_EN;
21
22         label_DATA_VALID : coverpoint seq_item_cov.DATA_valid;
23
24         label_tx_out : coverpoint seq_item_cov.TX_OUT;
25
26         label_busy : coverpoint seq_item_cov.busy;
27
28         cross_EN_TYP : cross label_PAR_EN , label_PAR_TYP ;
29
30         cross_DATA_valid : cross label_P_DATA , label_DATA_VALID
31             { ignore_bins bins_0_0 = binsof (label_P_DATA) intersect {0} &&
32               binsof (label_DATA_VALID) intersect {0};}
33
34         cross_tx_out_busy : cross label_busy , label_tx_out
35             { ignore_bins bins_0_0 = binsof (label_busy) intersect {0} &&
36               binsof (label_tx_out) intersect {0};}
37
38     endgroup
39
40     function new (string name = "uart_cov" , uvm_component parent = null);
41         super.new (name,parent);
42         cvr_gp = new();
43     endfunction
44
45     function void build_phase (uvm_phase phase);
46         super.build_phase (phase);
47         cov_export = new("cov_export",this);
48         cov_fifo = new("cov_fifo",this);
49     endfunction
50
51     function void connect_phase(uvm_phase phase);
52         super.connect_phase(phase);
53         cov_export.connect(cov_fifo.analysis_export);
54     endfunction
55
56     task run_phase (uvm_phase phase);
57         super.run_phase(phase);
58         forever begin
59             cov_fifo.get(seq_item_cov);
60             cvr_gp.sample();
61         end
62     endtask
63 endclass
64 endpackage
```

13.agent

```
uart_agents.v
1  package uart_agent_pkg;
2  import uart_driver_pkg::*;
3  import uart_monitor_pkg::*;
4  import uart_sequencer_pkg::*;
5  import uart_seq_item_pkg::*;
6  import uart_conf_pkg::*;
7  import uvm_pkg::*;
8  `include "uvm_macros.svh"
9
10 class uart_agent extends uvm_agent;
11     `uvm_component_utils(uart_agent)
12
13     uart_sequencer sqr;
14     uart_driver drv;
15     uart_monitor mon;
16     uart_config uart_cfg;
17     uvm_analysis_port #(uart_seq_item) agt_ap;
18
19     function new (string name = "uart_agent" , uvm_component parent = null);
20         super.new (name,parent);
21     endfunction
22
23     function void build_phase (uvm_phase phase);
24
25         super.build_phase (phase);
26         if (!uvm_config_db #(uart_config)::get(this,"","CFG",uart_cfg))
27             `uvm_fatal ("build_phase","test - unable to get the configuration");
28
29         if (uart_cfg.is_active == UVM_ACTIVE) begin
30             sqr = uart_sequencer::type_id::create ("sqr",this);
31             drv = uart_driver::type_id::create ("drv",this);
32         end
33         mon = uart_monitor::type_id::create ("mon",this);
34         agt_ap = new("agt_ap",this);
35     endfunction
36
37     function void connect_phase(uvm_phase phase);
38         mon.uart_vif = uart_cfg.uart_vif;
39         mon.mon_ap.connect(agt_ap);
40         if (uart_cfg.is_active == UVM_ACTIVE) begin
41             drv.uart_vif = uart_cfg.uart_vif;
42             drv.seq_item_port.connect(sqr.seq_item_export);
43         end
44     endfunction
45 endclass
46
47 endpackage
```

14.env

```
uart_env.sv
1
2 package uart_env_pkg;
3 import uart_agent_pkg::*;
4 import uart_sco_pkg::*;
5 import uart_coverage_pkg::*;
6 import uvm_pkg::*;
7 `include "uvm_macros.svh"
8
9 class uart_env extends uvm_env;
10     `uvm_component_utils (uart_env)
11
12     uart_agent agt;
13     uart_sco sb;
14     uart_coverage cov;
15
16     function new (string name = "uart_env" , uvm_component parent = null);
17         super.new (name,parent);
18     endfunction
19
20     function void build_phase (uvm_phase phase);
21         super.build_phase (phase);
22         agt = uart_agent::type_id::create ("agt",this);
23         sb = uart_sco::type_id::create ("sb",this);
24         cov = uart_coverage::type_id::create ("cov",this);
25     endfunction
26
27     function void connect_phase (uvm_phase phase);
28         agt.agt_ap.connect(sb.sb_export);
29         agt.agt_ap.connect(cov.cov_export);
30     endfunction
31
32 endclass
33 endpackage
```

15.test

