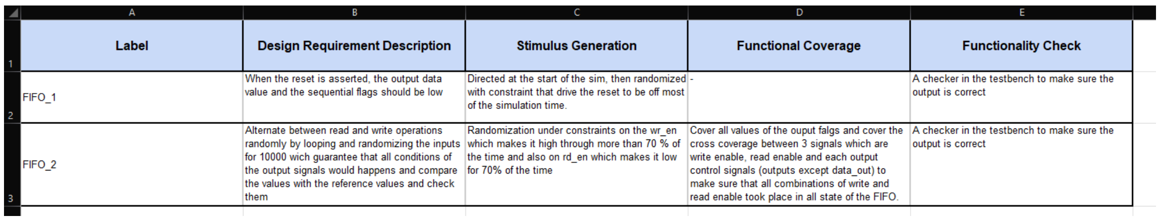
Synchronous FIFO

Verification of synchronous FIFO using SVA

1- verification plan



Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted, the output data value and the sequential flags should be low	Directed at the start of the sim, then randomized with constraint that drive the reset to be off most of the simulation time.	-	A checker in the testbench to make sure the output is correct
FIFO_2	Alternate between read and write operations randomly by looping and randomizing the inputs for 10000 wich guarantee that all conditions of the output signals would happens and compare the values with the reference values and check them	Randomization under constraints on the wr_en which makes it high through more than 70 % of the time and also on rd_en which makes it low for 70% of the time	Cover all values of the ouput falgs and cover the cross coverage between 3 signals which are write enable, read enable and each output control signals (outputs except data_out) to make sure that all combinations of write and read enable took place in all state of the FIFO.	A checker in the testbench to make sure the output is correct

2- bugs were in design

bugs fixed in FIFO project

1- in first always block

- add wr_ack and overflow signals to rst

2- in second always block

- add data_out to rst

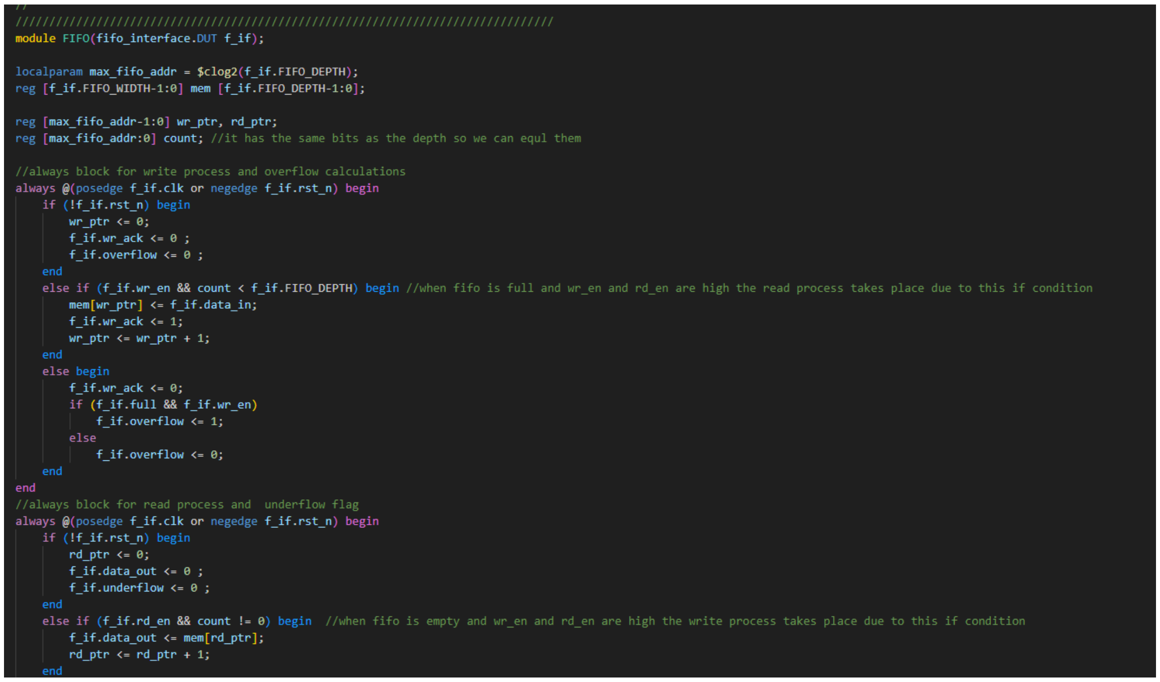
3- in third always block

- adding the priority conditions in counter logic

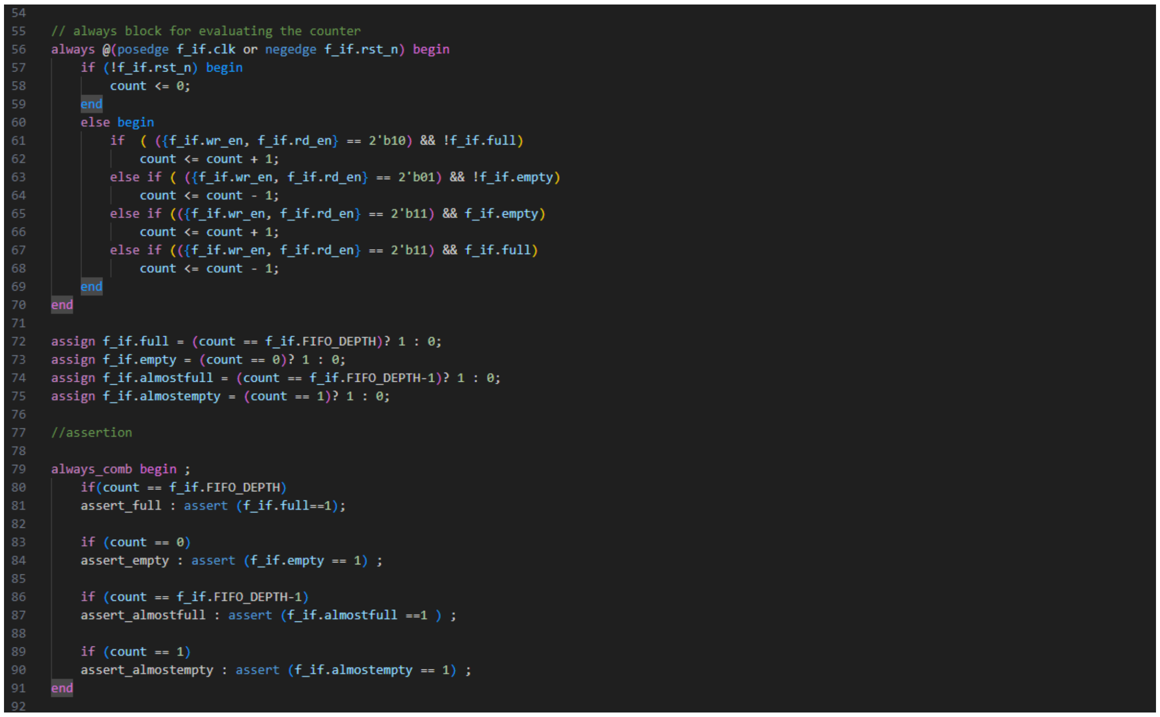
4- in the combinational output flags conditions

- **adjusting the conditions of the flags (almost_full)**

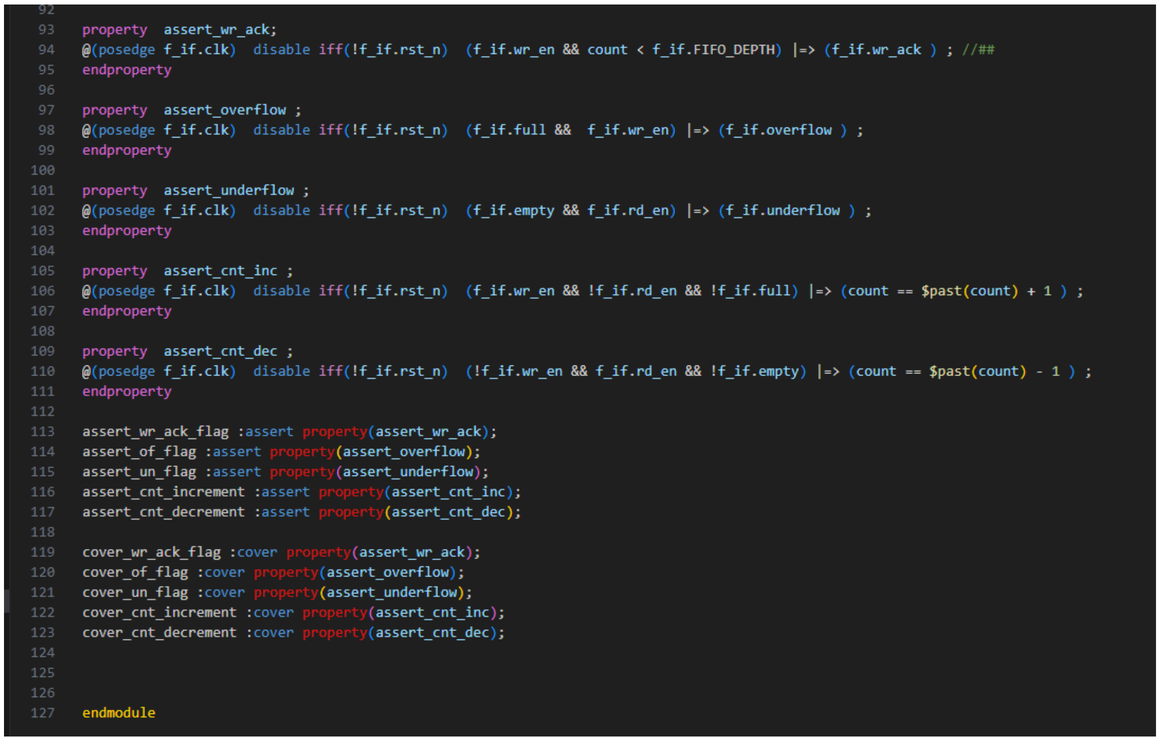
3- design fixing the bugs and the asseritions



```verilog
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(fifo_interface.DUT f_if);

localparam max_fifo_addr = $clog2(f_if.FIFO_DEPTH);
reg [f_if.FIFO_WIDTH-1:0] mem [f_if.FIFO_DEPTH-1:0];

reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
reg [max_fifo_addr:0] count; //it has the same bits as the depth so we can equl them

//always block for write process and overflow calculations
always @(posedge f_if.clk or negedge f_if.rst_n) begin
    if (!f_if.rst_n) begin
        wr_ptr <= 0;
        f_if.wr_ack <= 0 ;
        f_if.overflow <= 0 ;
    end
    else if (f_if.wr_en && count < f_if.FIFO_DEPTH) begin //when fifo is full and wr_en and rd_en are high the read process takes place due to this if condition
        mem[wr_ptr] <= f_if.data_in;
        f_if.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        f_if.wr_ack <= 0;
        if (f_if.full && f_if.wr_en)
            f_if.overflow <= 1;
        else
            f_if.overflow <= 0;
    end
end
//always block for read process and  underflow flag
always @(posedge f_if.clk or negedge f_if.rst_n) begin
    if (!f_if.rst_n) begin
        rd_ptr <= 0;
        f_if.data_out <= 0 ;
        f_if.underflow <= 0 ;
    end
    else if (f_if.rd_en && count != 0) begin  //when fifo is empty and wr_en and rd_en are high the write process takes place due to this if condition
        f_if.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
```

