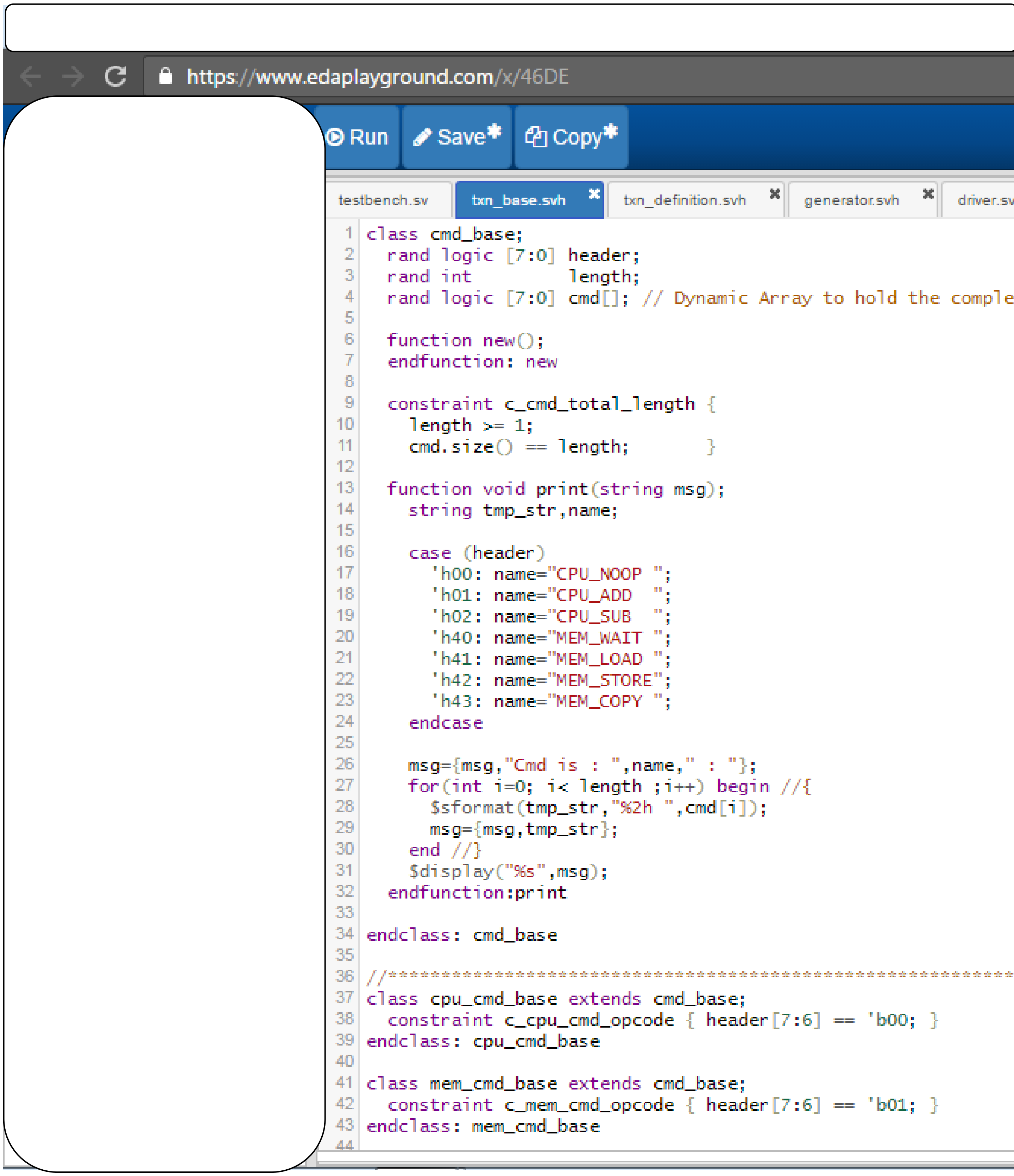
