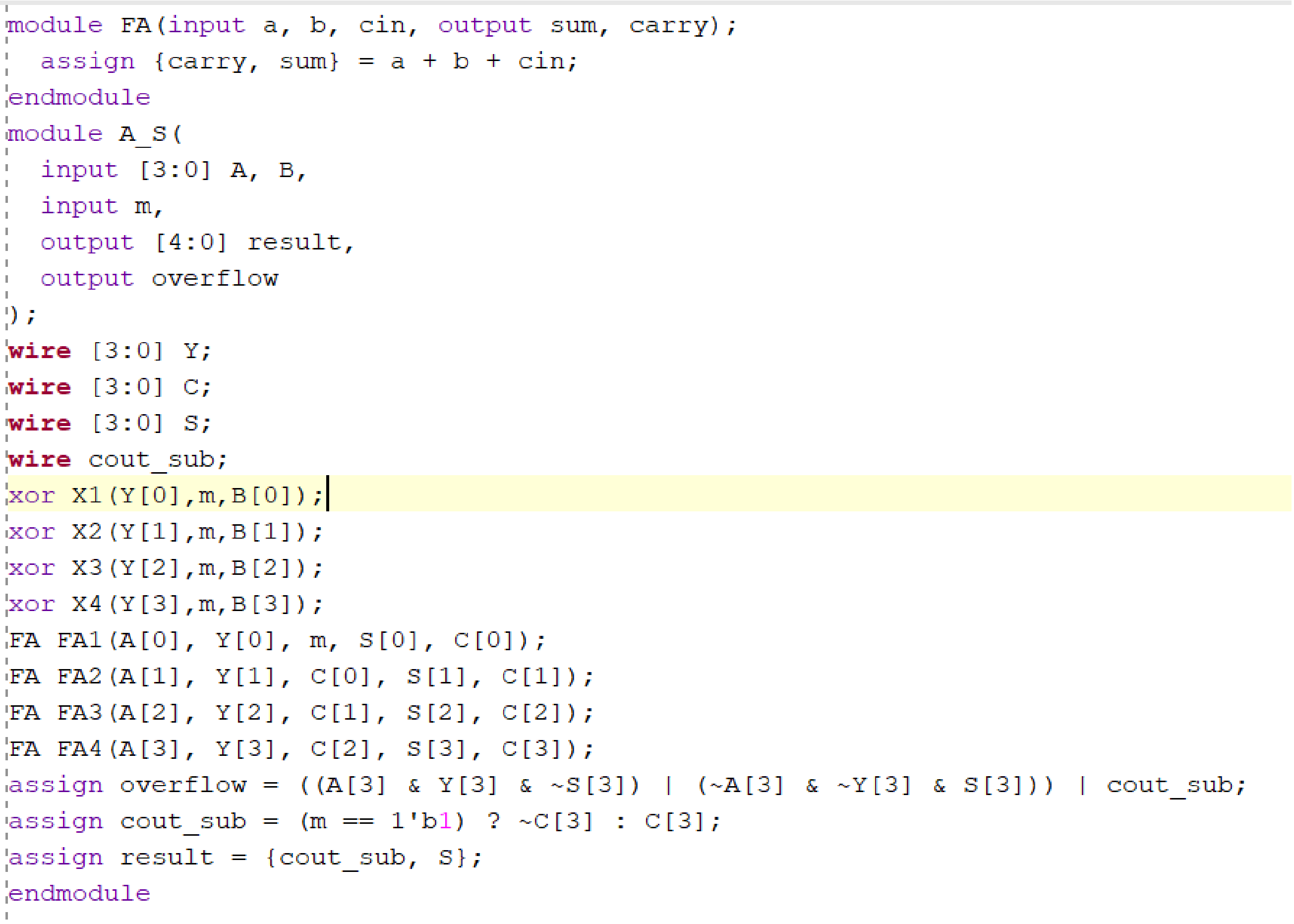
DAY-10

#100DAYSRTL

“Aim”:-To Verify the 4-bit Adder-Subtrator using System Verilog Layered Testbenches .