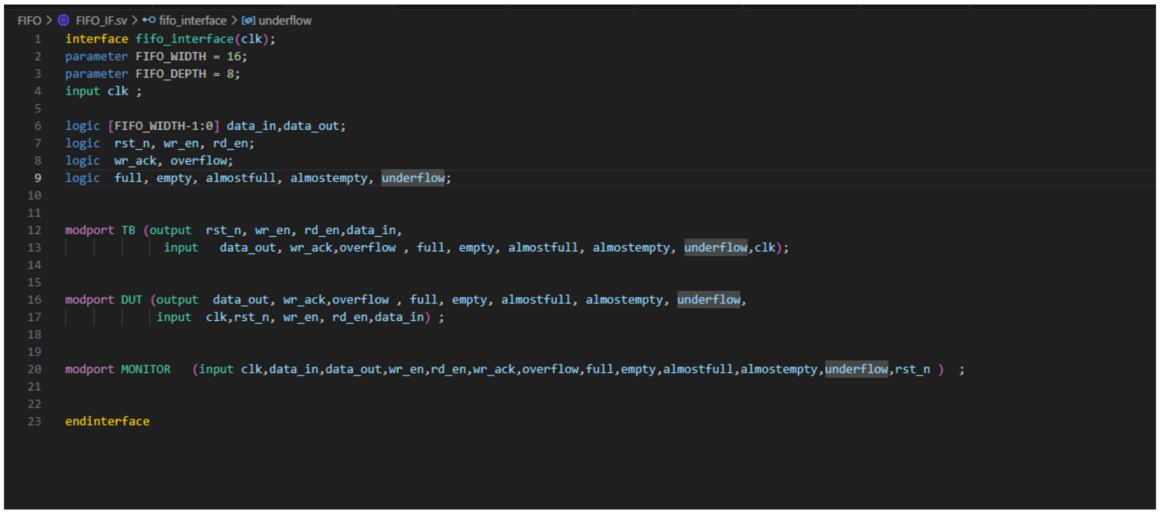


```verilog

 // always block for evaluating the counter
 always @(posedge f_if.clk or negedge f_if.rst_n) begin
     if (!f_if.rst_n) begin
         count <= 0;
     end
     else begin
         if  ( ({f_if.wr_en, f_if.rd_en} == 2'b10) && !f_if.full)
             count <= count + 1;
         else if ( ({f_if.wr_en, f_if.rd_en} == 2'b01) && !f_if.empty)
             count <= count - 1;
         else if (({f_if.wr_en, f_if.rd_en} == 2'b11) && f_if.empty)
             count <= count + 1;
         else if (({f_if.wr_en, f_if.rd_en} == 2'b11) && f_if.full)
             count <= count - 1;
     end
 end

 assign f_if.full = (count == f_if.FIFO_DEPTH)? 1 : 0;
 assign f_if.empty = (count == 0)? 1 : 0;
 assign f_if.almostfull = (count == f_if.FIFO_DEPTH-1)? 1 : 0;
 assign f_if.almostempty = (count == 1)? 1 : 0;

 //assertion

 always_comb begin ;
     if(count == f_if.FIFO_DEPTH)
     assert_full : assert (f_if.full==1);

     if (count == 0)
     assert_empty : assert (f_if.empty == 1) ;

     if (count == f_if.FIFO_DEPTH-1)
     assert_almostfull : assert (f_if.almostfull ==1 ) ;

     if (count == 1)
     assert_almostempty : assert (f_if.almostempty == 1) ;
 end

```



```
property  assert_wr_ack;
@(posedge f_if.clk)  disable iff(!f_if.rst_n)  (f_if.wr_en && count < f_if.FIFO_DEPTH) |=> (f_if.wr_ack ) ; //##
endproperty

property  assert_overflow ;
@(posedge f_if.clk)  disable iff(!f_if.rst_n)  (f_if.full &&  f_if.wr_en) |=> (f_if.overflow ) ;
endproperty

property  assert_underflow ;
@(posedge f_if.clk)  disable iff(!f_if.rst_n)  (f_if.empty && f_if.rd_en) |=> (f_if.underflow ) ;
endproperty

property  assert_cnt_inc ;
@(posedge f_if.clk)  disable iff(!f_if.rst_n)  (f_if.wr_en && !f_if.rd_en && !f_if.full) |=> (count == $past(count) + 1 ) ;
endproperty

property  assert_cnt_dec ;
@(posedge f_if.clk)  disable iff(!f_if.rst_n)  (!f_if.wr_en && f_if.rd_en && !f_if.empty) |=> (count == $past(count) - 1 ) ;
endproperty

assert_wr_ack_flag :assert property(assert_wr_ack);
assert_of_flag :assert property(assert_overflow);
assert_un_flag :assert property(assert_underflow);
assert_cnt_increment :assert property(assert_cnt_inc);
assert_cnt_decrement :assert property(assert_cnt_dec);

cover_wr_ack_flag :cover property(assert_wr_ack);
cover_of_flag :cover property(assert_overflow);
cover_un_flag :cover property(assert_underflow);
cover_cnt_increment :cover property(assert_cnt_inc);
cover_cnt_decrement :cover property(assert_cnt_dec);



endmodule
```

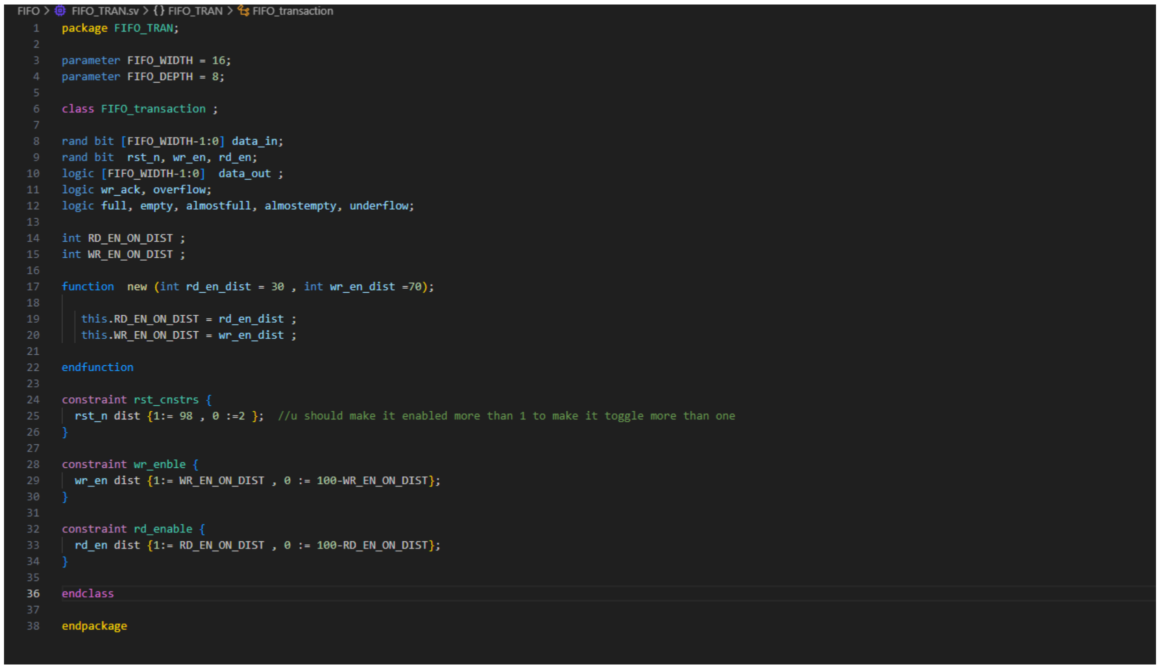
4- the interface module



FIFO > FIFO_IF.sv > fifo_interface > underflow

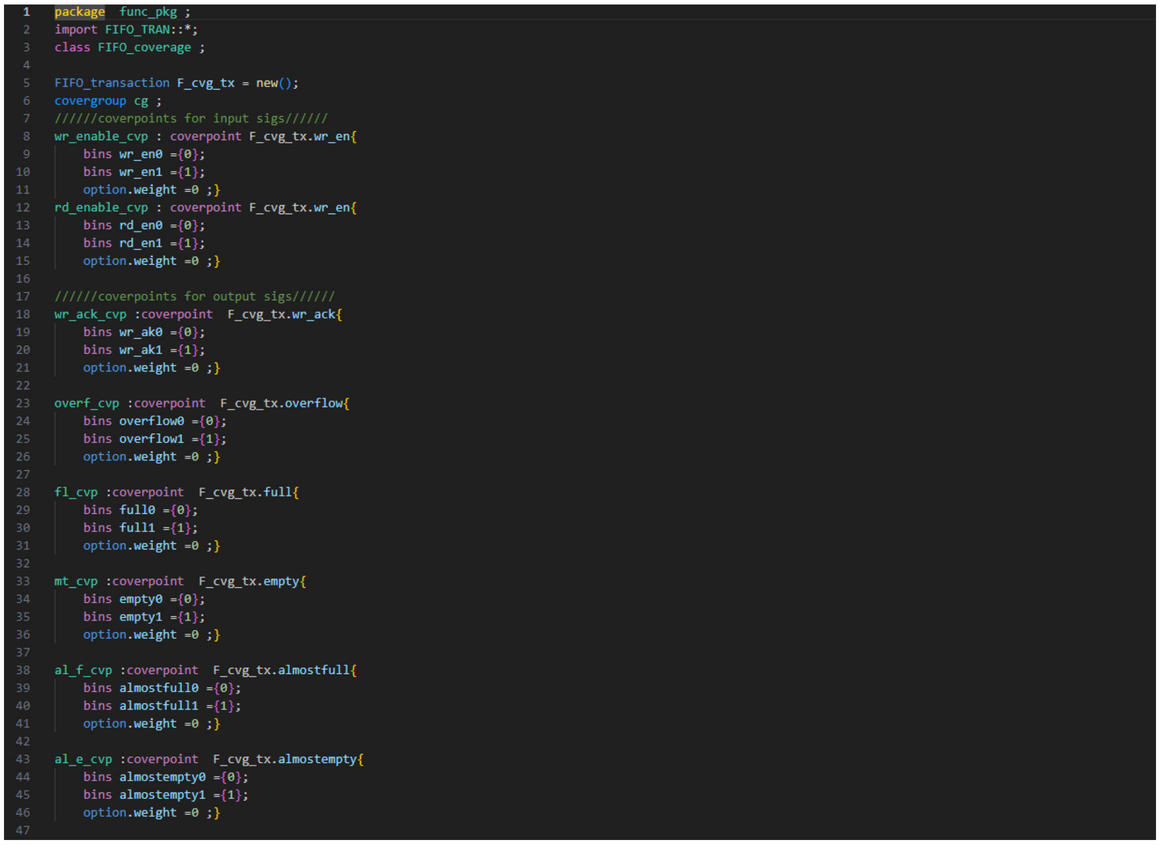
```
interface fifo_interface(clk);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
input clk ;

logic [FIFO_WIDTH-1:0] data_in,data_out;
logic  rst_n, wr_en, rd_en;
logic  wr_ack, overflow;
logic  full, empty, almostfull, almostempty, underflow;


modport TB (output  rst_n, wr_en, rd_en,data_in,
              input   data_out, wr_ack,overflow , full, empty, almostfull, almostempty, underflow,clk);


modport DUT (output  data_out, wr_ack,overflow , full, empty, almostfull, almostempty, underflow,
             input  clk,rst_n, wr_en, rd_en,data_in) ;


modport MONITOR   (input clk,data_in,data_out,wr_en,rd_en,wr_ack,overflow,full,empty,almostfull,almostempty,underflow,rst_n )  ;


endinterface
```