```
uart_test.sv
1
2 package uart_test_pkg;
3 import uart_env_pkg::*;
4 import uart_conf_pkg::*;
5 import uart_seq_pkg::*;
6 import uvm_pkg::*;
7 import uart_seq_item_pkg::*;
8 `include "uvm_macros.svh"
9
10 class uart_test extends uvm_test;
11
12     `uvm_component_utils (uart_test)
13
14     uart_env env;
15     uart_config uart_cfg;
16     virtual uart_if uart_vif;
17     uart_main_seq main_seq;
18     uart_reset_seq reset_seq;
19
20     function new (string name = "uart_env" , uvm_component parent = null);
21         super.new (name,parent);
22     endfunction
23
24     function void build_phase (uvm_phase phase);
25         super.build_phase(phase);
26         env = uart_env::type_id::create ("env",this);
27         uart_cfg = uart_config::type_id::create ("uart_cfg");
28         main_seq = uart_main_seq::type_id::create("main_seq");
29         reset_seq = uart_reset_seq::type_id::create("reset_seq");
30
31         uart_cfg.is_active = UVM_ACTIVE;
32
33         if (!uvm_config_db#(virtual uart_if)::get(this,"","uart_IF",uart_vif))
34             `uvm_fatal ("build_phase","test - unable to get the virtual interface");
35
36         uvm_config_db#(uart_config)::set(this,"*", "CFG",uart_cfg);
37
38     endfunction
39
40     task run_phase (uvm_phase phase);
41         super.run_phase(phase);
42         phase.raise_objection(this);
43
44         //reset