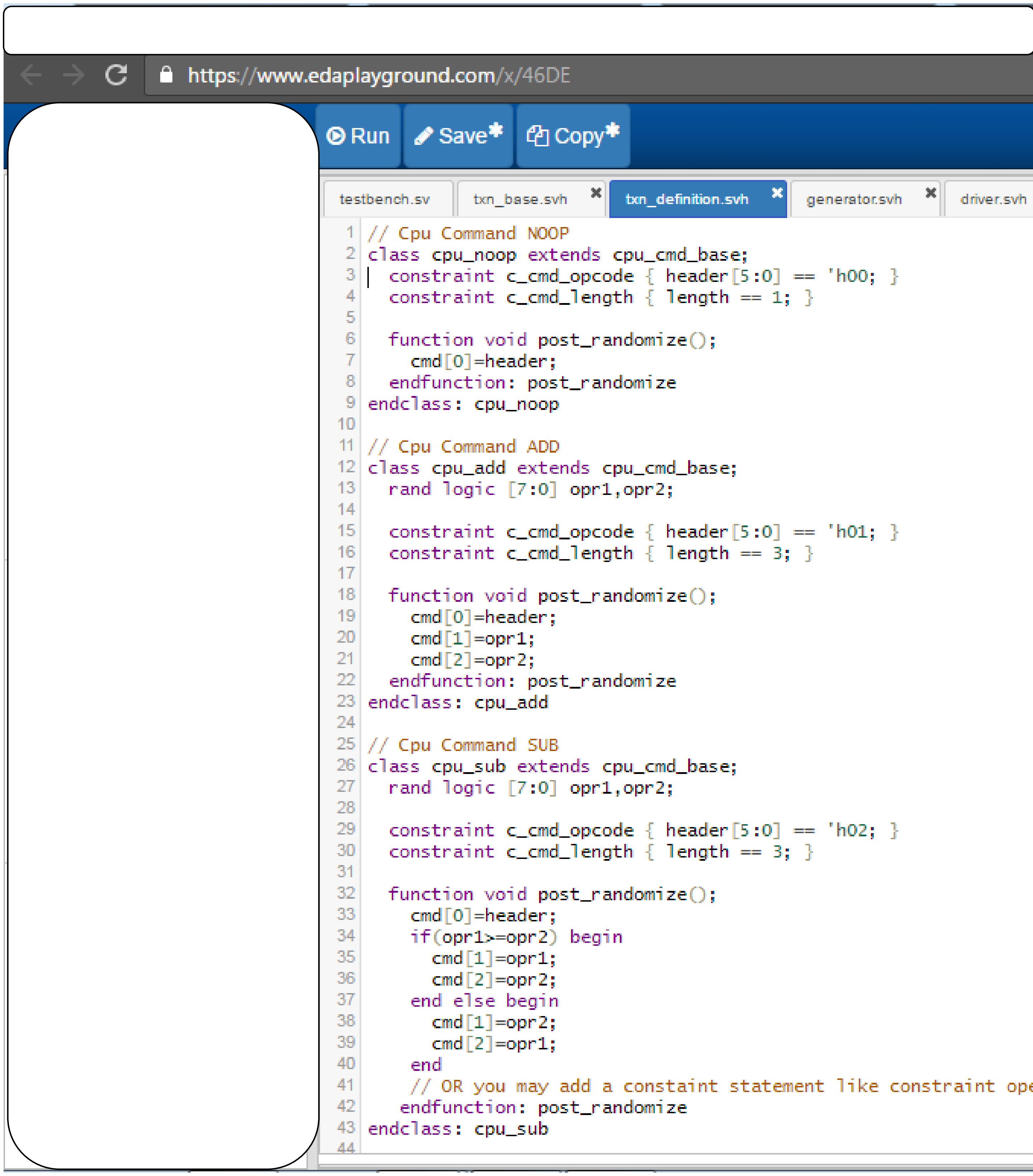
1.Txn Base Class