5- FIFO transaction package

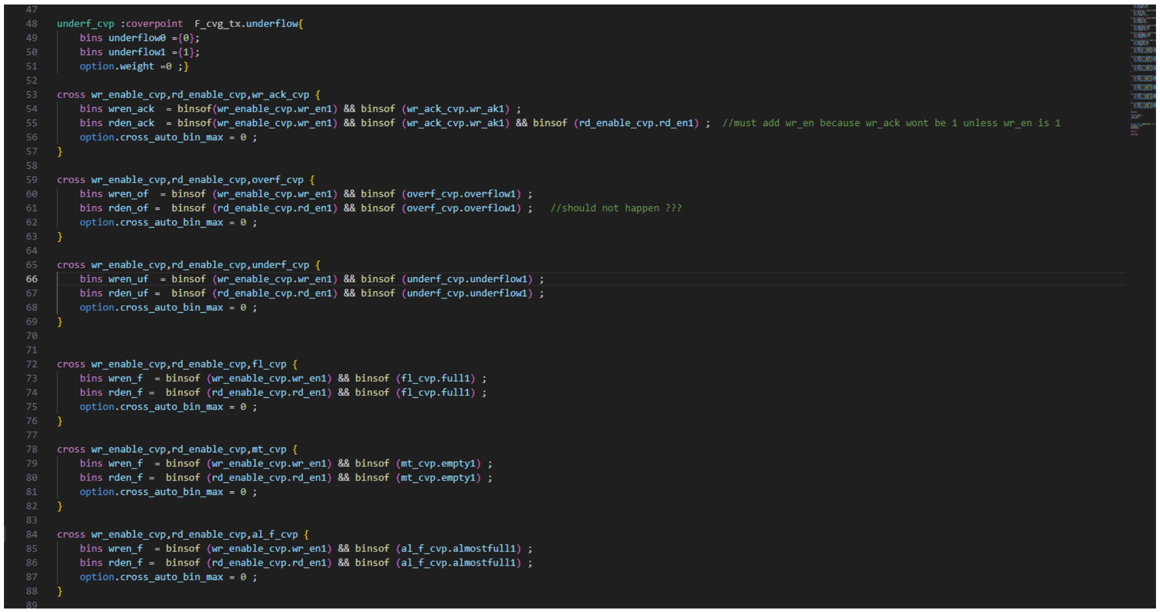

FIFO > FIFO_TRAN.sv > {} FIFO_TRAN > FIFO_transaction

```
package FIFO_TRAN;

parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

class FIFO_transaction ;

rand bit [FIFO_WIDTH-1:0] data_in;
rand bit  rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0]  data_out ;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;

int RD_EN_ON_DIST ;
int WR_EN_ON_DIST ;

function  new (int rd_en_dist = 30 , int wr_en_dist =70);

   this.RD_EN_ON_DIST = rd_en_dist ;
   this.WR_EN_ON_DIST = wr_en_dist ;

endfunction

constraint rst_cnstrs {
  rst_n dist {1:= 98 , 0 :=2 };  //u should make it enabled more than 1 to make it toggle more than one
}

constraint wr_enble {
  wr_en dist {1:= WR_EN_ON_DIST , 0 := 100-WR_EN_ON_DIST};
}

constraint rd_enable {
  rd_en dist {1:= RD_EN_ON_DIST , 0 := 100-RD_EN_ON_DIST};
}

endclass

endpackage
```

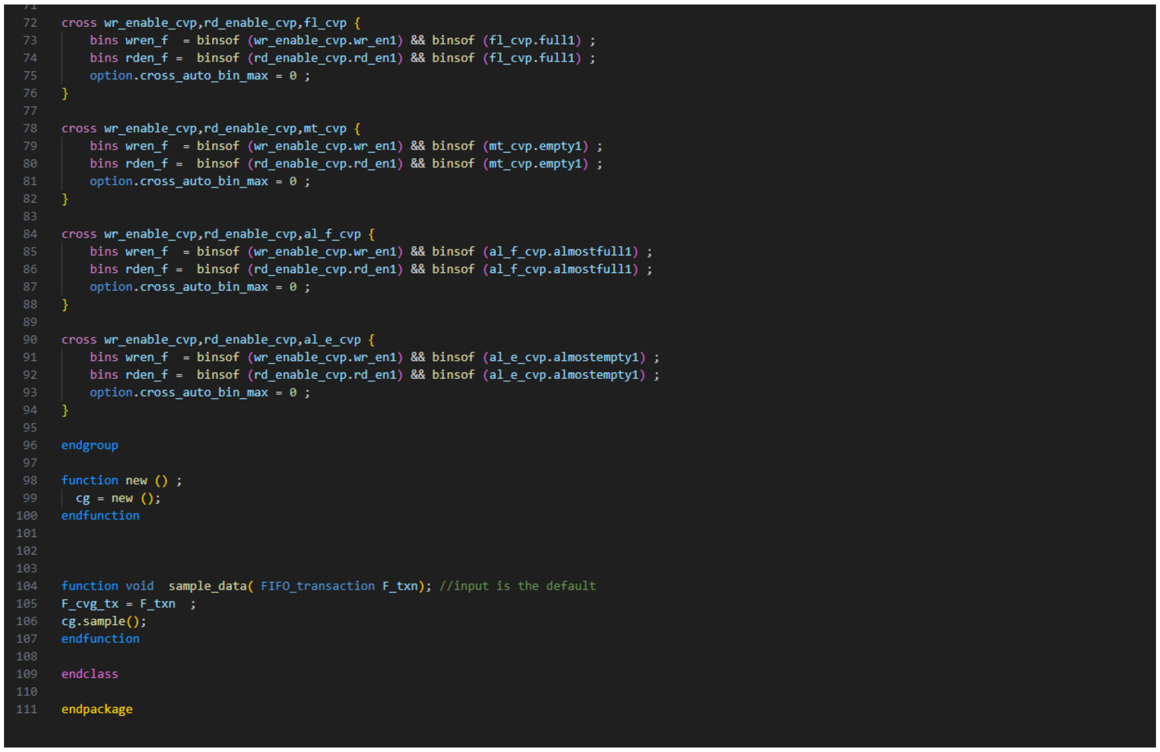
6- Functional coverage package



```
package  func_pkg ;
import FIFO_TRAN::*;
class FIFO_coverage ;

FIFO_transaction F_cvg_tx = new();
covergroup cg ;
//////coverpoints for input sigs//////
wr_enable_cvp : coverpoint F_cvg_tx.wr_en{
    bins wr_en0 ={0};
    bins wr_en1 ={1};
    option.weight =0 ;}
rd_enable_cvp : coverpoint F_cvg_tx.wr_en{
    bins rd_en0 ={0};
    bins rd_en1 ={1};
    option.weight =0 ;}

//////coverpoints for output sigs//////
wr_ack_cvp :coverpoint  F_cvg_tx.wr_ack{
    bins wr_ak0 ={0};
    bins wr_ak1 ={1};
    option.weight =0 ;}

overf_cvp :coverpoint  F_cvg_tx.overflow{
    bins overflow0 ={0};
    bins overflow1 ={1};
    option.weight =0 ;}

fl_cvp :coverpoint  F_cvg_tx.full{
    bins full0 ={0};
    bins full1 ={1};
    option.weight =0 ;}

mt_cvp :coverpoint  F_cvg_tx.empty{
    bins empty0 ={0};
    bins empty1 ={1};
    option.weight =0 ;}

al_f_cvp :coverpoint  F_cvg_tx.almostfull{
    bins almostfull0 ={0};
    bins almostfull1 ={1};
    option.weight =0 ;}

al_e_cvp :coverpoint  F_cvg_tx.almostempty{
    bins almostempty0 ={0};
    bins almostempty1 ={1};
    option.weight =0 ;}

```

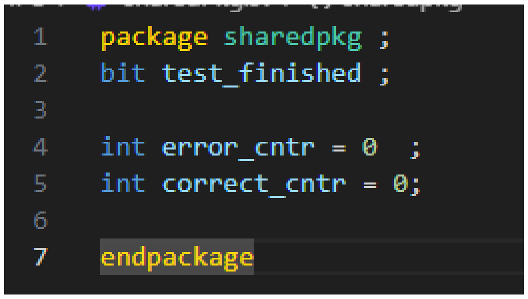


```systemverilog

underf_cvp :coverpoint  F_cvg_tx.underflow{
    bins underflow0 ={0};
    bins underflow1 ={1};
    option.weight =0 ;}

cross wr_enable_cvp,rd_enable_cvp,wr_ack_cvp {
    bins wren_ack  = binsof(wr_enable_cvp.wr_en1) && binsof (wr_ack_cvp.wr_ak1) ;
    bins rden_ack  = binsof(wr_enable_cvp.wr_en1) && binsof (wr_ack_cvp.wr_ak1) && binsof (rd_enable_cvp.rd_en1) ;  //must add wr_en because wr_ack wont be 1 unless wr_en is 1
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,overf_cvp {
    bins wren_of  = binsof (wr_enable_cvp.wr_en1) && binsof (overf_cvp.overflow1) ;
    bins rden_of =  binsof (rd_enable_cvp.rd_en1) && binsof (overf_cvp.overflow1) ;   //should not happen ???
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,underf_cvp {
    bins wren_uf  = binsof (wr_enable_cvp.wr_en1) && binsof (underf_cvp.underflow1) ;
    bins rden_uf =  binsof (rd_enable_cvp.rd_en1) && binsof (underf_cvp.underflow1) ;
    option.cross_auto_bin_max = 0 ;
}


cross wr_enable_cvp,rd_enable_cvp,fl_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (fl_cvp.full1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (fl_cvp.full1) ;
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,mt_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (mt_cvp.empty1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (mt_cvp.empty1) ;
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,al_f_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (al_f_cvp.almostfull1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (al_f_cvp.almostfull1) ;
    option.cross_auto_bin_max = 0 ;
}

