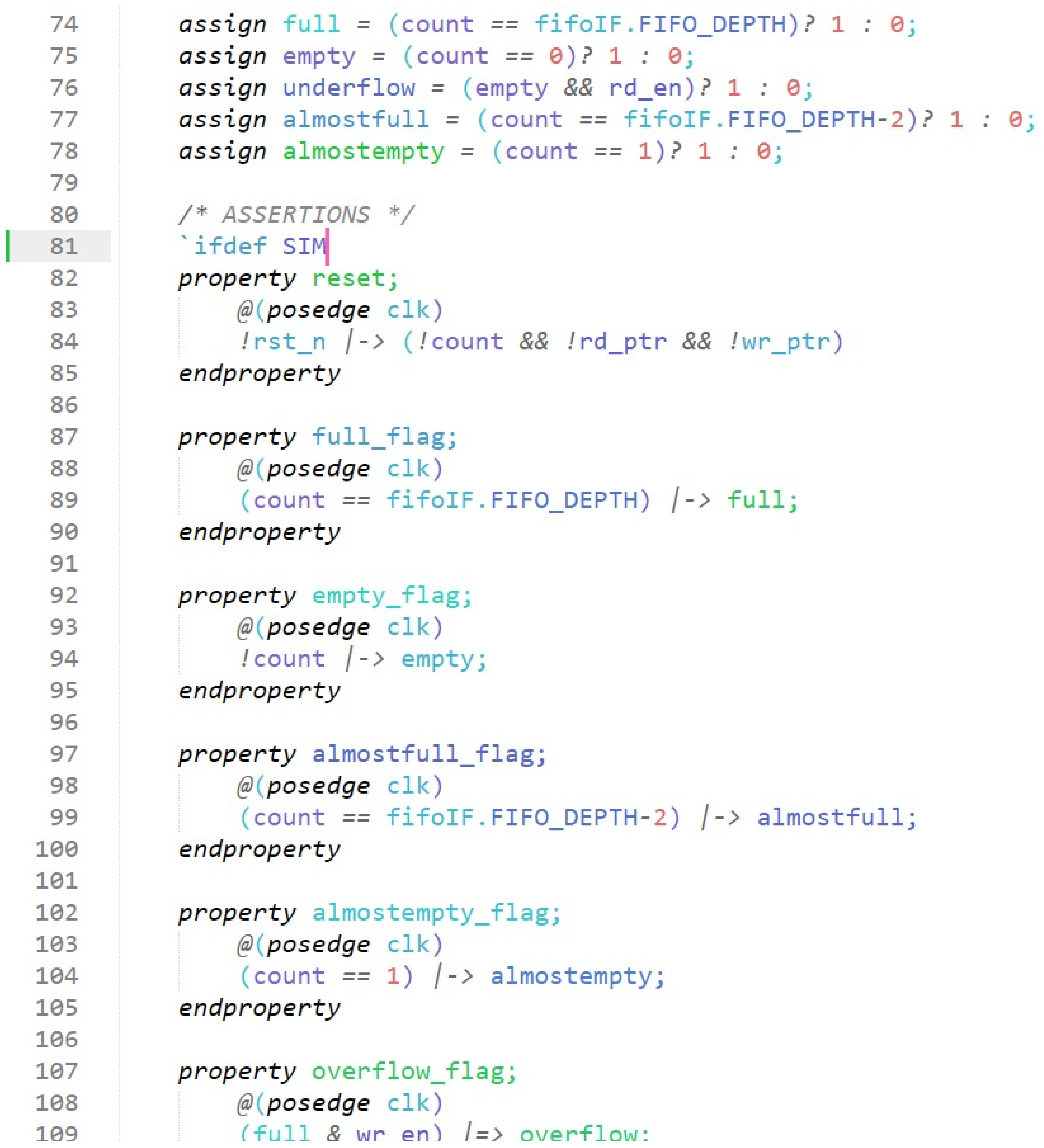
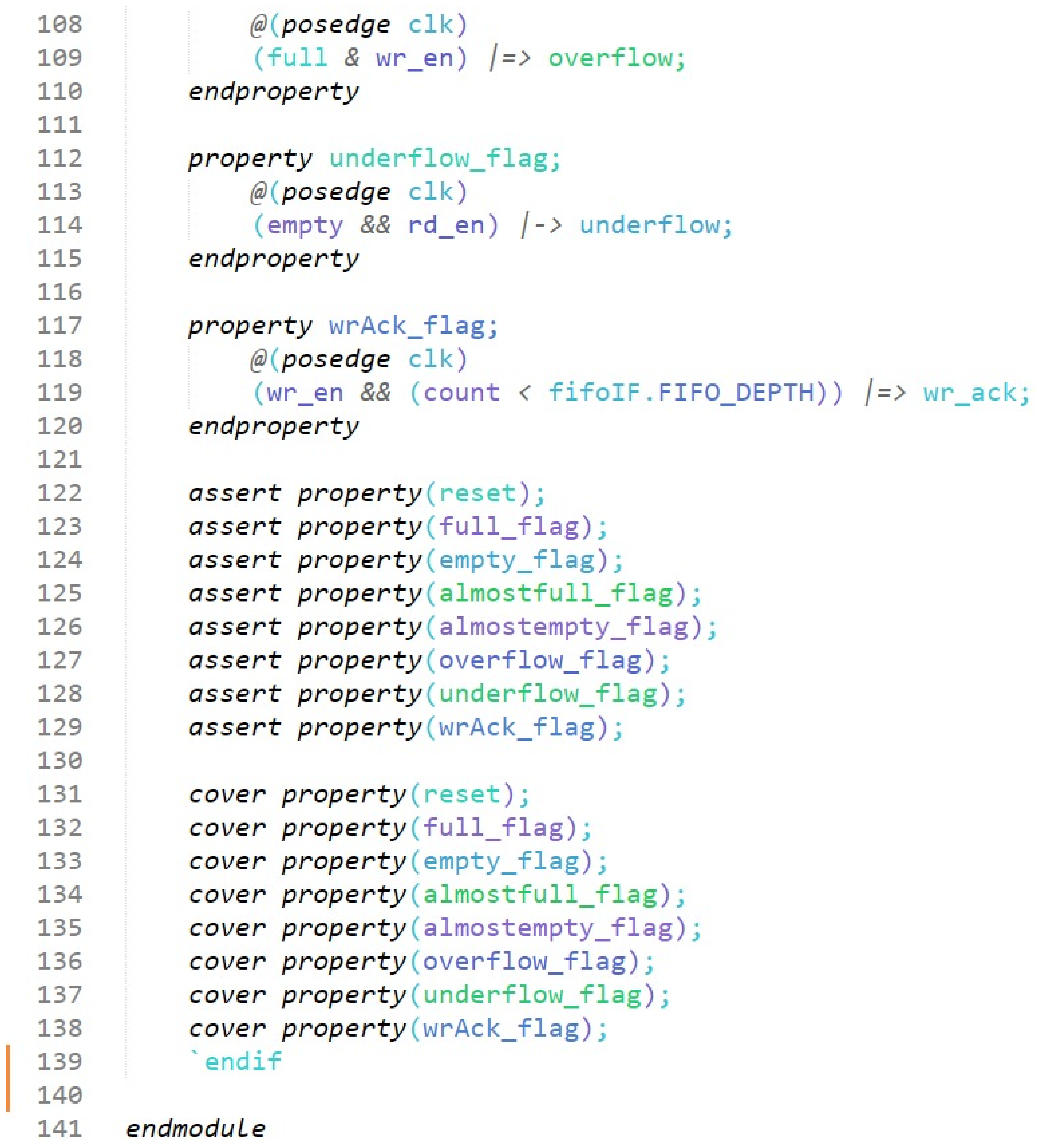
```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(FIFO_if.DUT fifoIF);

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

    assign clk                  = fifoIF.clk;
    assign rst_n                = fifoIF.rst_n;
    assign wr_en                = fifoIF.wr_en;
    assign rd_en                = fifoIF.rd_en;
    assign data_in              = fifoIF.data_in;
    assign fifoIF.full          = full;
    assign fifoIF.empty         = empty;
    assign fifoIF.almostfull    = almostfull;
    assign fifoIF.almostempty   = almostempty;
    assign fifoIF.wr_ack        = wr_ack;
    assign fifoIF.overflow      = overflow;
    assign fifoIF.underflow     = underflow;
    assign fifoIF.data_out      = data_out;

