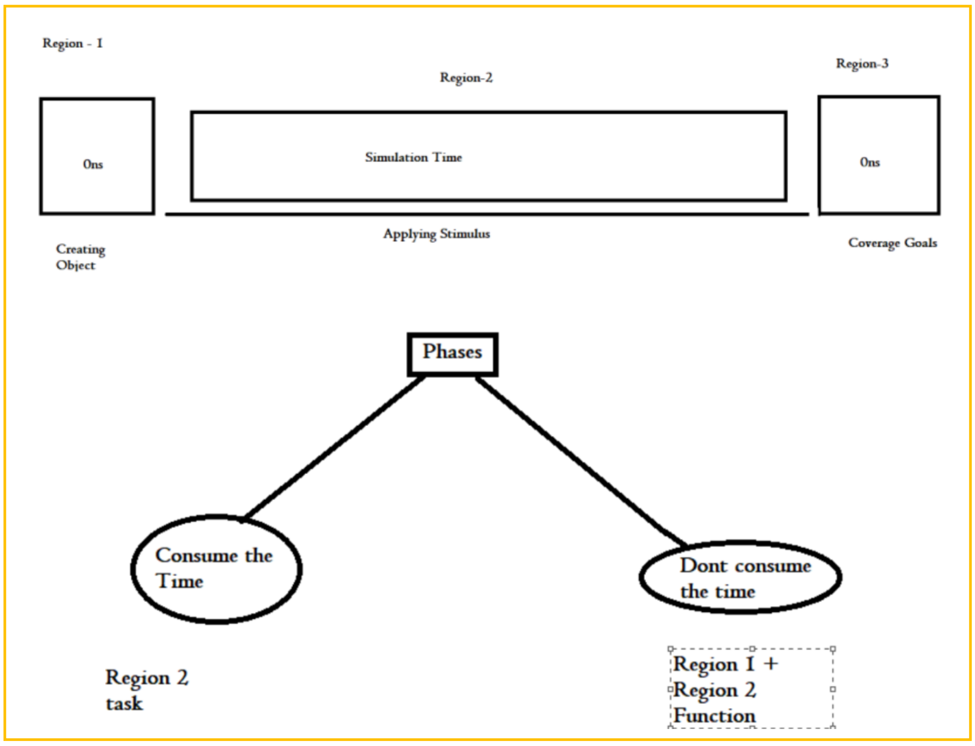
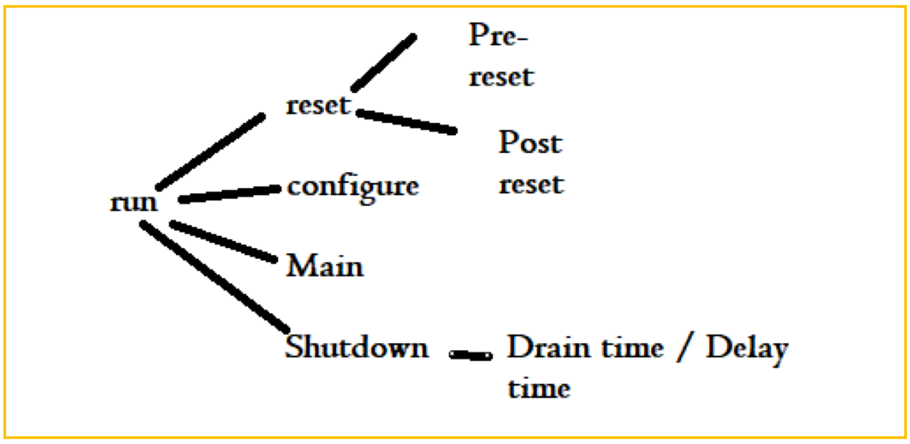
DAY-96
#100DAYSRTL

"UVM: UVM Component(Part-2)"

- UVMPhase+UVMTree are the extra added features for the uvm component.



