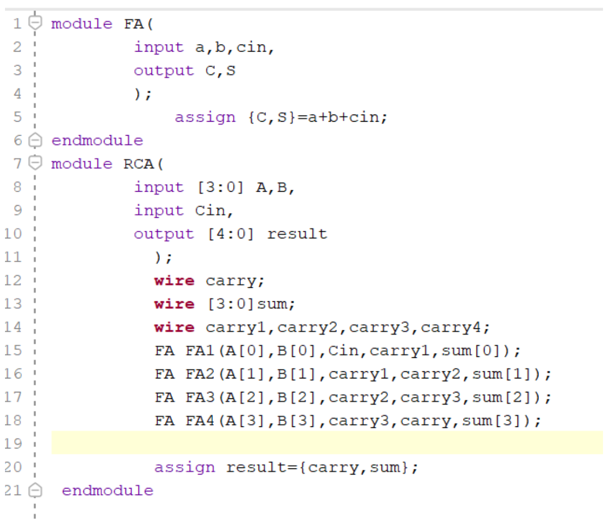
DAY-6

#100DAYSRTL

"Aim":- To Verify the 4-bit ripple carry adder using System Verilog Layered Testbenches

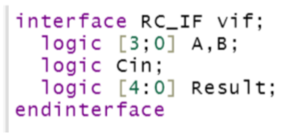
"System Verilog Testbenches":-

Design:-



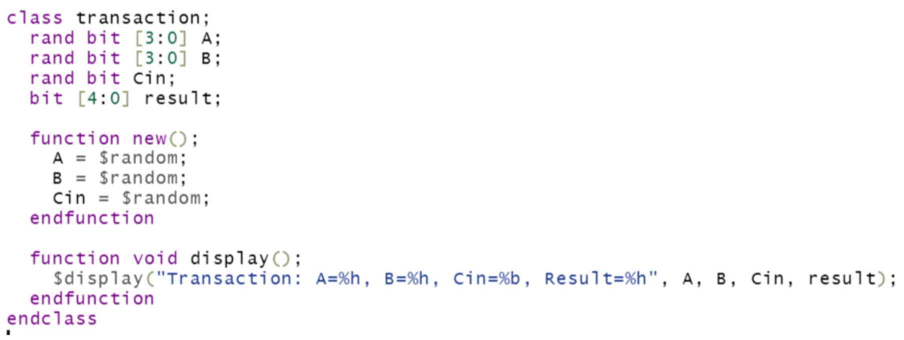
```
module FA(
        input a,b,cin,
        output C,S
        );
            assign {C,S}=a+b+cin;
endmodule
module RCA(
        input [3:0] A,B,
        input Cin,
        output [4:0] result
          );
          wire carry;
          wire [3:0]sum;
          wire carry1,carry2,carry3,carry4;
          FA FA1(A[0],B[0],Cin,carry1,sum[0]);
          FA FA2(A[1],B[1],carry1,carry2,sum[1]);
          FA FA3(A[2],B[2],carry2,carry3,sum[2]);
          FA FA4(A[3],B[3],carry3,carry,sum[3]);

          assign result={carry,sum};
 endmodule
```

Interface:-



```
interface RC_IF vif;
  logic [3;0] A,B;
  logic Cin;
  logic [4:0] Result;
endinterface
```

Transaction:-



```
class transaction;
  rand bit [3:0] A;
  rand bit [3:0] B;
  rand bit Cin;
  bit [4:0] result;

  function new();
    A = $random;
    B = $random;
    Cin = $random;
  endfunction

  function void display();
    $display("Transaction: A=%h, B=%h, Cin=%b, Result=%h", A, B, Cin, result);
  endfunction
endclass
```

Generator:-

```
class generator;
  mailbox #(transaction) mbx;
  event done;

  function new(mailbox #(transaction) mbx);
    this.mbx = mbx;
  endfunction

  task run(int count);
    repeat (count) begin
      transaction tr = new();
      tr.result = tr.A + tr.B + tr.Cin;
      mbx.put(tr);
      tr.display();
    end
    ->done;
  endtask
endclass
```