https://www.edaplayground.com/x/46DE

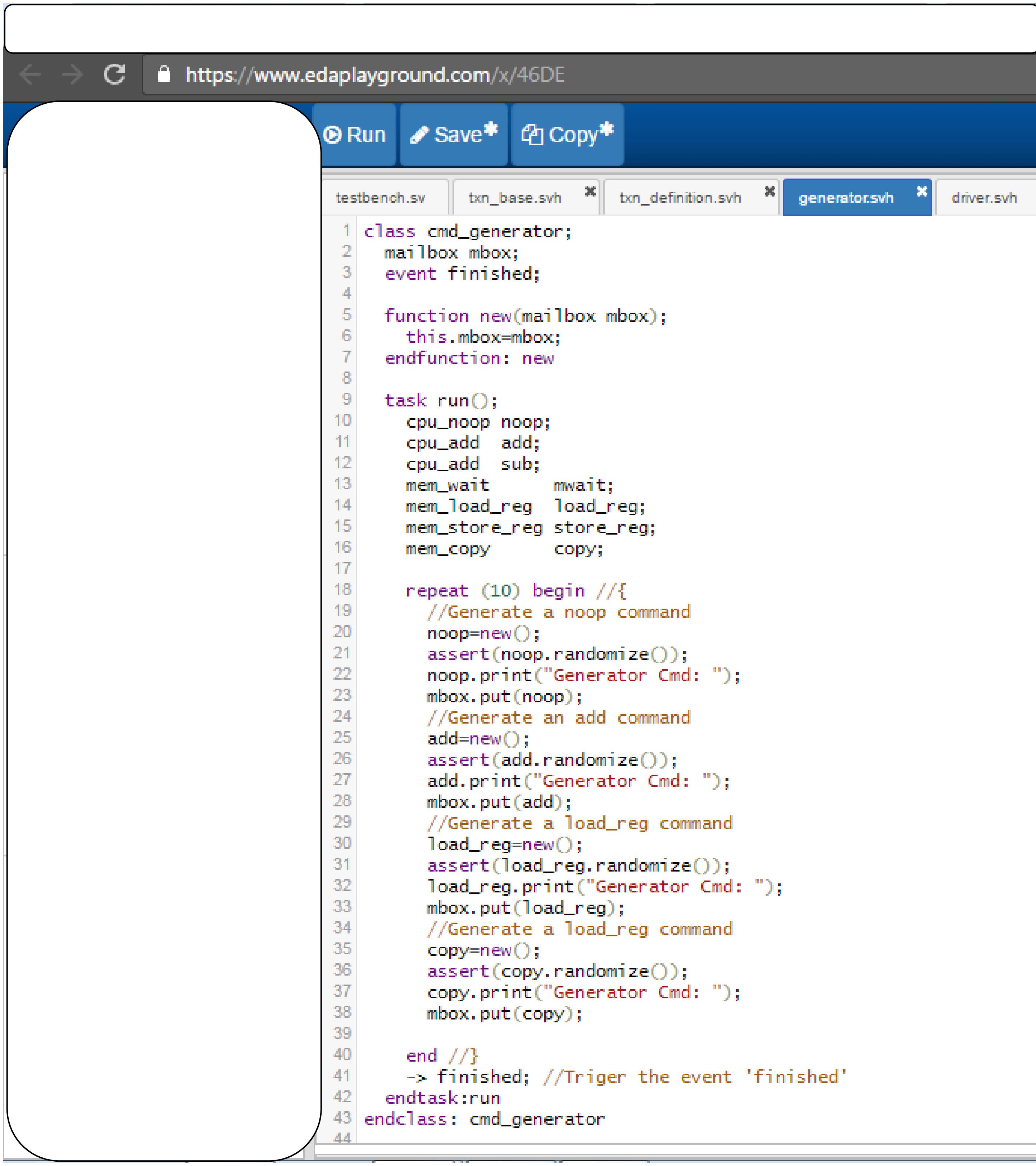
```
class cmd_base;
  rand logic [7:0] header;
  rand int         length;
  rand logic [7:0] cmd[]; // Dynamic Array to hold the comple

  function new();
  endfunction: new

  constraint c_cmd_total_length {
    length >= 1;
    cmd.size() == length;       }

  function void print(string msg);
    string tmp_str,name;

    case (header)
      'h00: name="CPU_NOOP ";
      'h01: name="CPU_ADD  ";
      'h02: name="CPU_SUB  ";
      'h40: name="MEM_WAIT ";
      'h41: name="MEM_LOAD ";
      'h42: name="MEM_STORE";
      'h43: name="MEM_COPY ";
    endcase

    msg={msg,"Cmd is : ",name," : "};
    for(int i=0; i< length ;i++) begin //{
      $sformat(tmp_str,"%2h ",cmd[i]);
      msg={msg,tmp_str};
    end //}
    $display("%s",msg);
  endfunction:print

endclass: cmd_base

//****************************************************************
class cpu_cmd_base extends cmd_base;
  constraint c_cpu_cmd_opcode { header[7:6] == 'b00; }
endclass: cpu_cmd_base

class mem_cmd_base extends cmd_base;
  constraint c_mem_cmd_opcode { header[7:6] == 'b01; }
endclass: mem_cmd_base