```



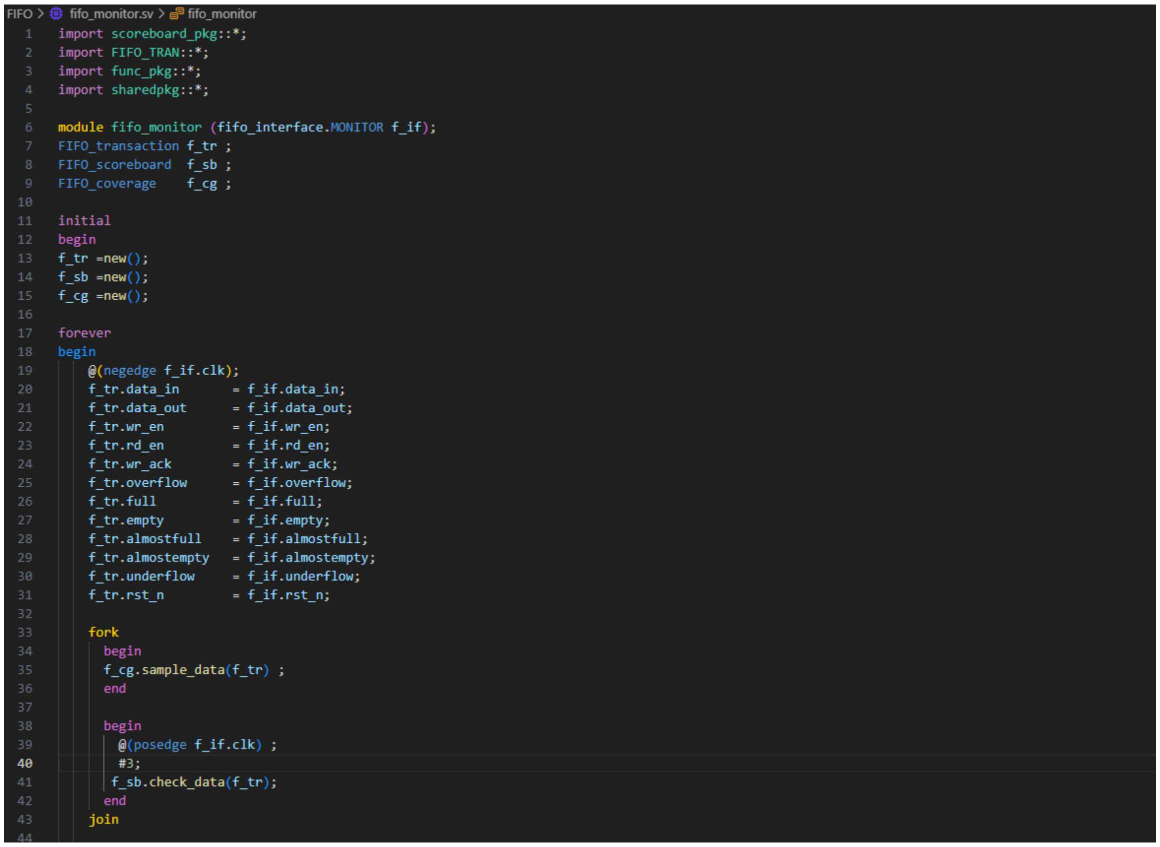
```systemverilog
cross wr_enable_cvp,rd_enable_cvp,fl_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (fl_cvp.full1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (fl_cvp.full1) ;
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,mt_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (mt_cvp.empty1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (mt_cvp.empty1) ;
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,al_f_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (al_f_cvp.almostfull1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (al_f_cvp.almostfull1) ;
    option.cross_auto_bin_max = 0 ;
}

cross wr_enable_cvp,rd_enable_cvp,al_e_cvp {
    bins wren_f  = binsof (wr_enable_cvp.wr_en1) && binsof (al_e_cvp.almostempty1) ;
    bins rden_f =  binsof (rd_enable_cvp.rd_en1) && binsof (al_e_cvp.almostempty1) ;
    option.cross_auto_bin_max = 0 ;
}

endgroup

function new () ;
  cg = new ();
endfunction



function void  sample_data( FIFO_transaction F_txn); //input is the default
F_cvg_tx = F_txn   ;
cg.sample();
endfunction

endclass

endpackage
```

7- shared package



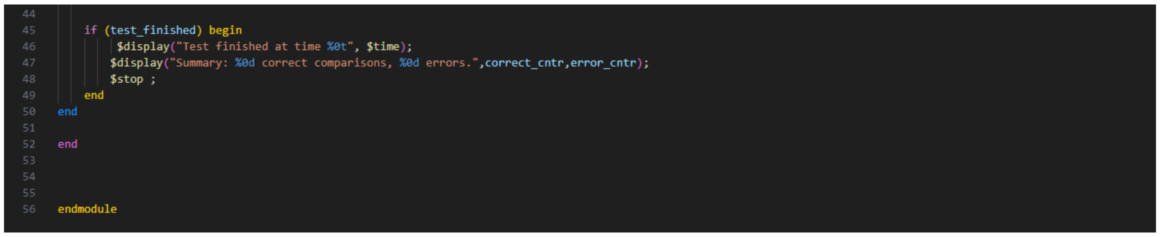
```
package sharedpkg ;
bit test_finished ;

int error_cntr = 0   ;
int correct_cntr = 0;

endpackage
```

8- monitor package



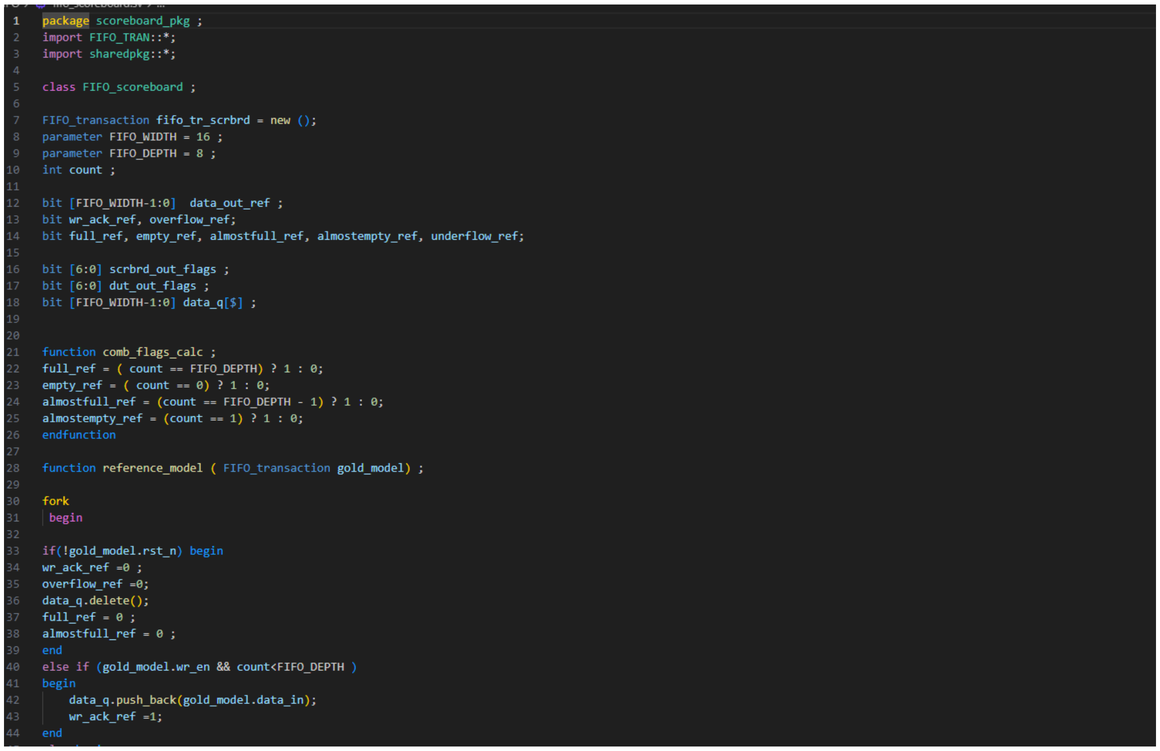
FIFO > fifo_monitor.sv > fifo_monitor

```
import scoreboard_pkg::*;
import FIFO_TRAN::*;
import func_pkg::*;
import sharedpkg::*;

module fifo_monitor (fifo_interface.MONITOR f_if);
FIFO_transaction f_tr ;
FIFO_scoreboard  f_sb ;
FIFO_coverage    f_cg ;

initial
begin
f_tr =new();
f_sb =new();
f_cg =new();

forever
begin
    @(negedge f_if.clk);
    f_tr.data_in        = f_if.data_in;
    f_tr.data_out       = f_if.data_out;
    f_tr.wr_en          = f_if.wr_en;
    f_tr.rd_en          = f_if.rd_en;
    f_tr.wr_ack         = f_if.wr_ack;
    f_tr.overflow       = f_if.overflow;
    f_tr.full           = f_if.full;
    f_tr.empty          = f_if.empty;
    f_tr.almostfull     = f_if.almostfull;
    f_tr.almostempty    = f_if.almostempty;
    f_tr.underflow      = f_if.underflow;
    f_tr.rst_n          = f_if.rst_n;

    fork
      begin
      f_cg.sample_data(f_tr) ;
      end

      begin
        @(posedge f_if.clk) ;
        #3;
       f_sb.check_data(f_tr);
      end
    join
```

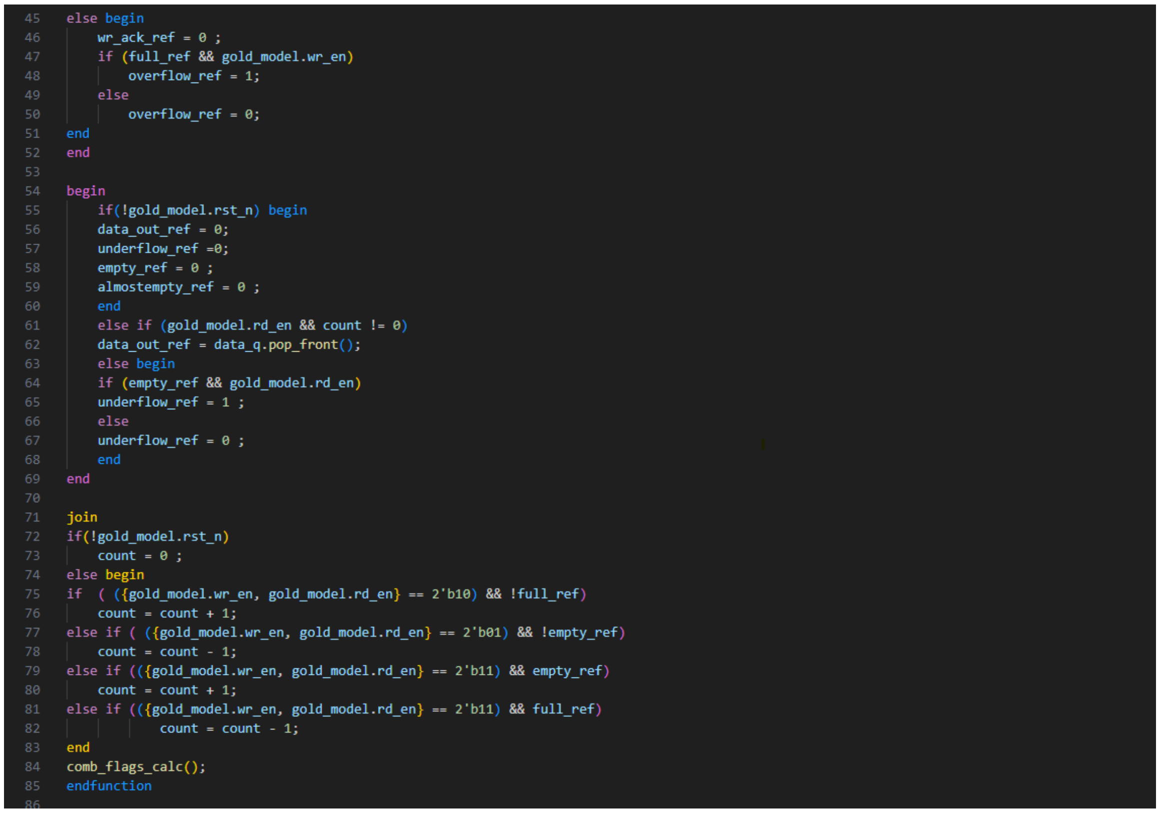


```
    if (test_finished) begin
        $display("Test finished at time %0t", $time);
       $display("Summary: %0d correct comparisons, %0d errors.",correct_cntr,error_cntr);
       $stop ;
    end
end

end



endmodule
```