45         `uvm_info ("run_phase" , "reset asserted" , UVM_LOW)
46         reset_seq.start(env.agt.sqr);
47         `uvm_info ("run_phase" , "reset deasserted" , UVM_LOW)
48
49         //main
50         `uvm_info ("run_phase" , "stimulus generation started 1" , UVM_LOW)
51         main_seq.start(env.agt.sqr);
52         `uvm_info ("run_phase" , "stimulus generation ended 1" , UVM_LOW)
53
54         //reset
55         `uvm_info ("run_phase" , "reset asserted" , UVM_LOW)
56         reset_seq.start(env.agt.sqr);
57         `uvm_info ("run_phase" , "reset deasserted" , UVM_LOW)
58
59         phase.drop_objection(this);
60     endtask
61
62 endclass: uart_test
63 endpackage
```

16.top

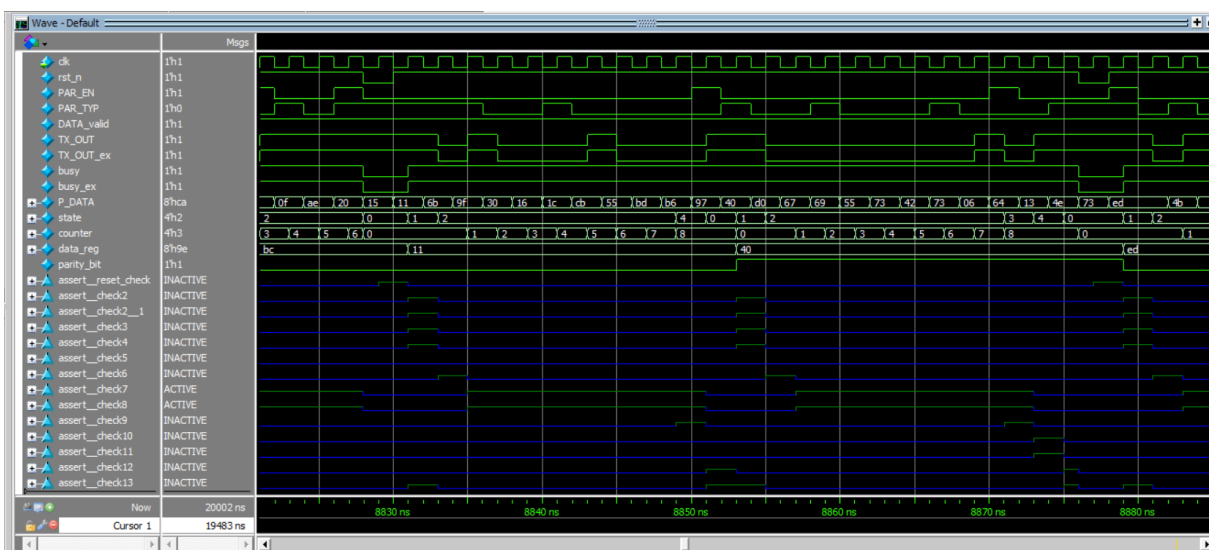
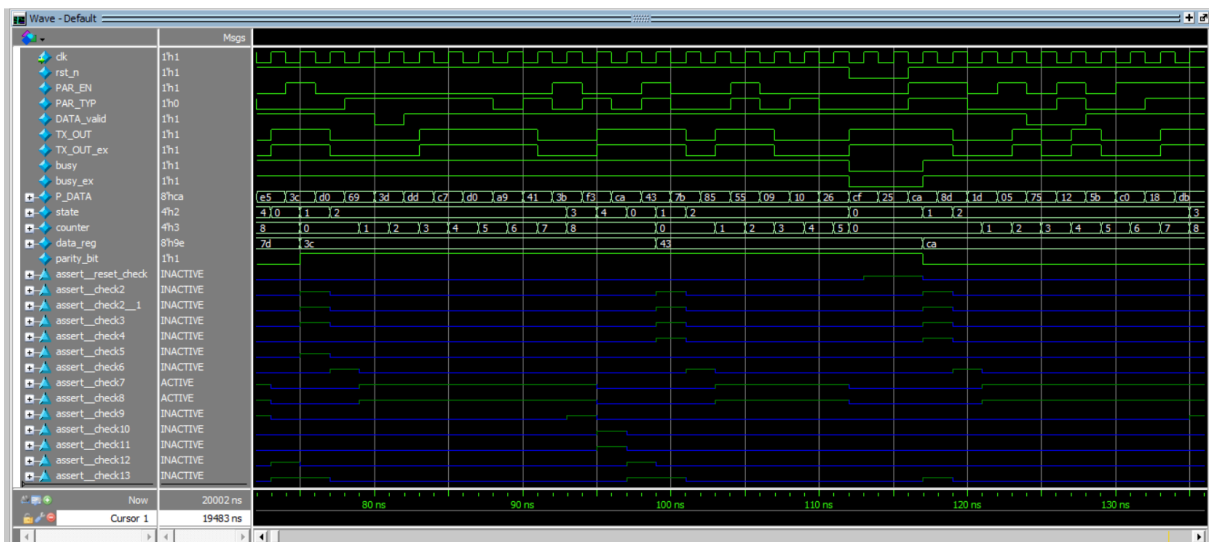
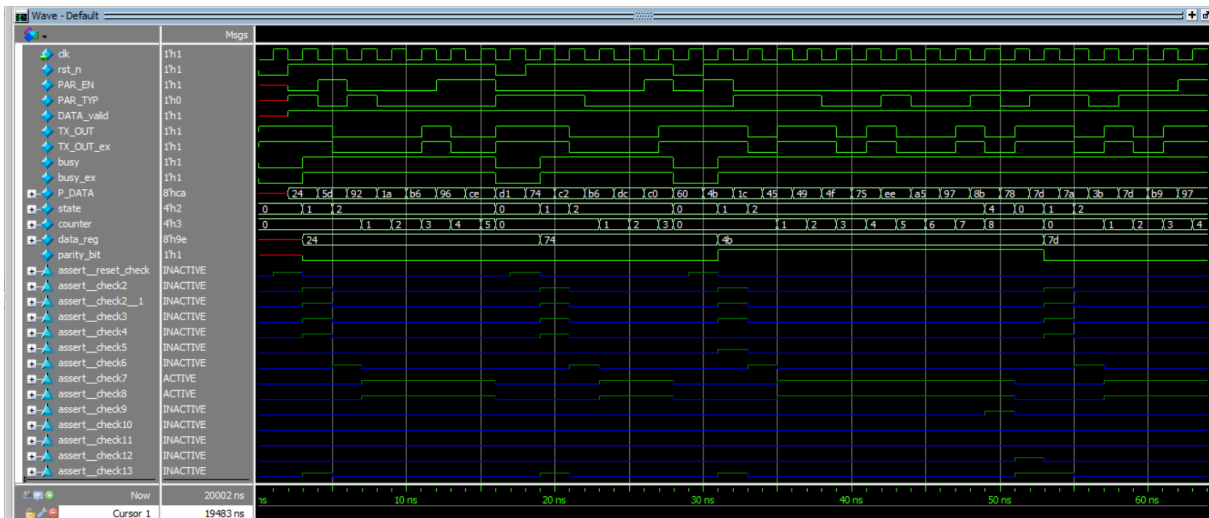
```
1
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4 import uart_test_pkg::*;
5
6 module uart_top();
7
8     bit clk;
9
10    initial begin
11        forever begin
12            #1 clk = ~clk;
13        end
14    end
15
16    uart_if uartif (clk);
17    uart DUT (uartif);
18    uart_golden_model GOLDEN_MODEL (uartif);
19    bind uart uart_sva sv (uartif);
20
21    initial begin
22        uvm_config_db#(virtual uart_if)::set(null,"uvm_test_top","uart_IF",uartif);
23        run_test("uart_test");
24    end
25
26 endmodule
```

17.do file

