**Module: SV for Verification**

**Section:** Testbench Basics **Task:** Creating Driver & Module tb

**Task 2 -** [**EDA**](https://www.edaplayground.com/x/NZMQ)

Creating Driver & Module tb

* **Code:**

// Author: Noman Rafiq

// Dated: Sep 12, 2024

// Driver class and tb

// Transaction class

class transaction;

rand bit a;

rand bit b;

rand bit c;

rand bit d;

bit y;

// copying method

function transaction copy();

transaction copy; // Declare a copy transaction

copy = new(); // create an object for this transaction

// create a copy for all the variables

copy.a = this.a;

copy.b = this.b;

copy.c = this.c;

copy.d = this.d;

copy.y = this.y;

return copy; // return the object handle

endfunction

endclass

// Generator Class

class generator;

transaction blueprint; // Declare a blueprint transaction

mailbox gen2drv; // Declare a gen2drv mailbox

event done; // Declare an event 'done'

// object construction

function new (mailbox mbx);

this.gen2drv = mbx; // Pass mailbox by reference

blueprint = new(); // construct a blueprint object

endfunction

// Run Task

task run();

repeat(5) begin

blueprint.randomize(); // Randomize blueprint object

gen2drv.try\_put(blueprint.copy()); // Put the blueprint copy into the shared mailbox

$display("@Generator, Values Generated :: a = %0b, b = %0b, c = %0b, d = %0b :: output: %0b", blueprint.a, blueprint.b, blueprint.c, blueprint.d, blueprint.y);

#1;

end

-> done; // Trigger the event

endtask

endclass

// Driver Class

class driver;

mailbox gen2drv; // Declare a local mailbox

transaction t; // Declare a local transacrion object to get values from mailbox

// Pass the shared mailbox

function new(mailbox mbx); // new constructor

this.gen2drv = mbx; // use the same shared mailbox

endfunction

// Run task

task run();

repeat(5) begin

t = new();

gen2drv.try\_get(t); // get transaction from the mailbox

$display("@Driver, Values Received :: a = %0b, b = %0b, c = %0b, d = %0b :: output: %0b", t.a, t.b, t.c, t.d, t.y);

#1;

end

endtask

endclass

// Test module

module tb;

transaction tr = new();

generator gen;

driver drv;

mailbox gen2drv;

initial begin

gen2drv = new();

gen = new(gen2drv);

drv = new(gen2drv);

fork

gen.run();

drv.run();

join\_any

// Wait for done event

wait(gen.done.triggered);

$display("Transactions completed Successfully!");

end

endmodule

* **Output logs:**