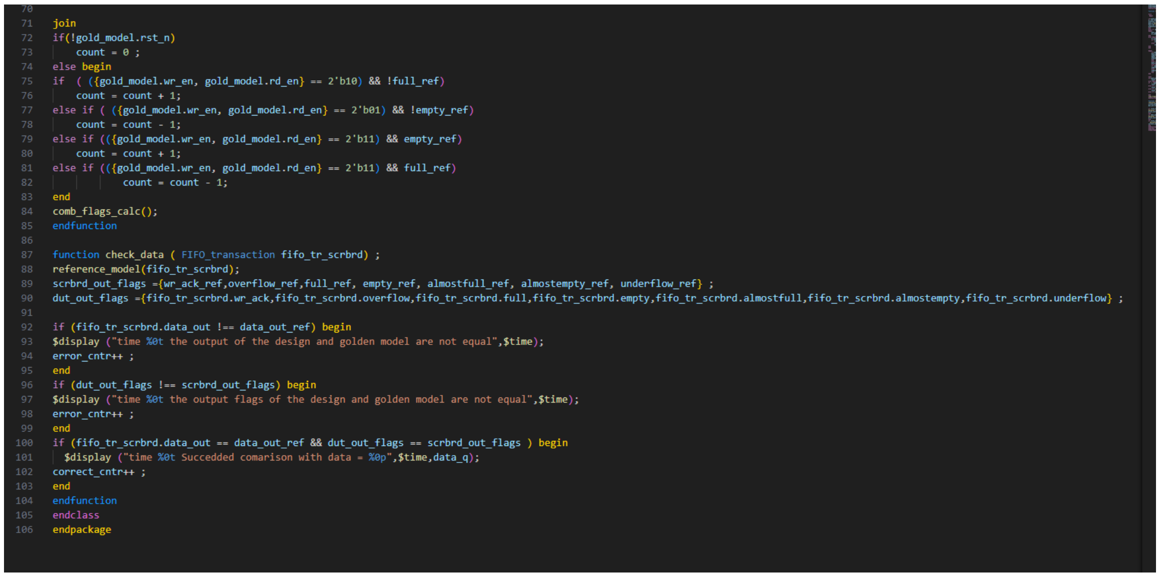
9- scoreboard package



```
package scoreboard_pkg ;
import FIFO_TRAN::*;
import sharedpkg::*;

class FIFO_scoreboard ;

FIFO_transaction fifo_tr_scrbrd = new ();
parameter FIFO_WIDTH = 16 ;
parameter FIFO_DEPTH = 8 ;
int count ;

bit [FIFO_WIDTH-1:0]  data_out_ref ;
bit wr_ack_ref, overflow_ref;
bit full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;

bit [6:0] scrbrd_out_flags ;
bit [6:0] dut_out_flags ;
bit [FIFO_WIDTH-1:0] data_q[$] ;


function comb_flags_calc ;
full_ref = ( count == FIFO_DEPTH) ? 1 : 0;
empty_ref = ( count == 0) ? 1 : 0;
almostfull_ref = (count == FIFO_DEPTH - 1) ? 1 : 0;
almostempty_ref = (count == 1) ? 1 : 0;
endfunction

function reference_model ( FIFO_transaction gold_model) ;

fork
 begin

if(!gold_model.rst_n) begin
wr_ack_ref =0 ;
overflow_ref =0;
data_q.delete();
full_ref = 0 ;
almostfull_ref = 0 ;
end
else if (gold_model.wr_en && count<FIFO_DEPTH )
begin
    data_q.push_back(gold_model.data_in);
    wr_ack_ref =1;
end
```

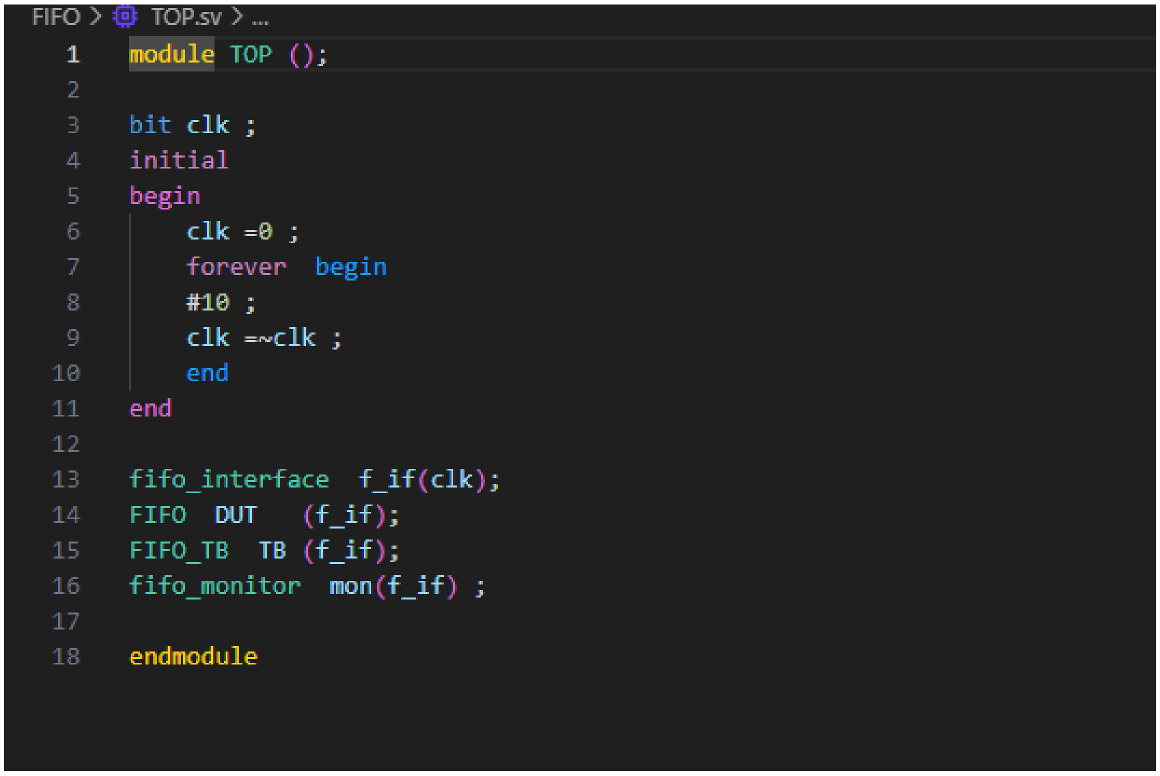


```
else begin
    wr_ack_ref = 0 ;
    if (full_ref && gold_model.wr_en)
        overflow_ref = 1;
    else
        overflow_ref = 0;
end
end

begin
    if(!gold_model.rst_n) begin
    data_out_ref = 0;
    underflow_ref =0;
    empty_ref = 0 ;
    almostempty_ref = 0 ;
    end
    else if (gold_model.rd_en && count != 0)
    data_out_ref = data_q.pop_front();
    else begin
    if (empty_ref && gold_model.rd_en)
    underflow_ref = 1 ;
    else
    underflow_ref = 0 ;
    end
end

join
if(!gold_model.rst_n)
    count = 0 ;
else begin
if  ( ({gold_model.wr_en, gold_model.rd_en} == 2'b10) && !full_ref)
    count = count + 1;
else if ( ({gold_model.wr_en, gold_model.rd_en} == 2'b01) && !empty_ref)
    count = count - 1;
else if (({gold_model.wr_en, gold_model.rd_en} == 2'b11) && empty_ref)
    count = count + 1;
else if (({gold_model.wr_en, gold_model.rd_en} == 2'b11) && full_ref)
            count = count - 1;
end
comb_flags_calc();
endfunction
```



```
join
if(!gold_model.rst_n)
    count = 0 ;
else begin
if  ( ({gold_model.wr_en, gold_model.rd_en} == 2'b10) && !full_ref)
    count = count + 1;
else if ( ({gold_model.wr_en, gold_model.rd_en} == 2'b01) && !empty_ref)
    count = count - 1;
else if (({gold_model.wr_en, gold_model.rd_en} == 2'b11) && empty_ref)
    count = count + 1;
else if (({gold_model.wr_en, gold_model.rd_en} == 2'b11) && full_ref)
            count = count - 1;
end
comb_flags_calc();
endfunction

function check_data ( FIFO_transaction fifo_tr_scrbrd) ;
reference_model(fifo_tr_scrbrd);
scrbrd_out_flags ={wr_ack_ref,overflow_ref,full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref} ;
dut_out_flags ={fifo_tr_scrbrd.wr_ack,fifo_tr_scrbrd.overflow,fifo_tr_scrbrd.full,fifo_tr_scrbrd.empty,fifo_tr_scrbrd.almostfull,fifo_tr_scrbrd.almostempty,fifo_tr_scrbrd.underflow} ;

if (fifo_tr_scrbrd.data_out !== data_out_ref) begin
$display ("time %0t the output of the design and golden model are not equal",$time);
error_cntr++ ;
end
if (dut_out_flags !== scrbrd_out_flags) begin
$display ("time %0t the output flags of the design and golden model are not equal",$time);
error_cntr++ ;
end
if (fifo_tr_scrbrd.data_out == data_out_ref && dut_out_flags == scrbrd_out_flags ) begin
  $display ("time %0t Succedded comarison with data = %0p",$time,data_q);
correct_cntr++ ;
end
endfunction
endclass
endpackage
```

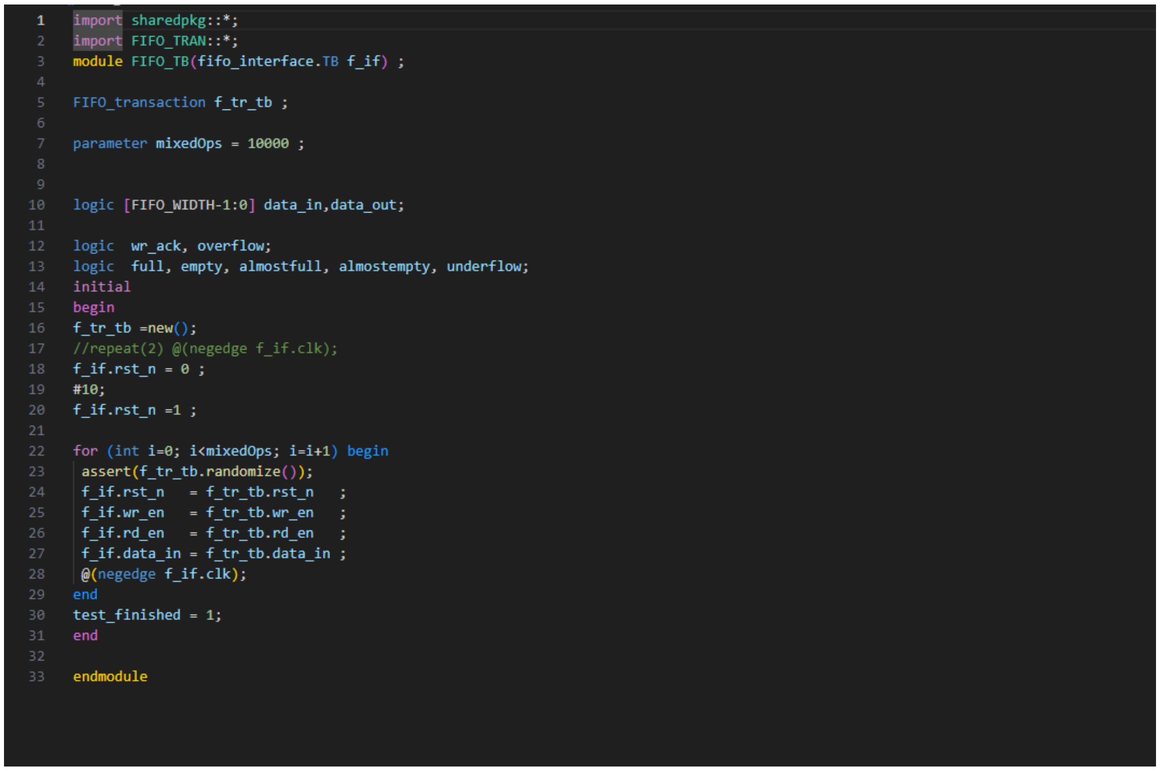
10- Top module



FIFO > TOP.sv > ...

