DAY-100
#100DAYSRTL

“UVM: Project-2”

“Aim”:-To Verify the 8 bit RAM using UVM test benches

“Design Code”:-

```
module ram (input clk, wr,input [7:0] din,output reg [7:0] dout,input [3:0] addr);
  reg [7:0] mem [15:0];
  always@(posedge clk)
    begin
      if(wr)
        begin
          mem[addr] <= din;
        end
      else
        begin
          dout <= mem[addr];
        end
    end
endmodule
```

“Interface”:-

```
interface ram_if;
  logic clk;
  logic wr;
  logic [7:0] din;
  logic [7:0] dout;
  logic [3:0] addr;
endinterface
```

“Transaction”:-

```
class transaction extends uvm_sequence_item;
  `uvm_object_utils(transaction)
rand bit wr;
rand bit [7:0] din;
rand bit [3:0] addr;
bit [7:0] dout;
constraint addr_c {addr == 3;};
  function new(input string inst = "transaction");
  super.new(inst);
  endfunction
endclass
```

“Generator”:-

```
class generator extends uvm_sequence#(transaction);
`uvm_object_utils(generator)
transaction t;
  function new(string inst = "generator");
  super.new(inst);
  endfunction
    virtual task body();
      for(int i =0; i< 10; i++) begin
        t = transaction::type_id::create("TRANS");
        start_item(t);
        t.randomize();
        $display("--------------------------------------------");
       `uvm_info("GEN", $sformatf("SEQ -> Driver wr:%0d addr :%0d din:%0d dout:%0d",t.wr, t.addr,t.din, t.dout), UVM_NONE);
        finish_item(t);
      end
    endtask
endclass
```

“Driver”:-

```
class driver extends uvm_driver#(transaction);
`uvm_component_utils(driver)
transaction t;
virtual ram_if rif;
  function new(input string inst = "driver", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    t = transaction::type_id::create("t");
    if(!uvm_config_db#(virtual ram_if)::get(this,"","rif",rif))
     `uvm_error("DRV", "Unable to access Interface");
  endfunction
virtual task run_phase(uvm_phase phase);
    forever begin
    seq_item_port.get_next_item(t);
    rif.wr   <= t.wr;
    rif.din  <= t.din;
    rif.addr <= t.addr;
      `uvm_info("DRV", $sformatf("DRV -> DUT wr:%0d addr :%0d din:%0d dout:%0d",t.wr, t.addr,t.din, t.dout), UVM_NONE);
    seq_item_port.item_done();
    @(posedge rif.clk);
    if(t.wr == 1'b0)
      @(posedge rif.clk);
    end
endtask
endclass
```

“Monitor”:-

```
class monitor extends uvm_monitor;
`uvm_component_utils(monitor)
uvm_analysis_port #(transaction) send;
virtual ram_if rif;
transaction t;
  function new(input string inst = "monitor", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  t = transaction::type_id::create("TRANS");
    send = new("send",this);
  if(!uvm_config_db#(virtual ram_if)::get(this,"","rif",rif))
  `uvm_error("MON", "Unable to access Interface");
endfunction
virtual task run_phase(uvm_phase phase);
forever begin
@(posedge rif.clk);
t.wr = rif.wr;
t.din = rif.din;
t.addr = rif.addr;
if(rif.wr == 1'b0) begin
    @(posedge rif.clk)
    t.dout = rif.dout;
end
  `uvm_info("MON", $sformatf("DUT -> MON wr:%0d addr :%0d din:%0d dout:%0d",t.wr, t.addr,t.din, t.dout), UVM_NONE);
send.write(t);
end
endtask
endclass
```

“Scoreboard”:-

```
class scoreboard extends uvm_scoreboard;
`uvm_component_utils(scoreboard)
uvm_analysis_imp #(transaction,scoreboard) recv;
reg [7:0] tarr[20] = '{default:0} ;
function new(input string inst = "SCO", uvm_component c);
super.new(inst,c);
recv = new("recv",this);
endfunction
virtual function void write(transaction data);
  if(data.wr == 1'b1)
    begin
      tarr[data.addr] = data.din;
      `uvm_info("SCO", $sformatf("MON -> SCO : DATA WRITE OP :  addr :%0d din:%0d ", data.addr,data.din), UVM_NONE);
    end
  if(data.wr == 1'b0)
    begin
      if(data.dout == tarr[data.addr])
        `uvm_info("SCO", $sformatf("TEST PASSED : DATA READ OP   addr :%0d din:%0d dout:%0d ", data.addr,data.din, data.dout), UVM_NONE)
      else
        `uvm_error("SCO", $sformatf("TEST FAILED : DATA READ OP :  addr :%0d din:%0d dout:%0d ", data.addr,data.din, data.dout))
    end
  $display("--------------------------------------------");
endfunction
endclass
```