```

2. Transaction definition



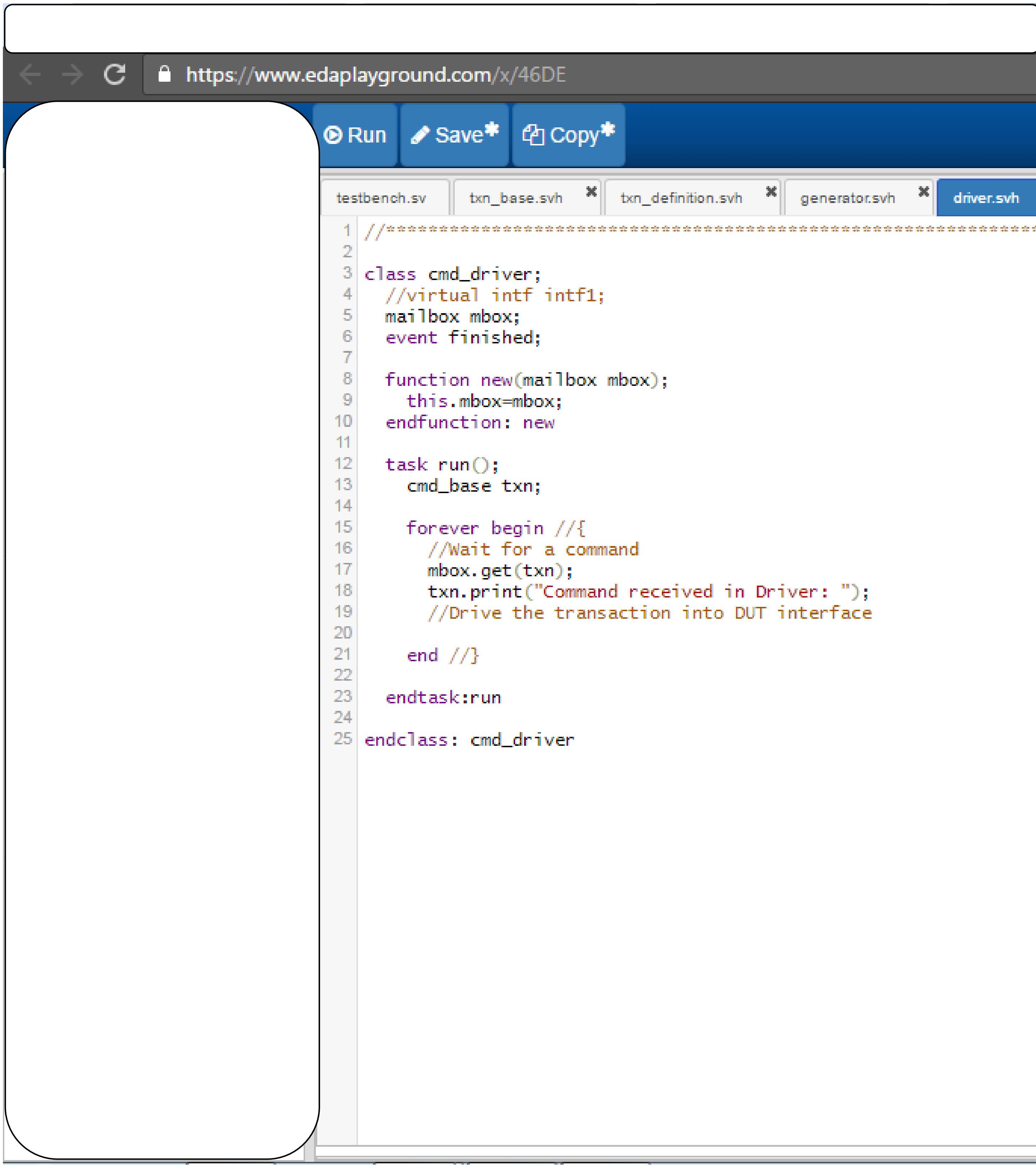
```
// Cpu Command NOOP
class cpu_noop extends cpu_cmd_base;
  constraint c_cmd_opcode { header[5:0] == 'h00; }
  constraint c_cmd_length { length == 1; }

  function void post_randomize();
    cmd[0]=header;
  endfunction: post_randomize
endclass: cpu_noop

// Cpu Command ADD
class cpu_add extends cpu_cmd_base;
  rand logic [7:0] opr1,opr2;

  constraint c_cmd_opcode { header[5:0] == 'h01; }
  constraint c_cmd_length { length == 3; }

  function void post_randomize();
    cmd[0]=header;
    cmd[1]=opr1;
    cmd[2]=opr2;
  endfunction: post_randomize
endclass: cpu_add

// Cpu Command SUB
class cpu_sub extends cpu_cmd_base;
  rand logic [7:0] opr1,opr2;

  constraint c_cmd_opcode { header[5:0] == 'h02; }
  constraint c_cmd_length { length == 3; }

  function void post_randomize();
    cmd[0]=header;
    if(opr1>=opr2) begin
      cmd[1]=opr1;
      cmd[2]=opr2;
    end else begin
      cmd[1]=opr2;
      cmd[2]=opr1;
    end
    // OR you may add a constaint statement like constraint ope
   endfunction: post_randomize
endclass: cpu_sub
```