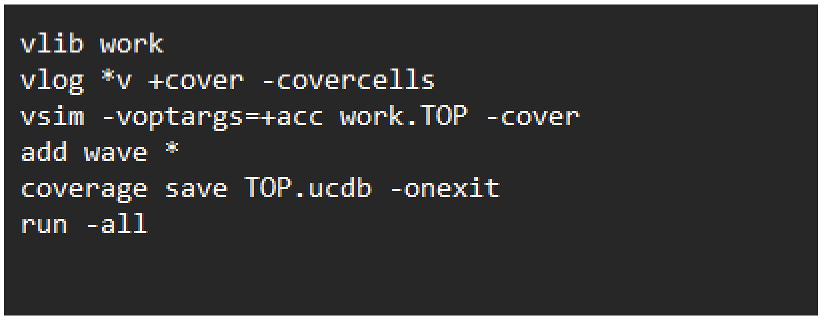
```
module TOP ();

bit clk ;
initial
begin
    clk =0 ;
    forever  begin
    #10 ;
    clk =~clk ;
    end
end

fifo_interface  f_if(clk);
FIFO  DUT   (f_if);
FIFO_TB  TB (f_if);
fifo_monitor  mon(f_if) ;

endmodule
```

11- Testbench module



```
import sharedpkg::*;
import FIFO_TRAN::*;
module FIFO_TB(fifo_interface.TB f_if) ;

FIFO_transaction f_tr_tb ;

parameter mixedOps = 10000 ;


logic [FIFO_WIDTH-1:0] data_in,data_out;

logic  wr_ack, overflow;
logic  full, empty, almostfull, almostempty, underflow;
initial
begin
f_tr_tb =new();
//repeat(2) @(negedge f_if.clk);
f_if.rst_n = 0 ;
#10;
f_if.rst_n =1 ;

for (int i=0; i<mixedOps; i=i+1) begin
 assert(f_tr_tb.randomize());
 f_if.rst_n   = f_tr_tb.rst_n   ;
 f_if.wr_en   = f_tr_tb.wr_en   ;
 f_if.rd_en   = f_tr_tb.rd_en   ;
 f_if.data_in = f_tr_tb.data_in ;
 @(negedge f_if.clk);
end
test_finished = 1;
end

endmodule
```

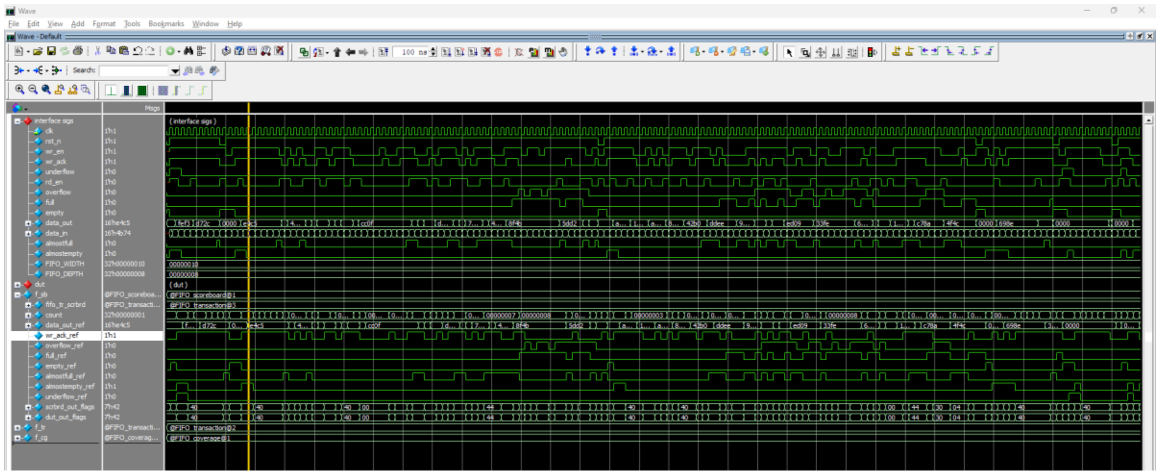
12- do run file (I only used this run file to generate the coverage reports but the simulations generated through the tool)



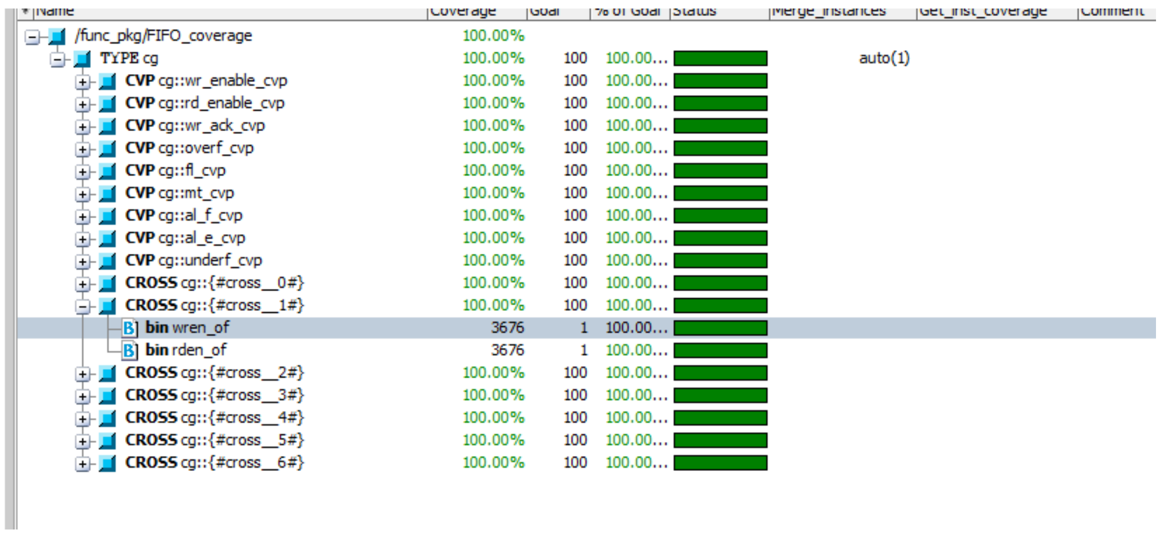
```
vlib work
vlog *v +cover -covercells
vsim -voptargs=+acc work.TOP -cover
add wave *
coverage save TOP.ucdb -onexit
run -all
```

13- snippets from the waveform and transcript and the coverage reports

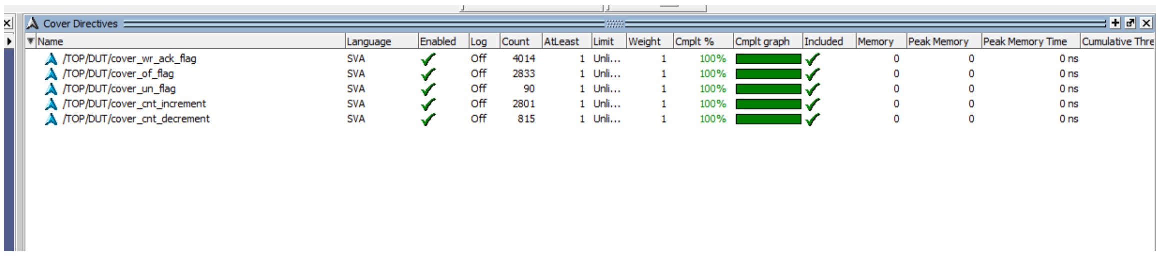
The waveform shows that the ref signals are having the same response of the design signals



```
interface sigs
  clk               1'h1
  rst_n             1'h1
  wr_en             1'h1
  wr_ack            1'h1
  underflow         1'h0
  rd_en             1'h0
  overflow          1'h0
  full              1'h0
  empty             1'h0
  data_out          16'he4c5
  data_in           16'h4b74
  almostfull        1'h0
  almostempty       1'h0
  FIFO_WIDTH        32'h00000010
  FIFO_DEPTH        32'h00000008
dut
f_sb                @FIFO_scoreboa...
  fifo_tr_scrbrd    @FIFO_transacti...
  count             32'h00000001
  data_out_ref      16'he4c5
  wr_ack_ref        1'h1
  overflow_ref      1'h0
  full_ref          1'h0
  empty_ref         1'h0
  almostfull_ref    1'h0
  almostempty_ref   1'h1
  underflow_ref     1'h0
  scrbrd_out_flags  7'h42
  dut_out_flags     7'h42
f_tr                @FIFO_transacti...
f_cg                @FIFO_coverag...
```