```
uart.do
1 vlib work
2 vlog -f uart_files.list
3 vsim -voptargs=+acc work.uart_top -classdebug -uvmcontrol=all
4 add wave /uart_top/uartif/*
5 run -all
```

```
uart_files.list
1 uart.sv
2 uart_golden_model.sv
3 uart_sva.sv
4 uart_if.sv
5 uart_seq_item.sv
6 uart_scoreboard.sv
7 uart_config.sv
8 uart_seq.sv
9 uart_sequencer.sv
10 uart_driver.sv
11 uart_monitor.sv
12 uart_coverage.sv
13 uart_agent.sv
14 uart_env.sv
15 uart_test.sv
16 uart_top.sv
17
```

18.waveform

























































19.Assertion

uart_seq_pkg::uart_main_seq::body/#ublk#2432689...	Immediate	SVA	on	0	1	-	-	-	-	off	assert (/randomize(...))	✓
uart_top/DUT/sv/assert_reset_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) (~uar...	✓
uart_top/DUT/sv/assert_check2	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check2_1	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check3	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check4	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check5	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check6	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check7	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check8	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check9	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check10	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check11	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check12	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓
uart_top/DUT/sv/assert_check13	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge uart_tf.clk) disab...	✓

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
uart_top/DUT/sv/cover_check13	SVA	✓	Off	1712	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check12	SVA	✓	Off	692	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check11	SVA	✓	Off	177	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check10	SVA	✓	Off	177	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check9	SVA	✓	Off	716	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check8	SVA	✓	Off	6515	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check7	SVA	✓	Off	6515	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check6	SVA	✓	Off	946	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check5	SVA	✓	Off	493	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check4	SVA	✓	Off	461	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check3	SVA	✓	Off	974	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check2_1	SVA	✓	Off	974	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_check2	SVA	✓	Off	974	1	Unlimited	1	100%		✓	0	0	0 ns	0
uart_top/DUT/sv/cover_reset_check	SVA	✓	Off	329	1	Unlimited	1	100%		✓	0	0	0 ns	0

20.covergroup

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_inst
uart_coverage_pk...		100.00%					
TYPE cvr_gp		100.00%	100	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin data...		1	1	100.00...		✓	
bin data...		1	1	100.00...		✓	
bin data...		33	1	100.00...		✓	
bin data...		40	1	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin auto...		4947	1	100.00...		✓	
bin auto...		5052	1	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin auto...		7461	1	100.00...		✓	
bin auto...		2538	1	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin auto...		507	1	100.00...		✓	
bin auto...		9492	1	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin auto...		4394	1	100.00...		✓	
bin auto...		5607	1	100.00...		✓	
CVP cvr_g...		100.00%	100	100.00...		✓	
bin auto...		379	1	100.00...		✓	
bin auto...		9622	1	100.00...		✓	
CROSS cvr...		100.00%	100	100.00...		✓	
bin <au...		1267	1	100.00...		✓	
bin <au...		3785	1	100.00...		✓	
bin <au...		1271	1	100.00...		✓	
bin <au...		3676	1	100.00...		✓	

	CROSS cvr...	100.00%	100	100.00...		
	bin <au...	1267	1	100.00...		
	bin <au...	3785	1	100.00...		
	bin <au...	1271	1	100.00...		
	bin <au...	3676	1	100.00...		
	CROSS cvr...	100.00%	100	100.00...		
	bin <da...	39	1	100.00...		
	bin <da...	30	1	100.00...		
	bin <da...	1	1	100.00...		
	bin <da...	1	1	100.00...		
	bin <da...	1	1	100.00...		
	bin <da...	3	1	100.00...		
	ignore...	0	-	-		
	CROSS cvr...	100.00%	100	100.00...		
	bin <au...	5228	1	100.00...		
	bin <au...	4394	1	100.00...		
	bin <au...	379	1	100.00...		
	ignore...	0	-	-		

21.output

```

Transcript
*****
# UVM_INFO uart_test.sv(47) @ 2: uvm_test_top [run_phase] reset deasserted
# UVM_INFO uart_test.sv(50) @ 2: uvm_test_top [run_phase] stimulus generation started 1
# UVM_INFO uart_test.sv(52) @ 20000: uvm_test_top [run_phase] stimulus generation ended 1
# UVM_INFO uart_test.sv(55) @ 20000: uvm_test_top [run_phase] reset asserted
# UVM_INFO uart_test.sv(57) @ 20002: uvm_test_top [run_phase] reset deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO uart_scoreboard.sv(44) @ 20002: uvm_test_top.env.sb [report_phase] correct = 10001
# UVM_INFO uart_scoreboard.sv(45) @ 20002: uvm_test_top.env.sb [report_phase] error = 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 12
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [report_phase] 2
# [run_phase] 6
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 20002 ns Iteration: 61 Instance: /uart_top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
V$SIM 17>

```