3. Generator



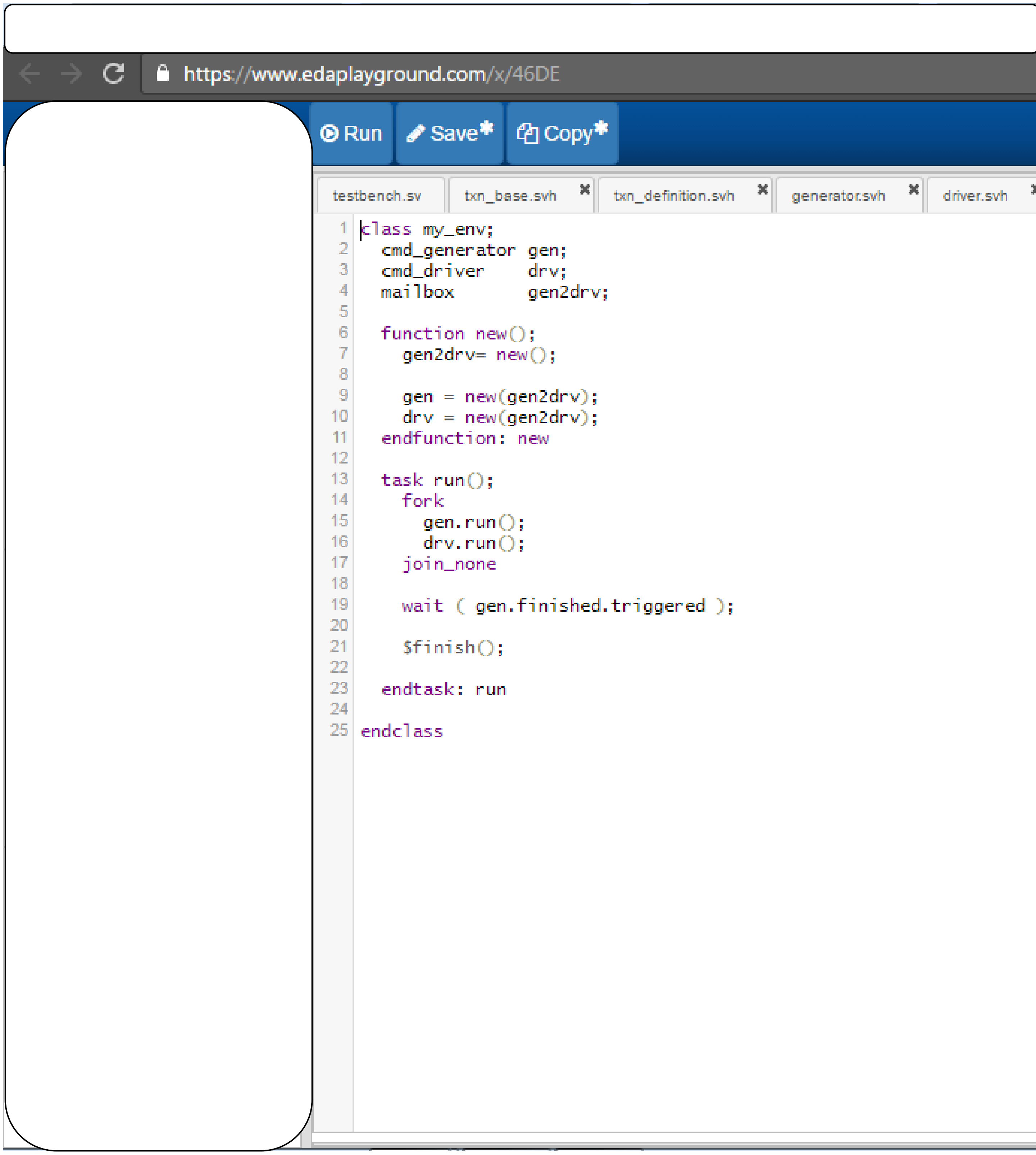
```
class cmd_generator;
  mailbox mbox;
  event finished;

  function new(mailbox mbox);
    this.mbox=mbox;
  endfunction: new

  task run();
    cpu_noop noop;
    cpu_add  add;
    cpu_add  sub;
    mem_wait      mwait;
    mem_load_reg  load_reg;
    mem_store_reg store_reg;
    mem_copy      copy;

    repeat (10) begin //{
      //Generate a noop command
      noop=new();
      assert(noop.randomize());
      noop.print("Generator Cmd: ");
      mbox.put(noop);
      //Generate an add command
      add=new();
      assert(add.randomize());
      add.print("Generator Cmd: ");
      mbox.put(add);
      //Generate a load_reg command
      load_reg=new();
      assert(load_reg.randomize());
      load_reg.print("Generator Cmd: ");
      mbox.put(load_reg);
      //Generate a load_reg command
      copy=new();
      assert(copy.randomize());
      copy.print("Generator Cmd: ");
      mbox.put(copy);

    end //}
    -> finished; //Triger the event 'finished'
  endtask:run
endclass: cmd_generator
```