Driver:-

```
class driver;
  mailbox #(transaction) mbx;
  virtual RCA_IF vif;

  function new(mailbox #(transaction) mbx);
    this.mbx = mbx;
  endfunction

  task run();
    transaction tr;
    forever begin
      mbx.get(tr);
      vif.A = tr.A;
      vif.B = tr.B;
      vif.Cin = tr.Cin;
      #1;
      tr.result = vif.result;
      tr.display();
    end
  endtask
endclass
```

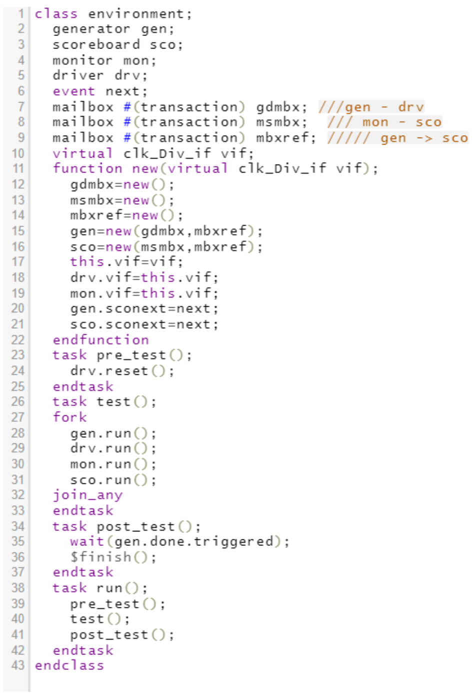
Monitor:-

```
class monitor;
  transaction tr;
  mailbox (transaction) mbx;
  virtual RC_Vif vif;
  function new(mailbox (transaction) mbx);
    this.mbx=mbx;
  endfunction
  task run();
    tr=new();
    forever begin
      tr.Result<=vif.Result;
      mbx.put(tr);
      tr.display("Mon");
    end
  endtask
endclass
```

Scoreboard:-

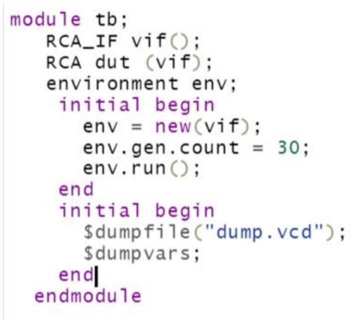
```
class scoreboard;
  mailbox #(transaction) mbx;
  mailbox #(transaction) mbxref;
  event sconext;

  function new(mailbox #(transaction) mbx, mailbox #(transaction) mbxref);
    this.mbx = mbx;
    this.mbxref = mbxref;
  endfunction

  task run();
    forever begin
      transaction tr;
      transaction trref;
      mbx.get(tr);
      mbxref.get(trref);
      tr.display();
      trref.display();
      if (tr.result === trref.result)
        $display("[SCO] : DATA MATCHED");
      else
        $display("[SCO] : DATA MISMATCHED");
      ->sconext;
    end
  endtask
endclass
```

Environment:-



```
class environment;
  generator gen;
  scoreboard sco;
  monitor mon;
  driver drv;
  event next;
  mailbox #(transaction) gdmbx; ///gen - drv
  mailbox #(transaction) msmbx;  /// mon - sco
  mailbox #(transaction) mbxref; ///// gen -> sco
  virtual clk_Div_if vif;
  function new(virtual clk_Div_if vif);
    gdmbx=new();
    msmbx=new();
    mbxref=new();
    gen=new(gdmbx,mbxref);
    sco=new(msmbx,mbxref);
    this.vif=vif;
    drv.vif=this.vif;
    mon.vif=this.vif;
    gen.sconext=next;
    sco.sconext=next;
  endfunction
  task pre_test();
    drv.reset();
  endtask
  task test();
  fork
    gen.run();
    drv.run();
    mon.run();
    sco.run();
  join_any
  endtask
  task post_test();
    wait(gen.done.triggered);
    $finish();
  endtask
  task run();
    pre_test();
    test();
    post_test();
  endtask
endclass
```

TopModule:-



```
module tb;
    RCA_IF vif();
    RCA dut (vif);
    environment env;
     initial begin
       env = new(vif);
       env.gen.count = 30;
       env.run();
     end
     initial begin
       $dumpfile("dump.vcd");
       $dumpvars;
     end
  endmodule
```