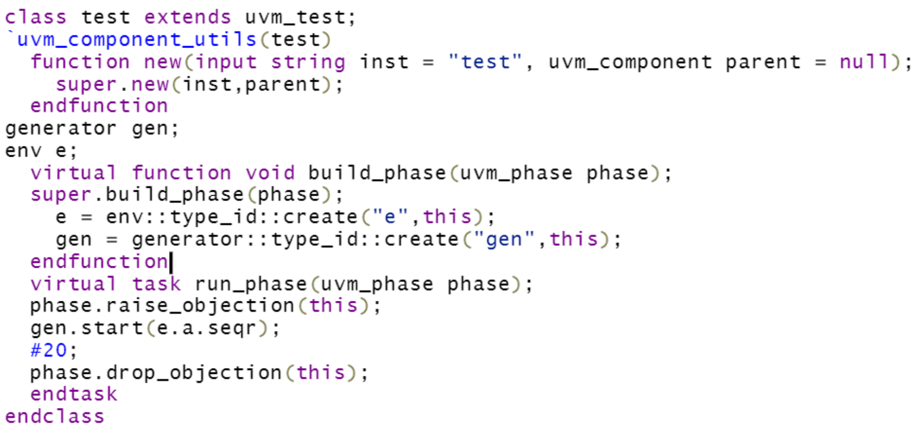
“Agent”:-

```
class agent extends uvm_agent;
`uvm_component_utils(agent)
  function new(input string inst = "agent", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
  monitor m;
  driver d;
  uvm_sequencer #(transaction) seqr;
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
    m = monitor::type_id::create("m",this);
    d = driver::type_id::create("d",this);
    seqr = uvm_sequencer #(transaction)::type_id::create("seqr",this);
  endfunction
  virtual function void connect_phase(uvm_phase phase);
  super.connect_phase(phase);
    d.seq_item_port.connect(seqr.seq_item_export);
  endfunction
endclass
```

“Environment”:-

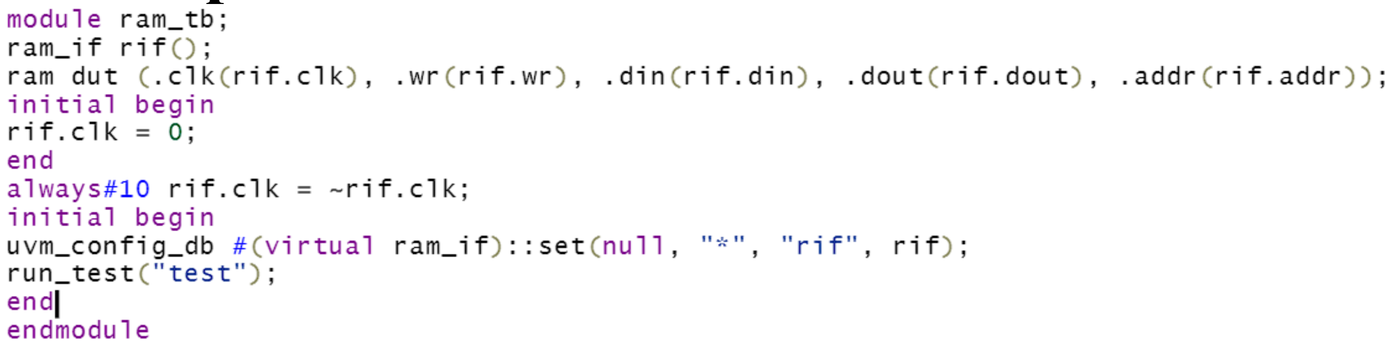
```
class env extends uvm_env;
`uvm_component_utils(env)
  function new(input string inst = "env", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
scoreboard s;
agent a;
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
    a = agent::type_id::create("a",this);
    s = scoreboard::type_id::create("s",this);
  endfunction
  virtual function void connect_phase(uvm_phase phase);
  super.connect_phase(phase);
  a.m.send.connect(s.recv);
  endfunction
endclass
```