4. Driver



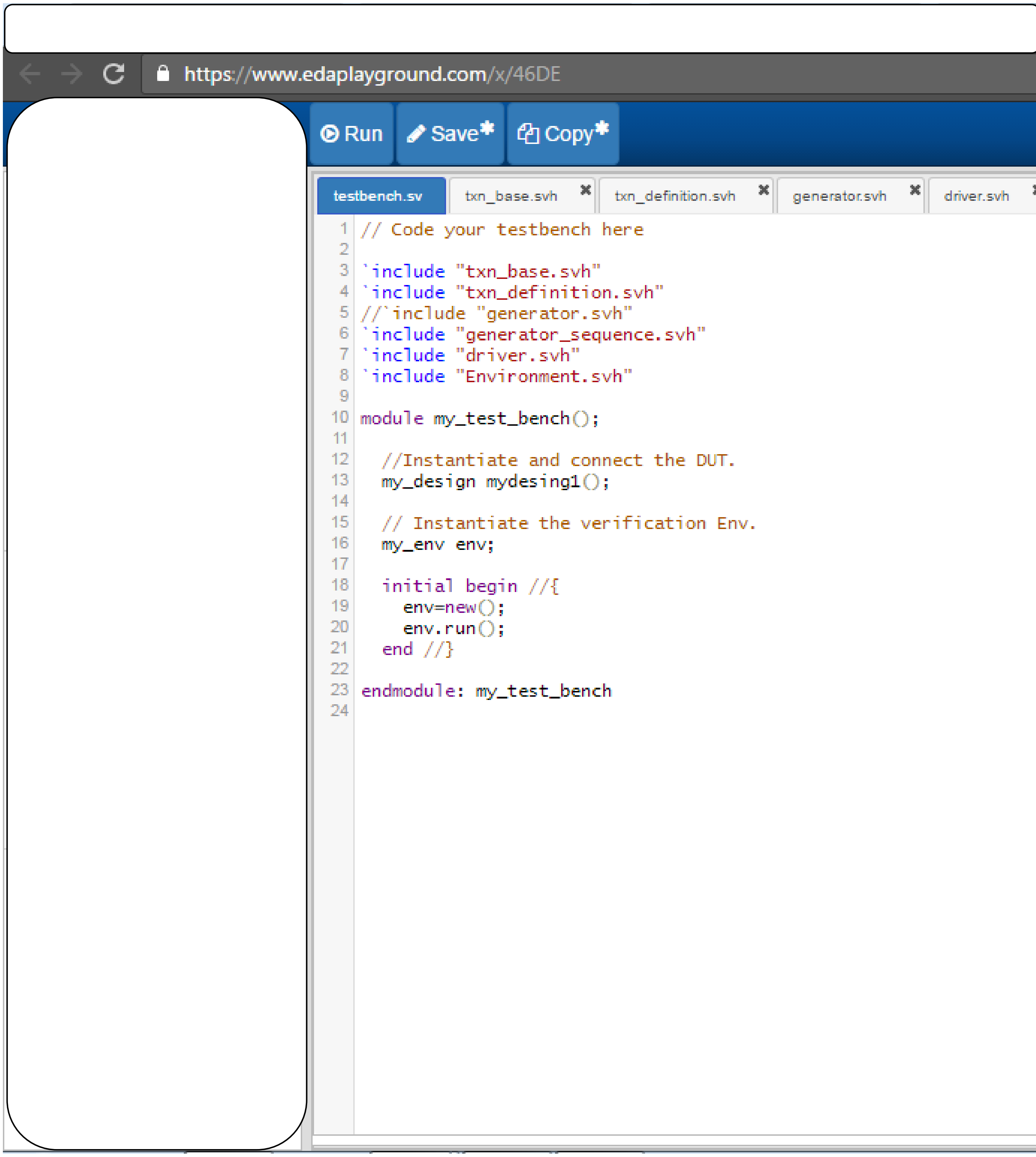
```
//**************************************************************

class cmd_driver;
  //virtual intf intf1;
  mailbox mbox;
  event finished;

  function new(mailbox mbox);
    this.mbox=mbox;
  endfunction: new

  task run();
    cmd_base txn;

    forever begin //{
      //Wait for a command
      mbox.get(txn);
      txn.print("Command received in Driver: ");
      //Drive the transaction into DUT interface

    end //}

  endtask:run

endclass: cmd_driver
```