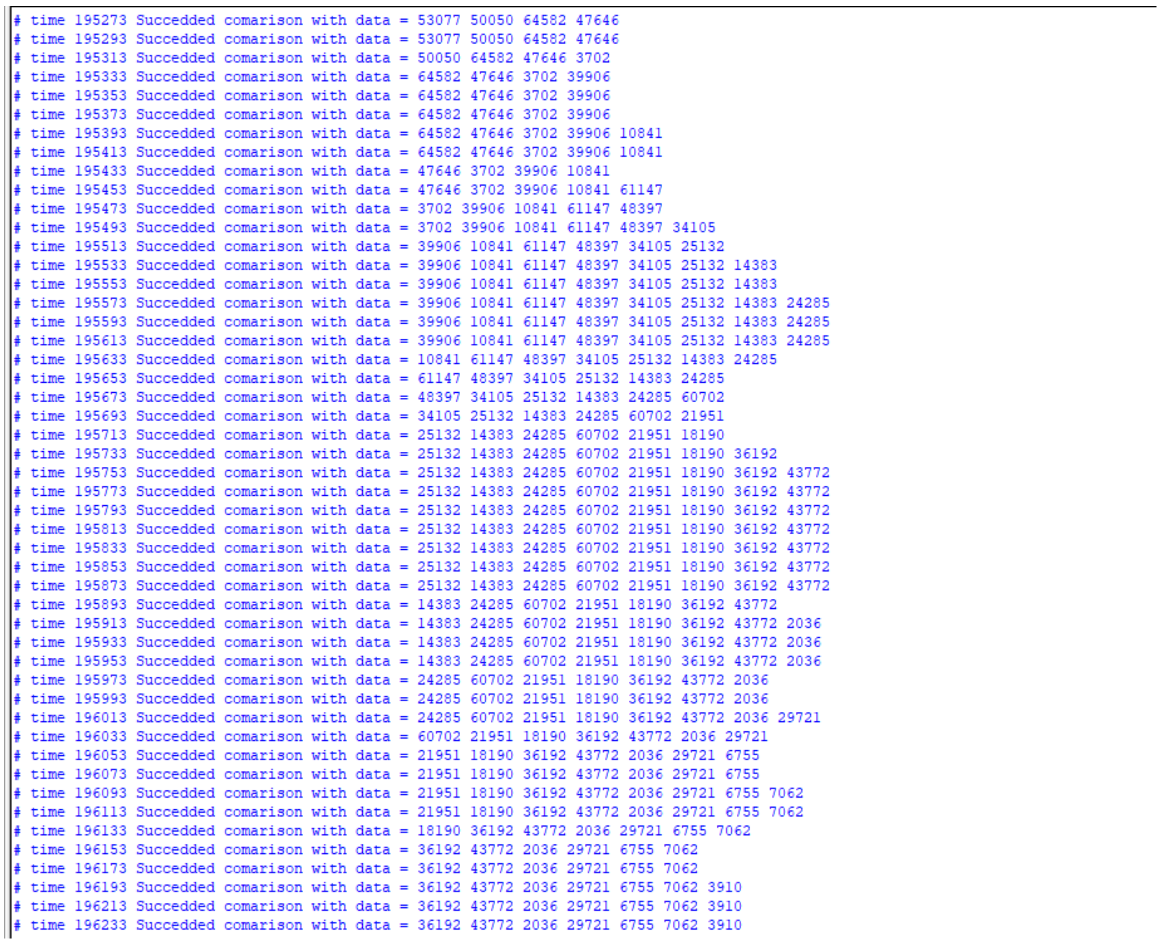


Name	Coverage	Goal	% of Goal	Status	Merge_instances	Get_inst_coverage	Comment
/func_pkg/FIFO_coverage	100.00%						
TYPE cg	100.00%	100	100.00...	🟩		auto(1)	
CVP cg::wr_enable_cvp	100.00%	100	100.00...	🟩			
CVP cg::rd_enable_cvp	100.00%	100	100.00...	🟩			
CVP cg::wr_ack_cvp	100.00%	100	100.00...	🟩			
CVP cg::overf_cvp	100.00%	100	100.00...	🟩			
CVP cg::fl_cvp	100.00%	100	100.00...	🟩			
CVP cg::mt_cvp	100.00%	100	100.00...	🟩			
CVP cg::al_f_cvp	100.00%	100	100.00...	🟩			
CVP cg::al_e_cvp	100.00%	100	100.00...	🟩			
CVP cg::underf_cvp	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__0#}	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__1#}	100.00%	100	100.00...	🟩			
bin wren_of	3676	1	100.00...	🟩			
bin rden_of	3676	1	100.00...	🟩			
CROSS cg::{#cross__2#}	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__3#}	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__4#}	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__5#}	100.00%	100	100.00...	🟩			
CROSS cg::{#cross__6#}	100.00%	100	100.00...	🟩			

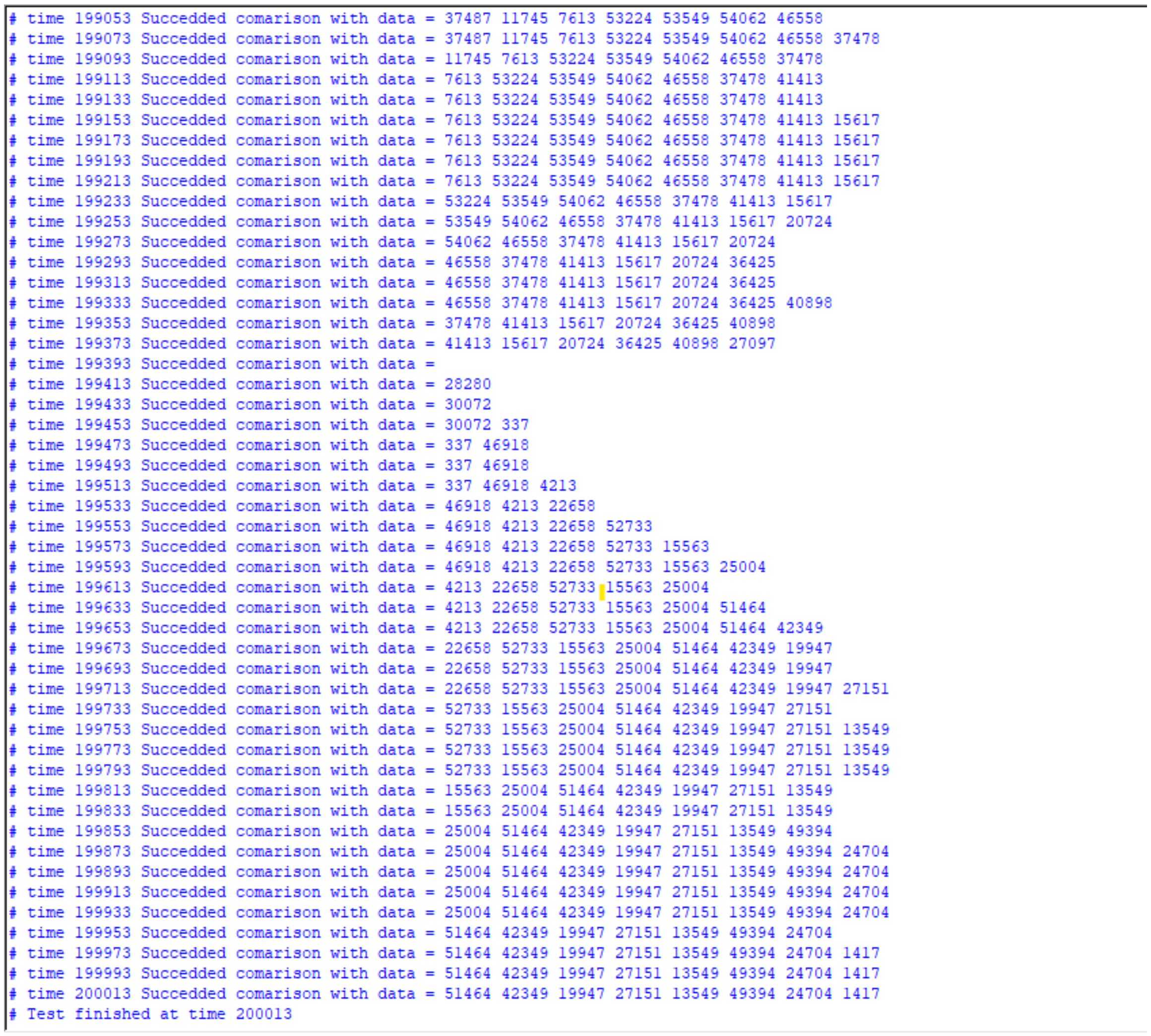


Cover Directives

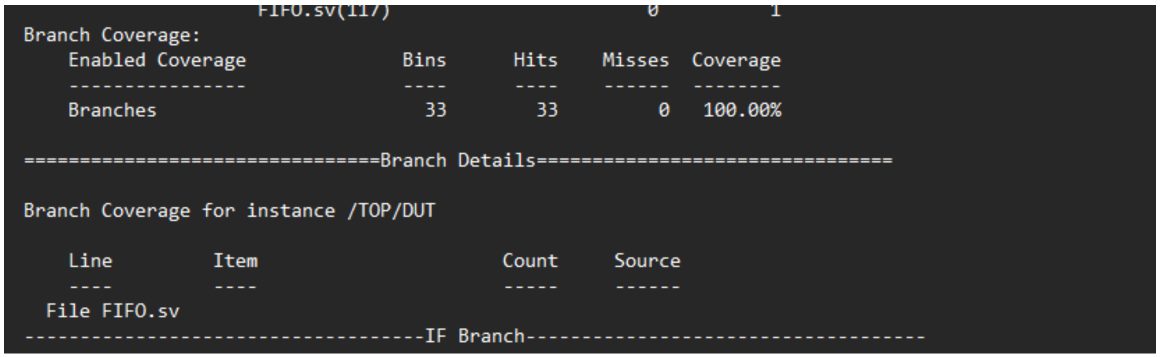
Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Thre
/TOP/DUT/cover_wr_ack_flag	SVA	✓	Off	4014	1	Unli...	1	100%	🟩	✓	0	0	0 ns	
/TOP/DUT/cover_of_flag	SVA	✓	Off	2833	1	Unli...	1	100%	🟩	✓	0	0	0 ns	
/TOP/DUT/cover_un_flag	SVA	✓	Off	90	1	Unli...	1	100%	🟩	✓	0	0	0 ns	
/TOP/DUT/cover_cnt_increment	SVA	✓	Off	2801	1	Unli...	1	100%	🟩	✓	0	0	0 ns	
/TOP/DUT/cover_cnt_decrement	SVA	✓	Off	815	1	Unli...	1	100%	🟩	✓	0	0	0 ns	