“Test”:-



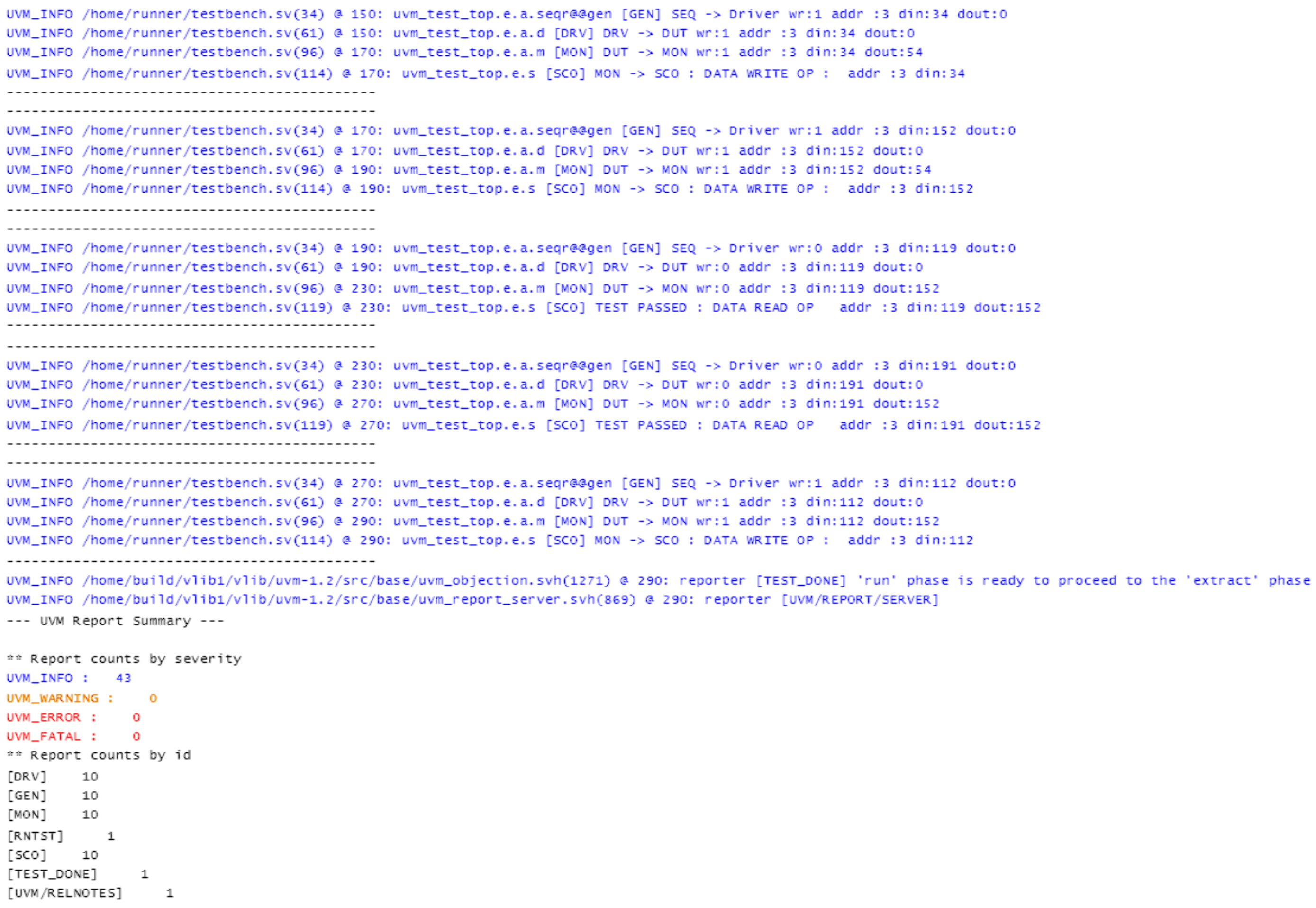
```
class test extends uvm_test;
`uvm_component_utils(test)
  function new(input string inst = "test", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
generator gen;
env e;
  virtual function void build_phase(uvm_phase phase);
  super.build_phase(phase);
    e = env::type_id::create("e",this);
    gen = generator::type_id::create("gen",this);
  endfunction
  virtual task run_phase(uvm_phase phase);
  phase.raise_objection(this);
  gen.start(e.a.seqr);
  #20;
  phase.drop_objection(this);
  endtask
endclass
```