    localparam max_fifo_addr = $clog2(fifoIF.FIFO_DEPTH);

    reg [fifoIF.FIFO_WIDTH-1:0] mem [fifoIF.FIFO_DEPTH-1:0];

    reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    reg [max_fifo_addr:0] count;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            wr_ptr <= 0;
```

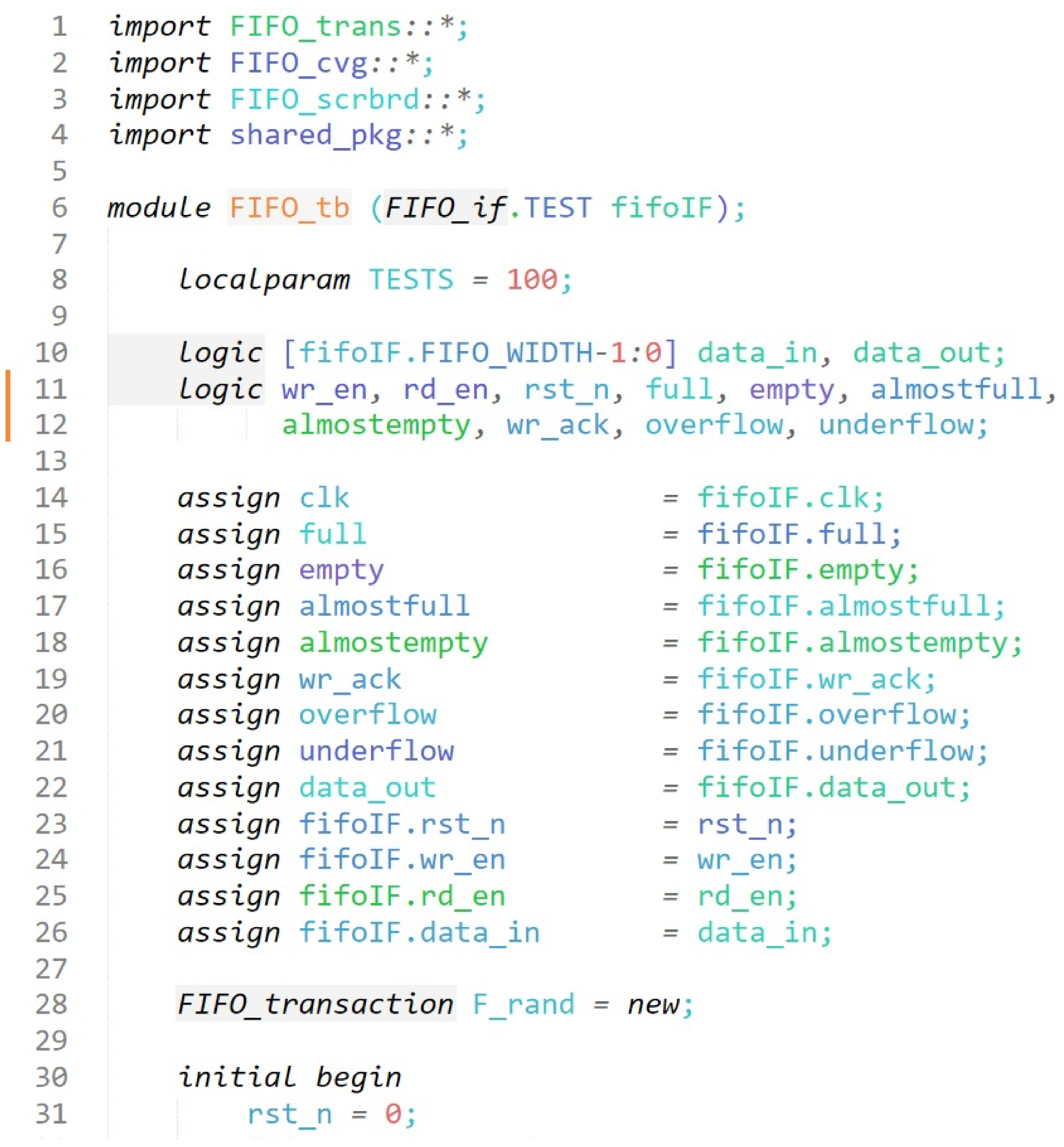
```verilog
        end
        else if (wr_en && (count < fifoIF.FIFO_DEPTH)) begin
            mem[wr_ptr] <= data_in;
            wr_ack <= 1;
            wr_ptr <= wr_ptr + 1;
        end
        else begin
            wr_ack <= 0;
            if (full & wr_en)
                overflow <= 1;
            else
                overflow <= 0;
        end
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            rd_ptr <= 0;
        end
        else if (rd_en && count != 0) begin