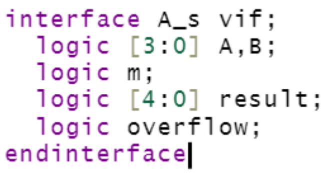
“System Verilog Testbenches”:-

Design:-



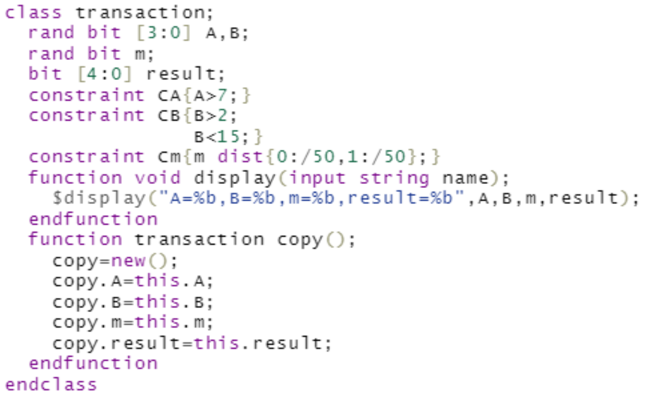
```
module FA(input a, b, cin, output sum, carry);
  assign {carry, sum} = a + b + cin;
endmodule
module A_S(
  input [3:0] A, B,
  input m,
  output [4:0] result,
  output overflow
);
wire [3:0] Y;
wire [3:0] C;
wire [3:0] S;
wire cout_sub;
xor X1(Y[0],m,B[0]);
xor X2(Y[1],m,B[1]);
xor X3(Y[2],m,B[2]);
xor X4(Y[3],m,B[3]);
FA FA1(A[0], Y[0], m, S[0], C[0]);
FA FA2(A[1], Y[1], C[0], S[1], C[1]);
FA FA3(A[2], Y[2], C[1], S[2], C[2]);
FA FA4(A[3], Y[3], C[2], S[3], C[3]);
assign overflow = ((A[3] & Y[3] & ~S[3]) | (~A[3] & ~Y[3] & S[3])) | cout_sub;
assign cout_sub = (m == 1'b1) ? ~C[3] : C[3];
assign result = {cout_sub, S};
endmodule
```

Interface:-



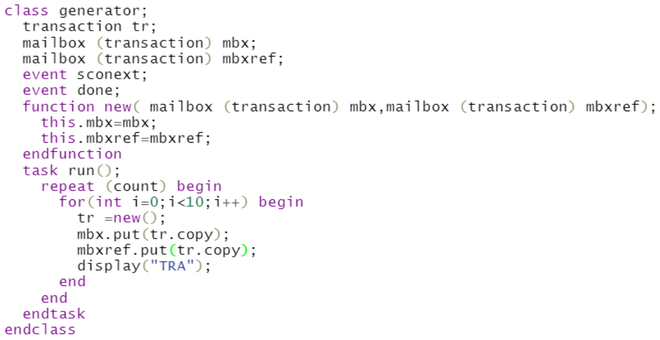
```
interface A_s vif;
  logic [3:0] A,B;
  logic m;
  logic [4:0] result;
  logic overflow;
endinterface
```

Transaction:-



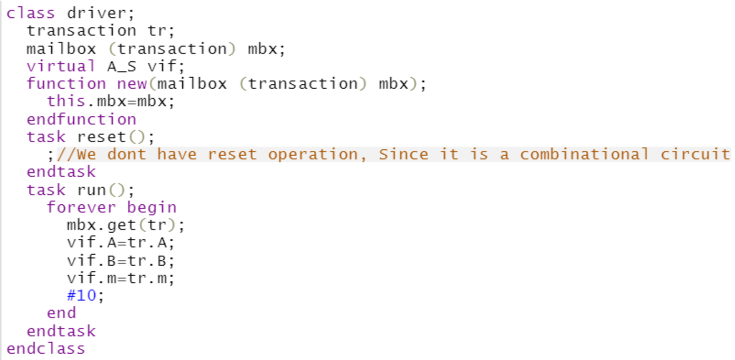
```
class transaction;
  rand bit [3:0] A,B;
  rand bit m;
  bit [4:0] result;
  constraint CA{A>7;}
  constraint CB{B>2;
                B<15;}
  constraint Cm{m dist{0:/50,1:/50};}
  function void display(input string name);
    $display("A=%b,B=%b,m=%b,result=%b",A,B,m,result);
  endfunction
  function transaction copy();
    copy=new();
    copy.A=this.A;
    copy.B=this.B;
    copy.m=this.m;
    copy.result=this.result;
  endfunction
endclass
```