5. Environment



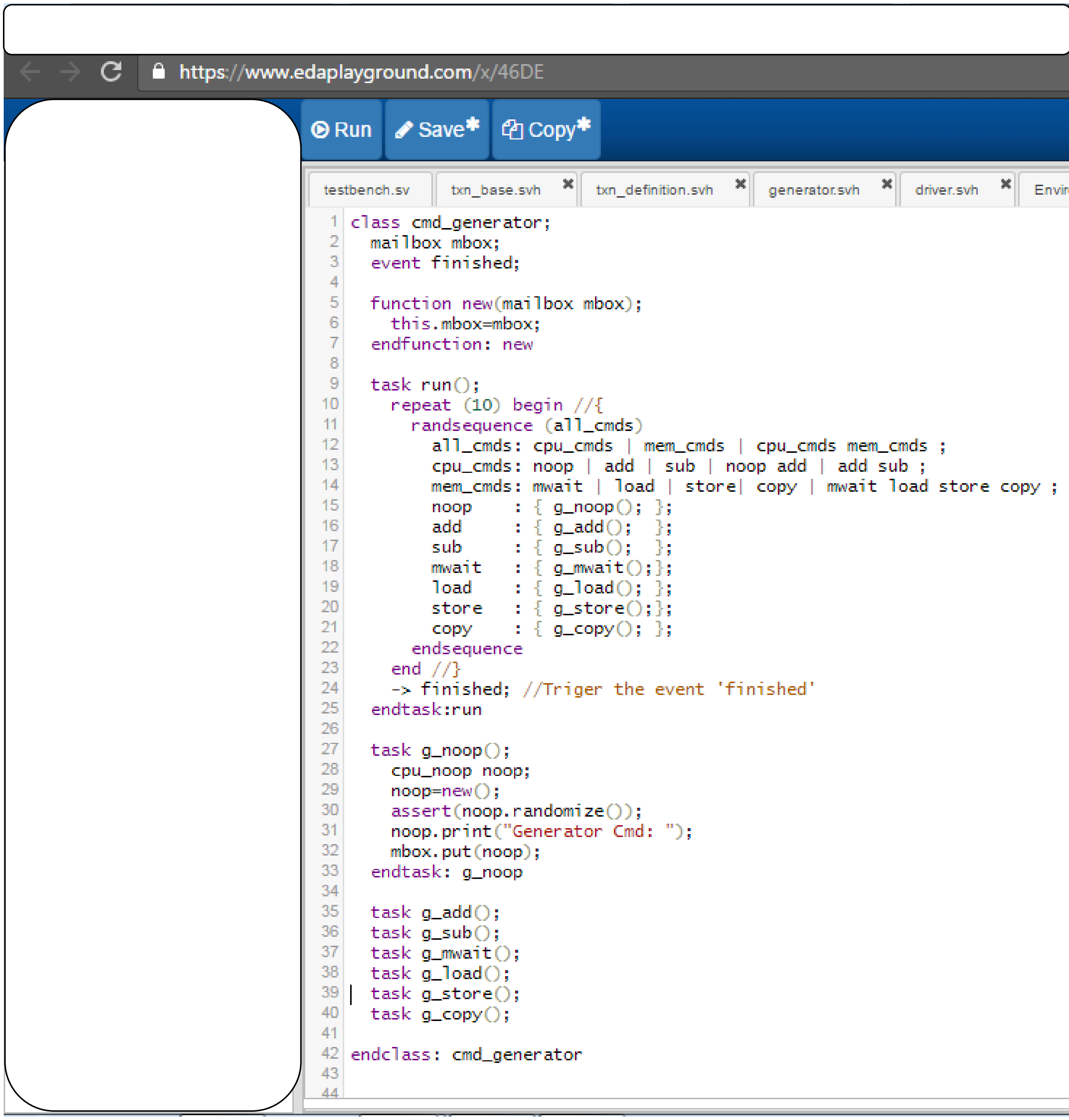
```
class my_env;
  cmd_generator gen;
  cmd_driver    drv;
  mailbox       gen2drv;

  function new();
    gen2drv= new();

    gen = new(gen2drv);
    drv = new(gen2drv);
  endfunction: new

  task run();
    fork
      gen.run();
      drv.run();
    join_none

    wait ( gen.finished.triggered );

    $finish();

  endtask: run

endclass
```

6. Test Bench



```
// Code your testbench here

`include "txn_base.svh"
`include "txn_definition.svh"
//`include "generator.svh"
`include "generator_sequence.svh"
`include "driver.svh"
`include "Environment.svh"

module my_test_bench();

  //Instantiate and connect the DUT.
  my_design mydesing1();

  // Instantiate the verification Env.
  my_env env;

  initial begin //{
    env=new();
    env.run();
  end //}

endmodule: my_test_bench
```

7. Generator using Randsequence



https://www.edaplayground.com/x/46DE

Run | Save | Copy

testbench.sv | txn_base.svh | txn_definition.svh | generator.svh | driver.svh | Envir

```
class cmd_generator;
  mailbox mbox;
  event finished;

  function new(mailbox mbox);
    this.mbox=mbox;
  endfunction: new

  task run();
    repeat (10) begin //{
      randsequence (all_cmds)
        all_cmds: cpu_cmds | mem_cmds | cpu_cmds mem_cmds ;
        cpu_cmds: noop | add | sub | noop add | add sub ;
        mem_cmds: mwait | load | store| copy | mwait load store copy ;
        noop     : { g_noop(); };
        add      : { g_add();  };
        sub      : { g_sub();  };
        mwait    : { g_mwait();};
        load     : { g_load(); };
        store    : { g_store();};
        copy     : { g_copy(); };
      endsequence
    end //}
    -> finished; //Triger the event 'finished'
  endtask:run

  task g_noop();
    cpu_noop noop;
    noop=new();
    assert(noop.randomize());
    noop.print("Generator Cmd: ");
    mbox.put(noop);
  endtask: g_noop

  task g_add();
  task g_sub();
  task g_mwait();
  task g_load();
  task g_store();
  task g_copy();

endclass: cmd_generator
```