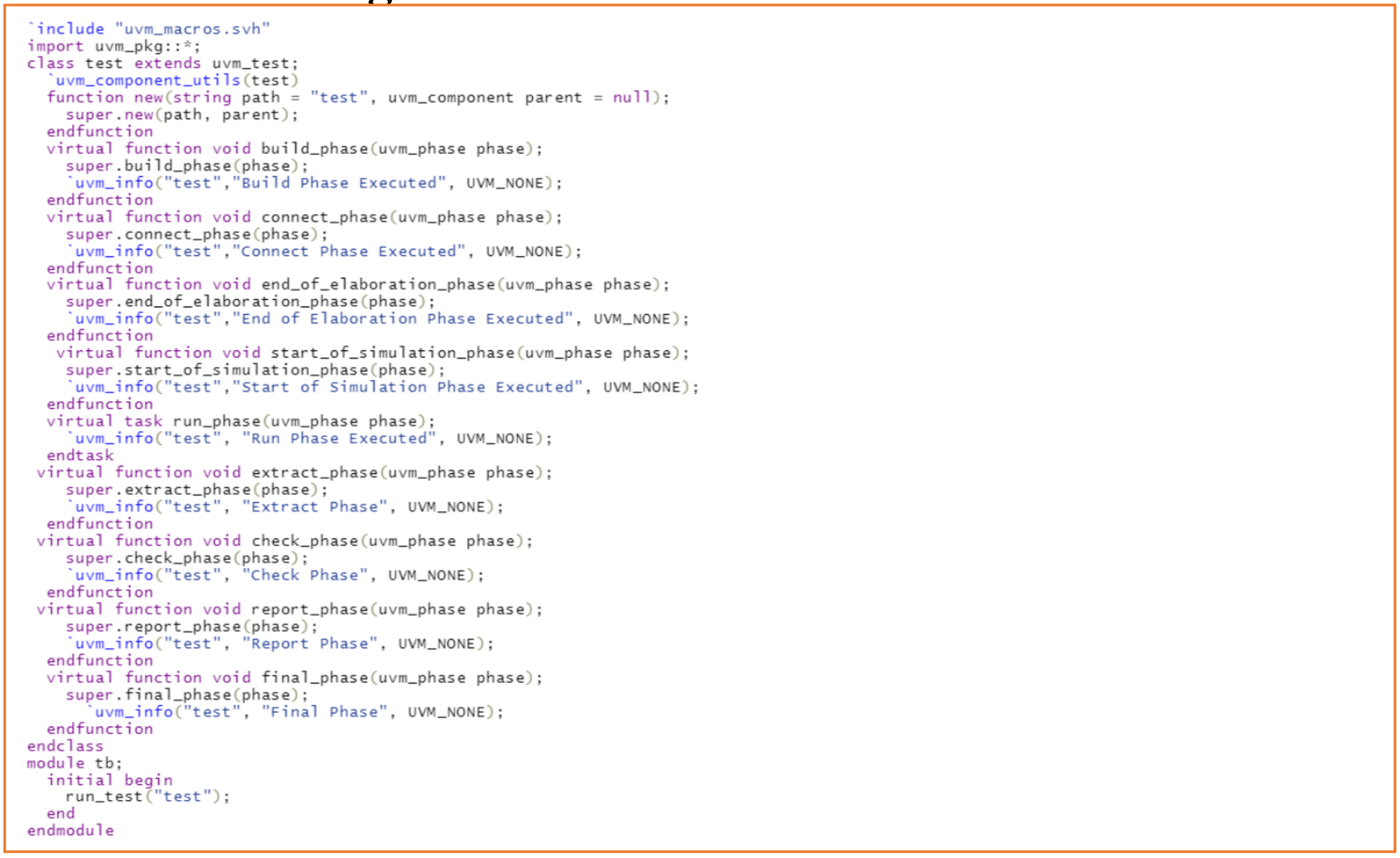
Region-1:-build, connect, End of elaboration, Start of simulation
Region-2:-Extract, check, report, Final
These regions need function as they don't need any time
Region-2:-



