

# TOP

## TEST

### ENVIRONMENT

#### GENERATOR

```
//generator code
class generator;
  rand transaction tr; //only this handle has to be
  //randomized otherwise the a,b,cin will not get randomized
  mailbox gen2drv;
  function new(mailbox gen2drv);
    this.gen2drv = gen2drv;
  endfunction
  //Implementation of the main task
  task main();
    repeat(10) begin
      tr = new();
      assert (tr.randomize());
      //after randomize packets are saved in mailbox
      gen2drv.put(tr);
      //creation of the transaction
      //display the transaction
      tr.display("GEN");
    end
  endtask
endclass
```

#### SCOREBOARD

```
//scoreboard implementation
class scoreboard;
  //declare handles
  transaction tr;
  mailbox mon2scb;
  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction
  //main sb logic
  task main();
    repeat(10) begin
      mon2scb.get(tr);
      //checker circuit
      if((tr.cout, tr.s) == tr.sctr.bctr.cin)
        $display("TEST PASSED !!!");
      else
        $error("TEST FAILED ....");
      tr.display("SCB");
    end
  endtask
endclass
```

#### TRANSACTION

```
//transaction
class transaction;
  rand bit a,b,cin;
  bit s,cout;
  function void display(string name);
    $display("%s: a = %b, b = %b, cin = %b, cout = %b, s = %b, cout = %b", name, a, b, cin, cout, s);
  endfunction
endclass
```

Each tr.

GEN2DRV

#### DRIVER

```
//driver code one of the most important component
class driver;
  transaction tr;
  mailbox gen2drv;
  virtual fa_intf vif;
  function new(mailbox gen2drv, virtual fa_intf vif);
    this.gen2drv = gen2drv;
    this.vif = vif;
  endfunction
  //main task
  task main();
    repeat(10) begin
      gen2drv.get(tr);
      //driver logic
      mailbox mon2scb;
      vif.a = tr.a;
      vif.b = tr.b;
      vif.cin = tr.cin;
      tr.display("DRV");
    end
  endtask
endclass
```

vif (I/P)

#### MONITOR

```
//monitor code
class monitor;
  //handles declaration
  transaction tr;
  mailbox mon2scb;
  virtual fa_intf vif;
  function new(mailbox mon2scb, virtual fa_intf vif);
    this.mon2scb = mon2scb;
    this.vif = vif;
  endfunction
  //main task implementation
  task main();
    repeat(10) begin
      tr = new();
      tr.a = vif.a;
      tr.b = vif.b;
      tr.cin = vif.cin;
      tr.cout = vif.cout;
      mon2scb.put(tr);
      tr.display("MON");
    end
  endtask
endclass
```

vif (O/P + I/P)

0s0 1b1 1cin=0 s=1

ENV

Other combinational circuits, full verification codes are present at [https://github.com/amitvsvravanishi04/amit\\_kvlsi\\_iitb/tree/main/FutureWiz\\_System\\_Verilog/verification\\_testbench\\_architectures\\_combinational\\_circuits](https://github.com/amitvsvravanishi04/amit_kvlsi_iitb/tree/main/FutureWiz_System_Verilog/verification_testbench_architectures_combinational_circuits)

Star out the repository if, you feel this PDF is worth understandable, any mistakes ?? Pls do comment..... Or raise an issue on GitHub.....

## DESIGN

### INTERFACE

```
//interface code
interface fa_intf;
  logic a;
  logic b;
  logic cin;
  logic s;
  logic cout;
endinterface
```

```
//module code (fa_dut.v)
module fa(input a,b,cin,
  output s,cout);
  assign {cout,s} =
    a+b+cin;
endmodule
```

cal<sup>n</sup> of op is made

Physical connection is made  
Because both the interface and the module both are  
Static in the nature so no virtual interface is required

## OUTPUT

```
`include "fa_environment.sv"
// Test
class test;
  env e;
  function new(virtual fa_intf vif);
    e = new(vif);
  endfunction
  task run();
    e.run();
  endtask
endclass
```

```
`include "fa_transaction.sv"
`include "fa_generator.sv"
`include "fa_driver.sv"
`include "fa_monitor.sv"
`include "fa_scoreboard.sv"
// Environment
class env;
  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;
  mailbox gen2drv;
  mailbox mon2scb;
  virtual fa_intf vif;
  function new(virtual fa_intf vif);
    this.vif = vif;
    gen2drv = new();
    mon2scb = new();
    gen = new(generator);
    drv = new(driver, vif);
    mon = new(monitor, vif);
    scb = new(scoreboard);
  endfunction
  task run();
    fork
      gen.main();
      drv.main();
      mon.main();
      scb.main();
    join
  endtask
endclass
```

```
@Log
$[SCB]: a = 1, b=1, cin=0, cout=1, s=0
$[DRV]: a = 1, b=1, cin=0, cout=0, s=0
$[MONITOR]: a = 1, b=1, cin=0, cout=1, s=0
TEST PASSED !!!
$[SCB]: a = 1, b=1, cin=0, cout=1, s=0
$[DRV]: a = 1, b=0, cin=1, cout=0, s=0
$[MONITOR]: a = 1, b=1, cin=0, cout=1, s=0
TEST PASSED !!!
$[SCB]: a = 1, b=1, cin=0, cout=1, s=0
$[DRV]: a = 0, b=1, cin=0, cout=0, s=0
$[MONITOR]: a = 1, b=0, cin=1, cout=1, s=0
TEST PASSED !!!
$[SCB]: a = 1, b=0, cin=1, cout=1, s=0
$[DRV]: a = 0, b=1, cin=0, cout=0, s=0
$[MONITOR]: a = 0, b=1, cin=0, cout=0, s=1
TEST PASSED !!!
$[SCB]: a = 0, b=1, cin=0, cout=0, s=1
$[DRV]: a = 0, b=0, cin=0, cout=0, s=0
$[MONITOR]: a = 0, b=1, cin=0, cout=0, s=1
TEST PASSED !!!
$[SCB]: a = 0, b=1, cin=0, cout=0, s=1
```

```
`include "fa_test.sv"
`include "fa_interface.sv"
// Top module
module top;
  fa_intf inf();
  fa
  dut(.a(inf.a), .b(inf.b), .cin(inf
    .cin), .s(inf.s), .cout(inf.cout))
  ;
  initial begin
    test t;
    t = new(inf);
    t.run();
    $finish;
  end
endmodule
```