            data_out <= mem[rd_ptr];
            rd_ptr <= rd_ptr + 1;
        end
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            count <= 0;
        end
        else begin
            if  ( ({wr_en, rd_en} == 2'b10) && !full)
                count <= count + 1;
            else if ( ({wr_en, rd_en} == 2'b01) && !empty)
                count <= count - 1;
        end
    end
```



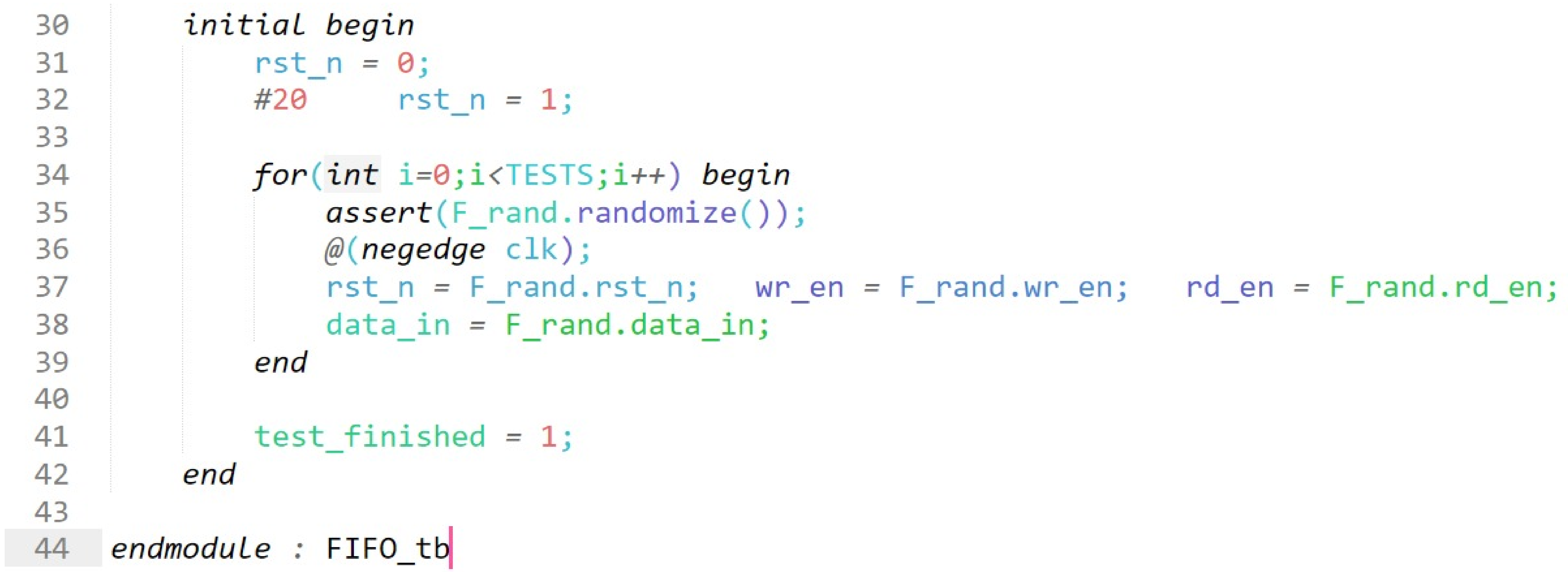
```
    assign full = (count == fifoIF.FIFO_DEPTH)? 1 : 0;
    assign empty = (count == 0)? 1 : 0;
    assign underflow = (empty && rd_en)? 1 : 0;
    assign almostfull = (count == fifoIF.FIFO_DEPTH-2)? 1 : 0;
    assign almostempty = (count == 1)? 1 : 0;

    /* ASSERTIONS */
    `ifdef SIM
    property reset;
        @(posedge clk)
        !rst_n |-> (!count && !rd_ptr && !wr_ptr)