All these are executed in sequential order in not parallel

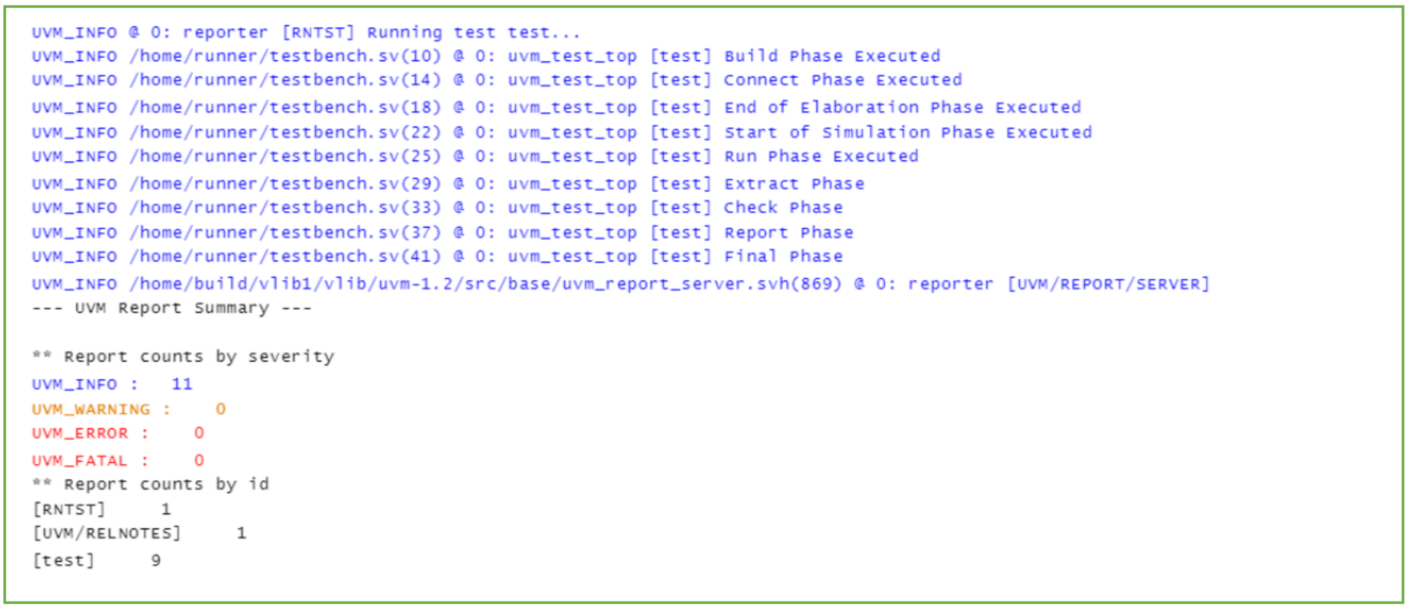
- Build phase => Create an object of class
- Connect Phase => Connection of port
- End of Elaboration => Analyze the hierarchy
- Start of simulation => to initialize the variable or to create the memory
- Run => reset -> apply stimulus -> Collect responses
- Cleanup Phases => General Report / Coverage Goal