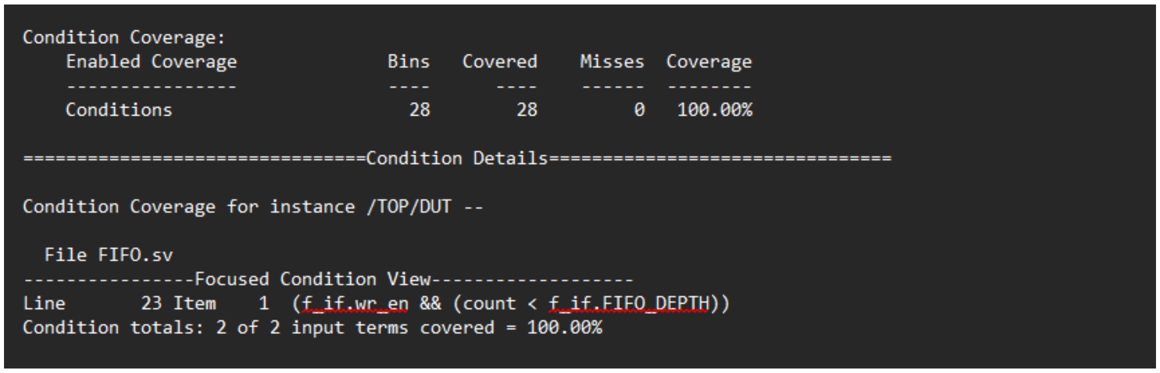
```
# time 195273 Succedded comarison with data = 53077 50050 64582 47646
# time 195293 Succedded comarison with data = 53077 50050 64582 47646
# time 195313 Succedded comarison with data = 50050 64582 47646 3702
# time 195333 Succedded comarison with data = 64582 47646 3702 39906
# time 195353 Succedded comarison with data = 64582 47646 3702 39906
# time 195373 Succedded comarison with data = 64582 47646 3702 39906
# time 195393 Succedded comarison with data = 64582 47646 3702 39906 10841
# time 195413 Succedded comarison with data = 64582 47646 3702 39906 10841
# time 195433 Succedded comarison with data = 47646 3702 39906 10841
# time 195453 Succedded comarison with data = 47646 3702 39906 10841 61147
# time 195473 Succedded comarison with data = 3702 39906 10841 61147 48397
# time 195493 Succedded comarison with data = 3702 39906 10841 61147 48397 34105
# time 195513 Succedded comarison with data = 39906 10841 61147 48397 34105 25132
# time 195533 Succedded comarison with data = 39906 10841 61147 48397 34105 25132 14383
# time 195553 Succedded comarison with data = 39906 10841 61147 48397 34105 25132 14383
# time 195573 Succedded comarison with data = 39906 10841 61147 48397 34105 25132 14383 24285
# time 195593 Succedded comarison with data = 39906 10841 61147 48397 34105 25132 14383 24285
# time 195613 Succedded comarison with data = 39906 10841 61147 48397 34105 25132 14383 24285
# time 195633 Succedded comarison with data = 10841 61147 48397 34105 25132 14383 24285
# time 195653 Succedded comarison with data = 61147 48397 34105 25132 14383 24285
# time 195673 Succedded comarison with data = 48397 34105 25132 14383 24285 60702
# time 195693 Succedded comarison with data = 34105 25132 14383 24285 60702 21951
# time 195713 Succedded comarison with data = 25132 14383 24285 60702 21951 18190
# time 195733 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192
# time 195753 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195773 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195793 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195813 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195833 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195853 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195873 Succedded comarison with data = 25132 14383 24285 60702 21951 18190 36192 43772
# time 195893 Succedded comarison with data = 14383 24285 60702 21951 18190 36192 43772
# time 195913 Succedded comarison with data = 14383 24285 60702 21951 18190 36192 43772 2036
# time 195933 Succedded comarison with data = 14383 24285 60702 21951 18190 36192 43772 2036
# time 195953 Succedded comarison with data = 14383 24285 60702 21951 18190 36192 43772 2036
# time 195973 Succedded comarison with data = 24285 60702 21951 18190 36192 43772 2036
# time 195993 Succedded comarison with data = 24285 60702 21951 18190 36192 43772 2036
# time 196013 Succedded comarison with data = 24285 60702 21951 18190 36192 43772 2036 29721
# time 196033 Succedded comarison with data = 60702 21951 18190 36192 43772 2036 29721
# time 196053 Succedded comarison with data = 21951 18190 36192 43772 2036 29721 6755
# time 196073 Succedded comarison with data = 21951 18190 36192 43772 2036 29721 6755
# time 196093 Succedded comarison with data = 21951 18190 36192 43772 2036 29721 6755 7062
# time 196113 Succedded comarison with data = 21951 18190 36192 43772 2036 29721 6755 7062
# time 196133 Succedded comarison with data = 18190 36192 43772 2036 29721 6755 7062
# time 196153 Succedded comarison with data = 36192 43772 2036 29721 6755 7062
# time 196173 Succedded comarison with data = 36192 43772 2036 29721 6755 7062
# time 196193 Succedded comarison with data = 36192 43772 2036 29721 6755 7062 3910
# time 196213 Succedded comarison with data = 36192 43772 2036 29721 6755 7062 3910
# time 196233 Succedded comarison with data = 36192 43772 2036 29721 6755 7062 3910
```



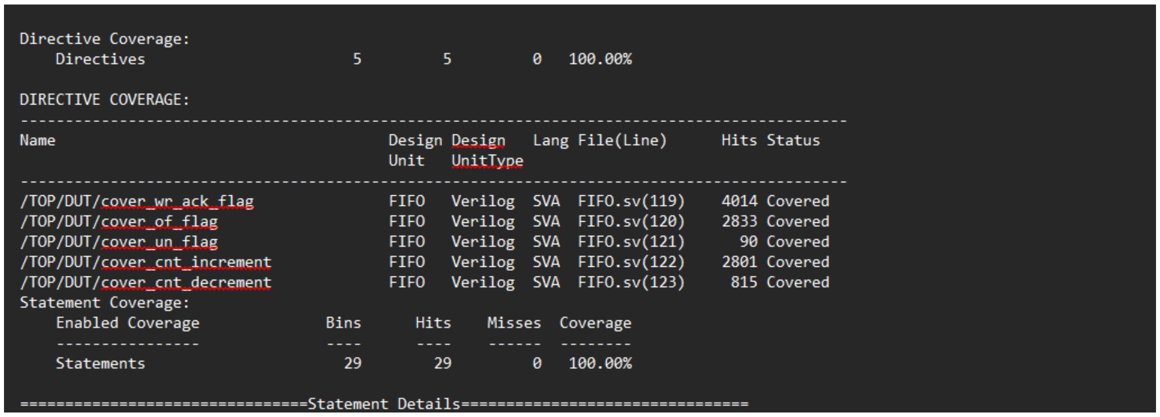
```
# time 199053 Succedded comarison with data = 37487 11745 7613 53224 53549 54062 46558
# time 199073 Succedded comarison with data = 37487 11745 7613 53224 53549 54062 46558 37478
# time 199093 Succedded comarison with data = 11745 7613 53224 53549 54062 46558 37478
# time 199113 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413
# time 199133 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413
# time 199153 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413 15617
# time 199173 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413 15617
# time 199193 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413 15617
# time 199213 Succedded comarison with data = 7613 53224 53549 54062 46558 37478 41413 15617
# time 199233 Succedded comarison with data = 53224 53549 54062 46558 37478 41413 15617
# time 199253 Succedded comarison with data = 53549 54062 46558 37478 41413 15617 20724
# time 199273 Succedded comarison with data = 54062 46558 37478 41413 15617 20724
# time 199293 Succedded comarison with data = 46558 37478 41413 15617 20724 36425
# time 199313 Succedded comarison with data = 46558 37478 41413 15617 20724 36425
# time 199333 Succedded comarison with data = 46558 37478 41413 15617 20724 36425 40898
# time 199353 Succedded comarison with data = 37478 41413 15617 20724 36425 40898
# time 199373 Succedded comarison with data = 41413 15617 20724 36425 40898 27097
# time 199393 Succedded comarison with data =
# time 199413 Succedded comarison with data = 28280
# time 199433 Succedded comarison with data = 30072
# time 199453 Succedded comarison with data = 30072 337
# time 199473 Succedded comarison with data = 337 46918
# time 199493 Succedded comarison with data = 337 46918
# time 199513 Succedded comarison with data = 337 46918 4213
# time 199533 Succedded comarison with data = 46918 4213 22658
# time 199553 Succedded comarison with data = 46918 4213 22658 52733
# time 199573 Succedded comarison with data = 46918 4213 22658 52733 15563
# time 199593 Succedded comarison with data = 46918 4213 22658 52733 15563 25004
# time 199613 Succedded comarison with data = 4213 22658 52733 15563 25004
# time 199633 Succedded comarison with data = 4213 22658 52733 15563 25004 51464
# time 199653 Succedded comarison with data = 4213 22658 52733 15563 25004 51464 42349
# time 199673 Succedded comarison with data = 22658 52733 15563 25004 51464 42349 19947
# time 199693 Succedded comarison with data = 22658 52733 15563 25004 51464 42349 19947
# time 199713 Succedded comarison with data = 22658 52733 15563 25004 51464 42349 19947 27151
# time 199733 Succedded comarison with data = 52733 15563 25004 51464 42349 19947 27151
# time 199753 Succedded comarison with data = 52733 15563 25004 51464 42349 19947 27151 13549
# time 199773 Succedded comarison with data = 52733 15563 25004 51464 42349 19947 27151 13549
# time 199793 Succedded comarison with data = 52733 15563 25004 51464 42349 19947 27151 13549
# time 199813 Succedded comarison with data = 15563 25004 51464 42349 19947 27151 13549
# time 199833 Succedded comarison with data = 15563 25004 51464 42349 19947 27151 13549
# time 199853 Succedded comarison with data = 25004 51464 42349 19947 27151 13549 49394
# time 199873 Succedded comarison with data = 25004 51464 42349 19947 27151 13549 49394 24704
# time 199893 Succedded comarison with data = 25004 51464 42349 19947 27151 13549 49394 24704
# time 199913 Succedded comarison with data = 25004 51464 42349 19947 27151 13549 49394 24704
# time 199933 Succedded comarison with data = 25004 51464 42349 19947 27151 13549 49394 24704
# time 199953 Succedded comarison with data = 51464 42349 19947 27151 13549 49394 24704
# time 199973 Succedded comarison with data = 51464 42349 19947 27151 13549 49394 24704 1417
# time 199993 Succedded comarison with data = 51464 42349 19947 27151 13549 49394 24704 1417
# time 200013 Succedded comarison with data = 51464 42349 19947 27151 13549 49394 24704 1417
# Test finished at time 200013
```



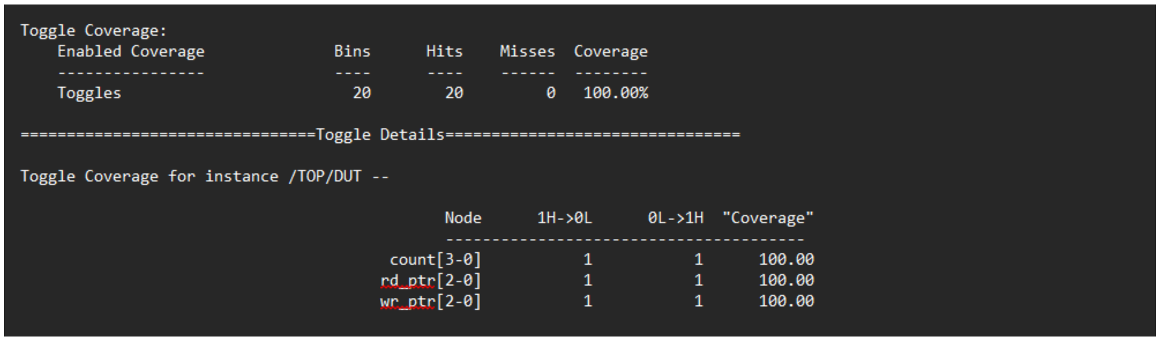
```
                  FIFO.sv(117)                          0          1
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        33        33         0   100.00%

===============================Branch Details===============================

Branch Coverage for instance /TOP/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
-----------------------------------IF Branch-----------------------------------
```



```
Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Conditions                      28         28         0   100.00%

================================Condition Details================================

Condition Coverage for instance /TOP/DUT --

  File FIFO.sv
----------------Focused Condition View-------------------
Line       23 Item    1  (f_if.wr_en && (count < f_if.FIFO_DEPTH))
Condition totals: 2 of 2 input terms covered = 100.00%
```



```
Directive Coverage:
    Directives                      5         5         0   100.00%

DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
-------------------------------------------------------------------------------------------
/TOP/DUT/cover_wr_ack_flag               FIFO   Verilog  SVA  FIFO.sv(119)    4014 Covered
/TOP/DUT/cover_of_flag                   FIFO   Verilog  SVA  FIFO.sv(120)    2833 Covered
/TOP/DUT/cover_un_flag                   FIFO   Verilog  SVA  FIFO.sv(121)      90 Covered
/TOP/DUT/cover_cnt_increment             FIFO   Verilog  SVA  FIFO.sv(122)    2801 Covered
/TOP/DUT/cover_cnt_decrement             FIFO   Verilog  SVA  FIFO.sv(123)     815 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      29        29         0   100.00%

================================Statement Details================================
```



```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         20        20         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /TOP/DUT --

                                              Node      1H->0L      0L->1H  "Coverage"
                                              ---------------------------------------
                                        count[3-0]           1           1      100.00
                                       rd_ptr[2-0]           1           1      100.00
                                       wr_ptr[2-0]           1           1      100.00
```