Generator:-



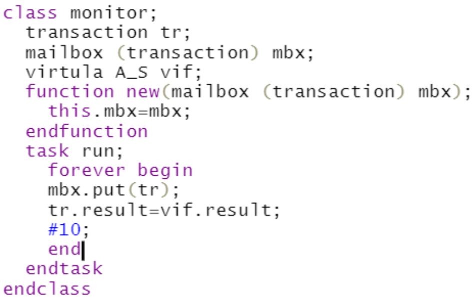
```
class generator;
  transaction tr;
  mailbox (transaction) mbx;
  mailbox (transaction) mbxref;
  event sconext;
  event done;
  function new( mailbox (transaction) mbx,mailbox (transaction) mbxref);
    this.mbx=mbx;
    this.mbxref=mbxref;
  endfunction
  task run();
    repeat (count) begin
      for(int i=0;i<10;i++) begin
        tr =new();
        mbx.put(tr.copy);
        mbxref.put(tr.copy);
        display("TRA");
      end
    end
  endtask
endclass
```

Driver:-



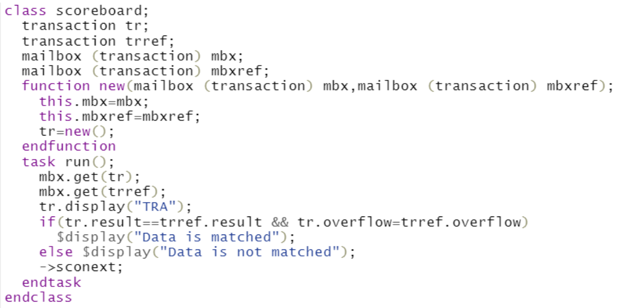
```
class driver;
  transaction tr;
  mailbox (transaction) mbx;
  virtual A_S vif;
  function new(mailbox (transaction) mbx);
    this.mbx=mbx;
  endfunction
  task reset();
    ;//We dont have reset operation, Since it is a combinational circuit
  endtask
  task run();
    forever begin
      mbx.get(tr);
      vif.A=tr.A;
      vif.B=tr.B;
      vif.m=tr.m;
      #10;
    end
  endtask
endclass
```

Monitor:-



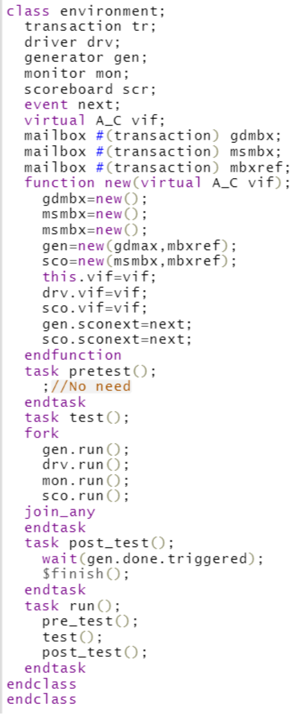
```
class monitor;
  transaction tr;
  mailbox (transaction) mbx;
  virtula A_S vif;
  function new(mailbox (transaction) mbx);
    this.mbx=mbx;
  endfunction
  task run;
    forever begin
    mbx.put(tr);
    tr.result=vif.result;
    #10;
    end
  endtask
endclass
```

Scoreboard:-



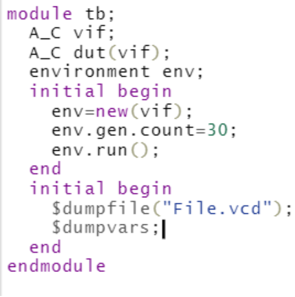
```
class scoreboard;
  transaction tr;
  transaction trref;
  mailbox (transaction) mbx;
  mailbox (transaction) mbxref;
  function new(mailbox (transaction) mbx,mailbox (transaction) mbxref);
    this.mbx=mbx;
    this.mbxref=mbxref;
    tr=new();
  endfunction
  task run();
    mbx.get(tr);
    mbx.get(trref);
    tr.display("TRA");
    if(tr.result==trref.result && tr.overflow=trref.overflow)
      $display("Data is matched");
    else $display("Data is not matched");
    ->sconext;
  endtask
endclass
```

Environment:-



```
class environment;
  transaction tr;
  driver drv;
  generator gen;
  monitor mon;
  scoreboard scr;
  event next;
  virtual A_C vif;
  mailbox #(transaction) gdmbx;
  mailbox #(transaction) msmbx;
  mailbox #(transaction) mbxref;
  function new(virtual A_C vif);
    gdmbx=new();
    msmbx=new();
    msmbx=new();
    gen=new(gdmax,mbxref);
    sco=new(msmbx,mbxref);
    this.vif=vif;
    drv.vif=vif;
    sco.vif=vif;
    gen.sconext=next;
    sco.sconext=next;
  endfunction
  task pretest();
    ;//No need
  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
endclass
```

Top Module:-



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