“Code Practising”:-



```
`include "uvm_macros.svh"
import uvm_pkg::*;
class test extends uvm_test;
  `uvm_component_utils(test)
  function new(string path = "test", uvm_component parent = null);
    super.new(path, parent);
  endfunction
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
     `uvm_info("test","Build Phase Executed", UVM_NONE);
  endfunction
  virtual function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
     `uvm_info("test","Connect Phase Executed", UVM_NONE);
  endfunction
  virtual function void end_of_elaboration_phase(uvm_phase phase);
    super.end_of_elaboration_phase(phase);
     `uvm_info("test","End of Elaboration Phase Executed", UVM_NONE);
  endfunction
   virtual function void start_of_simulation_phase(uvm_phase phase);
    super.start_of_simulation_phase(phase);
     `uvm_info("test","Start of Simulation Phase Executed", UVM_NONE);
  endfunction
  virtual task run_phase(uvm_phase phase);
     `uvm_info("test", "Run Phase Executed", UVM_NONE);
  endtask
 virtual function void extract_phase(uvm_phase phase);
    super.extract_phase(phase);
     `uvm_info("test", "Extract Phase", UVM_NONE);
  endfunction
 virtual function void check_phase(uvm_phase phase);
    super.check_phase(phase);
     `uvm_info("test", "Check Phase", UVM_NONE);
  endfunction
 virtual function void report_phase(uvm_phase phase);
    super.report_phase(phase);
     `uvm_info("test", "Report Phase", UVM_NONE);
  endfunction
  virtual function void final_phase(uvm_phase phase);
    super.final_phase(phase);
       `uvm_info("test", "Final Phase", UVM_NONE);
  endfunction
endclass
module tb;
  initial begin
    run_test("test");
  end
endmodule
```

“Result”:-



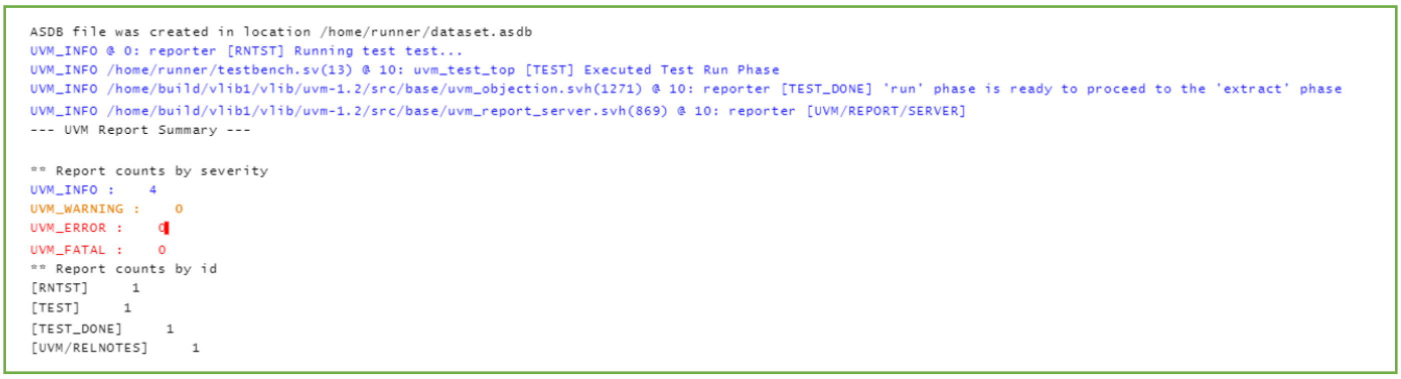
```
UVM_INFO @ 0: reporter [RNTST] Running test test...
UVM_INFO /home/runner/testbench.sv(10) @ 0: uvm_test_top [test] Build Phase Executed
UVM_INFO /home/runner/testbench.sv(14) @ 0: uvm_test_top [test] Connect Phase Executed
UVM_INFO /home/runner/testbench.sv(18) @ 0: uvm_test_top [test] End of Elaboration Phase Executed
UVM_INFO /home/runner/testbench.sv(22) @ 0: uvm_test_top [test] Start of Simulation Phase Executed
UVM_INFO /home/runner/testbench.sv(25) @ 0: uvm_test_top [test] Run Phase Executed
UVM_INFO /home/runner/testbench.sv(29) @ 0: uvm_test_top [test] Extract Phase
UVM_INFO /home/runner/testbench.sv(33) @ 0: uvm_test_top [test] Check Phase
UVM_INFO /home/runner/testbench.sv(37) @ 0: uvm_test_top [test] Report Phase
UVM_INFO /home/runner/testbench.sv(41) @ 0: uvm_test_top [test] Final Phase
UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_report_server.svh(869) @ 0: reporter [UVM/REPORT/SERVER]
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :    11
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[UVM/RELNOTES]     1
[test]     9
```