“Tb Top”:-



```
module ram_tb;
ram_if rif();
ram dut (.clk(rif.clk), .wr(rif.wr), .din(rif.din), .dout(rif.dout), .addr(rif.addr));
initial begin
rif.clk = 0;
end
always#10 rif.clk = ~rif.clk;
initial begin
uvm_config_db #(virtual ram_if)::set(null, "*", "rif", rif);
run_test("test");
end
endmodule
```

“Result”:-



```
UVM_INFO /home/runner/testbench.sv(34) @ 150: uvm_test_top.e.a.seqr@@gen [GEN] SEQ -> Driver wr:1 addr :3 din:34 dout:0
UVM_INFO /home/runner/testbench.sv(61) @ 150: uvm_test_top.e.a.d [DRV] DRV -> DUT wr:1 addr :3 din:34 dout:0
UVM_INFO /home/runner/testbench.sv(96) @ 170: uvm_test_top.e.a.m [MON] DUT -> MON wr:1 addr :3 din:34 dout:54
UVM_INFO /home/runner/testbench.sv(114) @ 170: uvm_test_top.e.s [SCO] MON -> SCO : DATA WRITE OP :  addr :3 din:34
----------------------------------------
----------------------------------------
UVM_INFO /home/runner/testbench.sv(34) @ 170: uvm_test_top.e.a.seqr@@gen [GEN] SEQ -> Driver wr:1 addr :3 din:152 dout:0
UVM_INFO /home/runner/testbench.sv(61) @ 170: uvm_test_top.e.a.d [DRV] DRV -> DUT wr:1 addr :3 din:152 dout:0
UVM_INFO /home/runner/testbench.sv(96) @ 190: uvm_test_top.e.a.m [MON] DUT -> MON wr:1 addr :3 din:152 dout:54
UVM_INFO /home/runner/testbench.sv(114) @ 190: uvm_test_top.e.s [SCO] MON -> SCO : DATA WRITE OP :  addr :3 din:152
----------------------------------------
----------------------------------------
UVM_INFO /home/runner/testbench.sv(34) @ 190: uvm_test_top.e.a.seqr@@gen [GEN] SEQ -> Driver wr:0 addr :3 din:119 dout:0
UVM_INFO /home/runner/testbench.sv(61) @ 190: uvm_test_top.e.a.d [DRV] DRV -> DUT wr:0 addr :3 din:119 dout:0
UVM_INFO /home/runner/testbench.sv(96) @ 230: uvm_test_top.e.a.m [MON] DUT -> MON wr:0 addr :3 din:119 dout:152
UVM_INFO /home/runner/testbench.sv(119) @ 230: uvm_test_top.e.s [SCO] TEST PASSED : DATA READ OP   addr :3 din:119 dout:152
----------------------------------------
----------------------------------------
UVM_INFO /home/runner/testbench.sv(34) @ 230: uvm_test_top.e.a.seqr@@gen [GEN] SEQ -> Driver wr:0 addr :3 din:191 dout:0
UVM_INFO /home/runner/testbench.sv(61) @ 230: uvm_test_top.e.a.d [DRV] DRV -> DUT wr:0 addr :3 din:191 dout:0
UVM_INFO /home/runner/testbench.sv(96) @ 270: uvm_test_top.e.a.m [MON] DUT -> MON wr:0 addr :3 din:191 dout:152
UVM_INFO /home/runner/testbench.sv(119) @ 270: uvm_test_top.e.s [SCO] TEST PASSED : DATA READ OP   addr :3 din:191 dout:152
----------------------------------------
----------------------------------------
UVM_INFO /home/runner/testbench.sv(34) @ 270: uvm_test_top.e.a.seqr@@gen [GEN] SEQ -> Driver wr:1 addr :3 din:112 dout:0
UVM_INFO /home/runner/testbench.sv(61) @ 270: uvm_test_top.e.a.d [DRV] DRV -> DUT wr:1 addr :3 din:112 dout:0
UVM_INFO /home/runner/testbench.sv(96) @ 290: uvm_test_top.e.a.m [MON] DUT -> MON wr:1 addr :3 din:112 dout:152
UVM_INFO /home/runner/testbench.sv(114) @ 290: uvm_test_top.e.s [SCO] MON -> SCO : DATA WRITE OP :  addr :3 din:112
----------------------------------------
UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_objection.svh(1271) @ 290: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_report_server.svh(869) @ 290: reporter [UVM/REPORT/SERVER]
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   43
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[DRV]    10
[GEN]    10
[MON]    10
[RNTST]     1
[SCO]    10
[TEST_DONE]     1
[UVM/RELNOTES]     1
```