    endproperty

    property full_flag;
        @(posedge clk)
        (count == fifoIF.FIFO_DEPTH) |-> full;
    endproperty

    property empty_flag;
        @(posedge clk)
        !count |-> empty;
    endproperty

    property almostfull_flag;
        @(posedge clk)
        (count == fifoIF.FIFO_DEPTH-2) |-> almostfull;
    endproperty

    property almostempty_flag;
        @(posedge clk)
        (count == 1) |-> almostempty;
    endproperty

    property overflow_flag;
        @(posedge clk)
        (full & wr_en) |=> overflow;
```



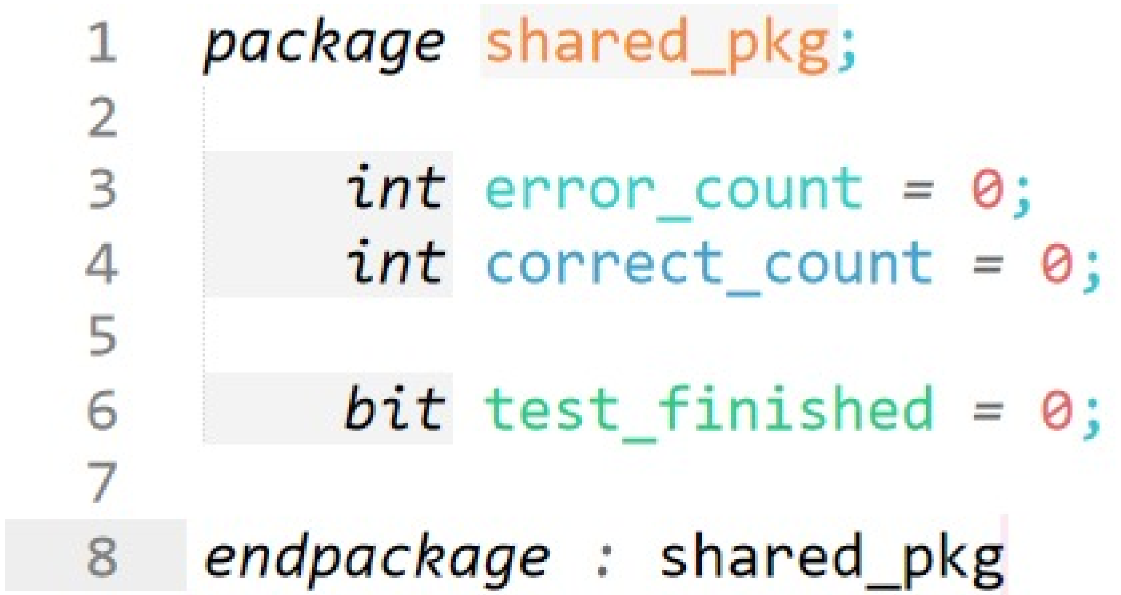
```
        @(posedge clk)
        (full & wr_en) |=> overflow;
    endproperty

    property underflow_flag;
        @(posedge clk)
        (empty && rd_en) |-> underflow;
    endproperty

    property wrAck_flag;