- **Simulating for more than 0 n sec:-**

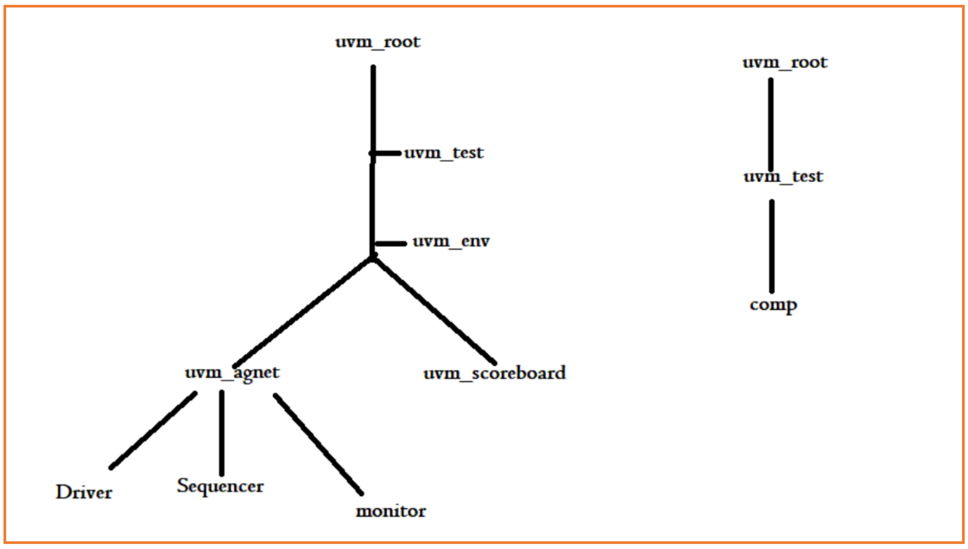
“Code Practising”:-

```
//////////////////////////////////////////////////////
`include "uvm_macros.svh"
import uvm_pkg::*;
///////////////////////////////////////////////////////////////////
class test extends uvm_test;
`uvm_component_utils(test)
  function new(input string inst = "test", uvm_component parent = null);
    super.new(inst,parent);
  endfunction
  virtual task run_phase(uvm_phase phase);
    phase.raise_objection(this);
     #10;
    `uvm_info("TEST", "Executed Test Run Phase", UVM_NONE);
    phase.drop_objection(this);
  endtask
endclass
///////////////////////////////////////////////
module tb;
initial begin
run_test("test");
end
endmodule
```

