```
// Adder TB

// Only Driver is there in the environment. There is Sequence, Sequencer,
Monitor, Scoreboard

`include "uvm_macros.svh"
import uvm_pkg::*;

interface add_if;
      logic [3:0] a;
      logic [3:0] b;
      logic [4:0] out;

endinterface


class driver extends uvm_driver;
      `uvm_component_utils(driver)

      virtual add_if aif;

      // constructor
      function new(string path = "driver", uvm_component parent = null);
            super.new(path, parent);
      endfunction

      // build phase
      virtual function void build_phase(uvm_phase phase);
            super.build_phase(phase);

            if(!uvm_config_db#(virtual add_if)::get(this,"","aif",aif))
//uvm_test_top.e.a.d.aif
                  `uvm_error("DRV", "Unable to access Interface");
      endfunction


      // run phase
      virtual task run_phase(uvm_phase phase);
            phase.raise_objection(this);

            // generating and driving stimulus
            for(int i=0; i<=10; i++) begin
                  `uvm_info("DRV", $sformatf("Before stimulus - a = %0d, b
= %0d", aif.a,aif.b), UVM_NONE);
                  aif.a <= $random;
                  aif.b <= $random;
                  `uvm_info("DRV", $sformatf("After stimulus - a = %0d, b
= %0d", aif.a,aif.b), UVM_NONE);
                  #10;
            end
            phase.drop_objection(this);
      endtask

endclass
```

```
class agent extends uvm_agent;
      `uvm_component_utils(agent);

      driver d;

      // custom constructor
      function new(string path = "agent", uvm_component parent);
            super.new(path, parent);
      endfunction

      // build phase
      virtual function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            d = driver::type_id::create("d",this);
      endfunction
endclass: agent


class env extends uvm_env;
      `uvm_component_utils(env)

      agent a;

      function new(string path = "env", uvm_component parent);
            super.new(path, parent);
      endfunction

      virtual function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            a = agent::type_id::create("a", this);
      endfunction
endclass


class test extends uvm_test;
      `uvm_component_utils(test)

      env e;

      function new(string path = "test", uvm_component parent);
            super.new(path, parent);
      endfunction

      virtual function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            e = env::type_id::create("e", this);
      endfunction
endclass
```