“Result”:-

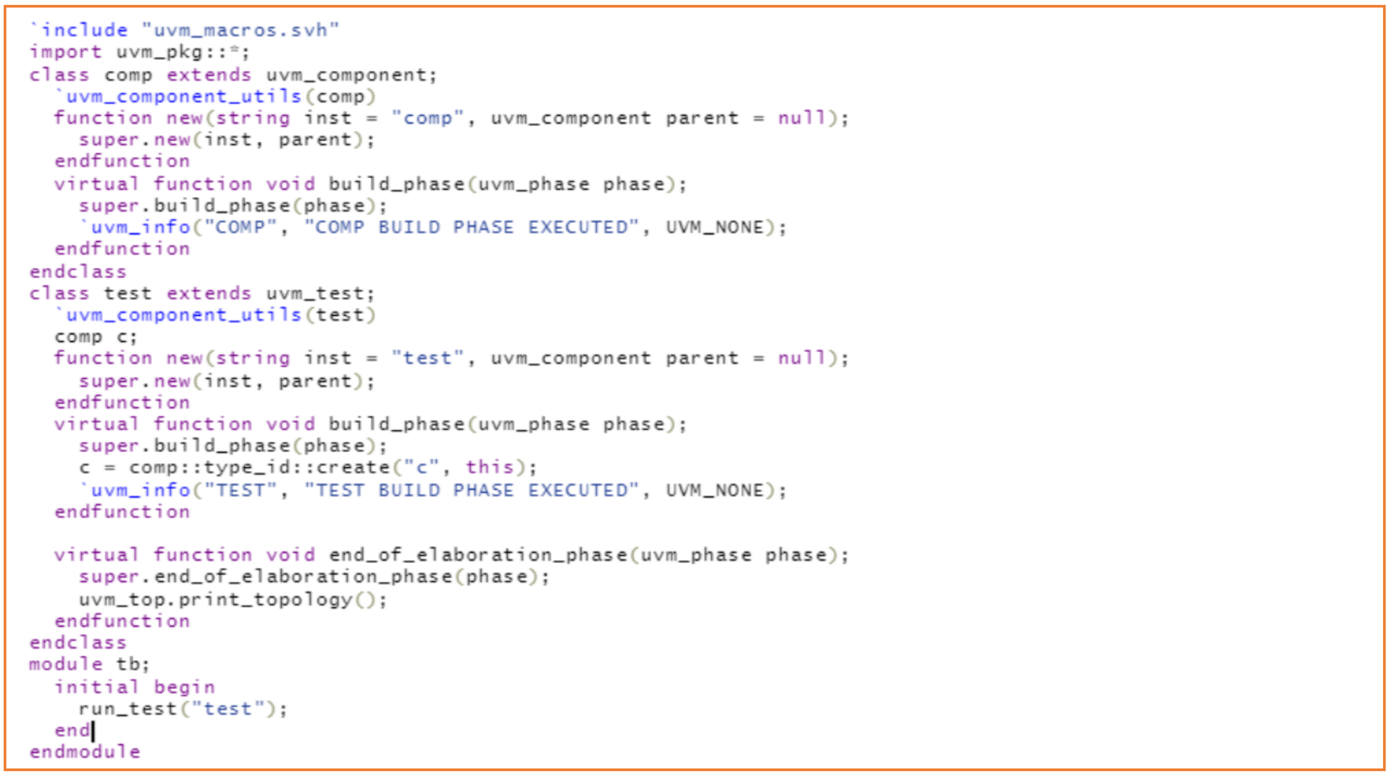


```
ASDB file was created in location /home/runner/dataset.asdb
UVM_INFO @ 0: reporter [RNTST] Running test test...
UVM_INFO /home/runner/testbench.sv(13) @ 10: uvm_test_top [TEST] Executed Test Run Phase
UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_objection.svh(1271) @ 10: reporter [TEST_DONE] '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) @ 10: reporter [UVM/REPORT/SERVER]
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :    4
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST]     1
[TEST_DONE]     1
[UVM/RELNOTES]     1
```



- By defining only one root we can achieve a single standard verification methodology.
- We can’t directly take uvm_root, so UVM has a universal class called uvm_top. Using this we can make uvm_root a parent class

“Code Practising”:-



```
`include "uvm_macros.svh"
import uvm_pkg::*;
class comp extends uvm_component;
  `uvm_component_utils(comp)
  function new(string inst = "comp", uvm_component parent = null);
    super.new(inst, parent);
  endfunction
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    `uvm_info("COMP", "COMP BUILD PHASE EXECUTED", UVM_NONE);
  endfunction
endclass
class test extends uvm_test;
  `uvm_component_utils(test)
  comp c;
  function new(string inst = "test", uvm_component parent = null);
    super.new(inst, parent);
  endfunction
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    c = comp::type_id::create("c", this);
    `uvm_info("TEST", "TEST BUILD PHASE EXECUTED", UVM_NONE);
  endfunction

  virtual function void end_of_elaboration_phase(uvm_phase phase);
    super.end_of_elaboration_phase(phase);
    uvm_top.print_topology();
  endfunction
endclass
module tb;
  initial begin
    run_test("test");
  end
endmodule
```

“Reference”:-

https://www.udemy.com/share/105G4g3@GVFxAk7r9EDrA0C0mmIhykUyDhagH9KVJF9y4xqUa1kIjvAJlngVHo6o3_b-KzeWxA==/