        @(posedge clk)
        (wr_en && (count < fifoIF.FIFO_DEPTH)) |=> wr_ack;
    endproperty

    assert property(reset);
    assert property(full_flag);
    assert property(empty_flag);
    assert property(almostfull_flag);
    assert property(almostempty_flag);
    assert property(overflow_flag);
    assert property(underflow_flag);
    assert property(wrAck_flag);

    cover property(reset);
    cover property(full_flag);
    cover property(empty_flag);
    cover property(almostfull_flag);
    cover property(almostempty_flag);
    cover property(overflow_flag);
    cover property(underflow_flag);
    cover property(wrAck_flag);
    `endif

endmodule
```



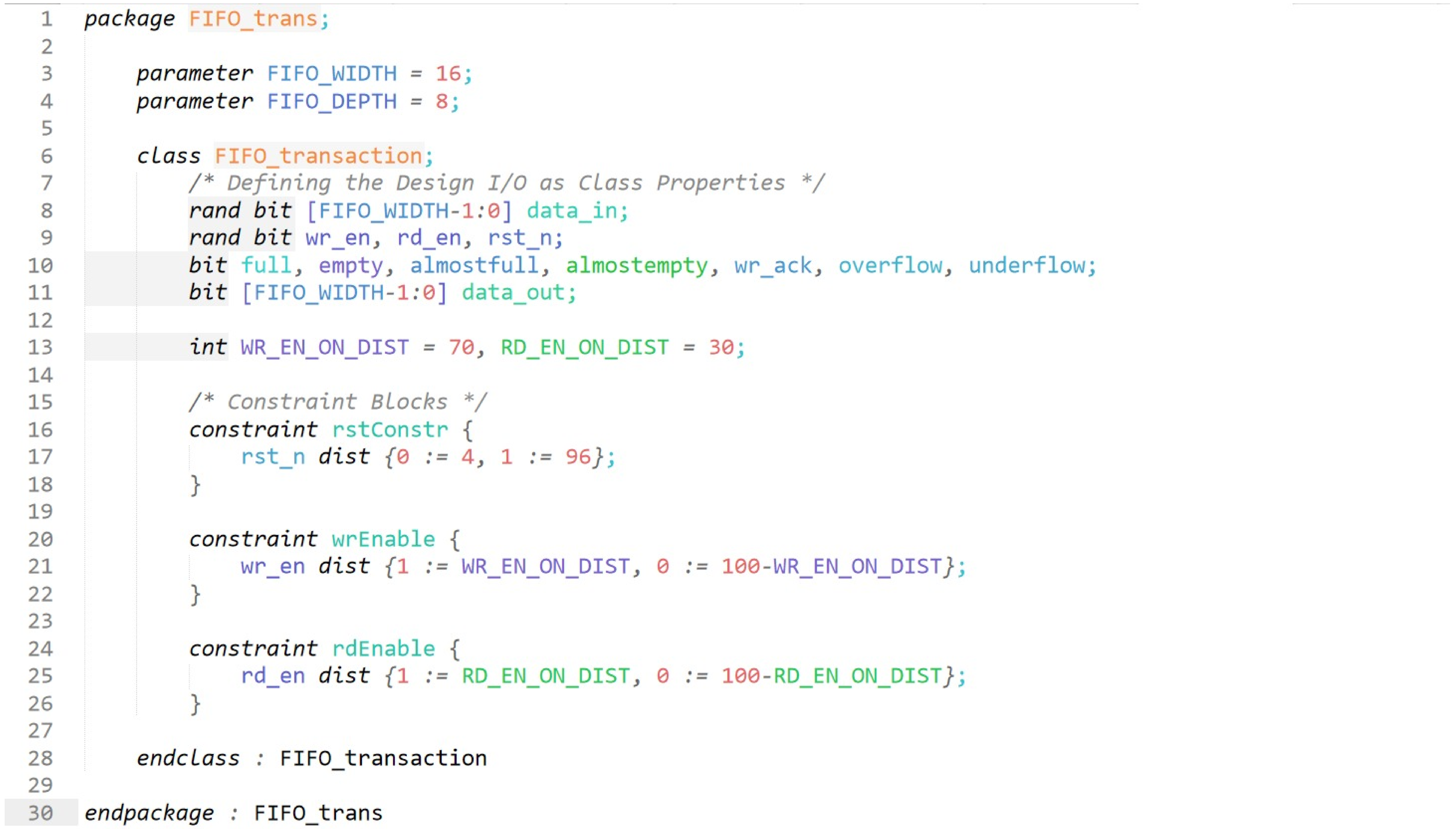
```systemverilog
import FIFO_trans::*;
import FIFO_cvg::*;
import FIFO_scrbrd::*;
import shared_pkg::*;

module FIFO_tb (FIFO_if.TEST fifoIF);

    localparam TESTS = 100;

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

    assign clk                  = fifoIF.clk;
    assign full                 = fifoIF.full;
    assign empty                = fifoIF.empty;
    assign almostfull           = fifoIF.almostfull;
    assign almostempty          = fifoIF.almostempty;
    assign wr_ack               = fifoIF.wr_ack;
    assign overflow             = fifoIF.overflow;
    assign underflow            = fifoIF.underflow;
    assign data_out             = fifoIF.data_out;
    assign fifoIF.rst_n         = rst_n;
    assign fifoIF.wr_en         = wr_en;
    assign fifoIF.rd_en         = rd_en;
    assign fifoIF.data_in       = data_in;

    FIFO_transaction F_rand = new;

    initial begin
        rst_n = 0;
```



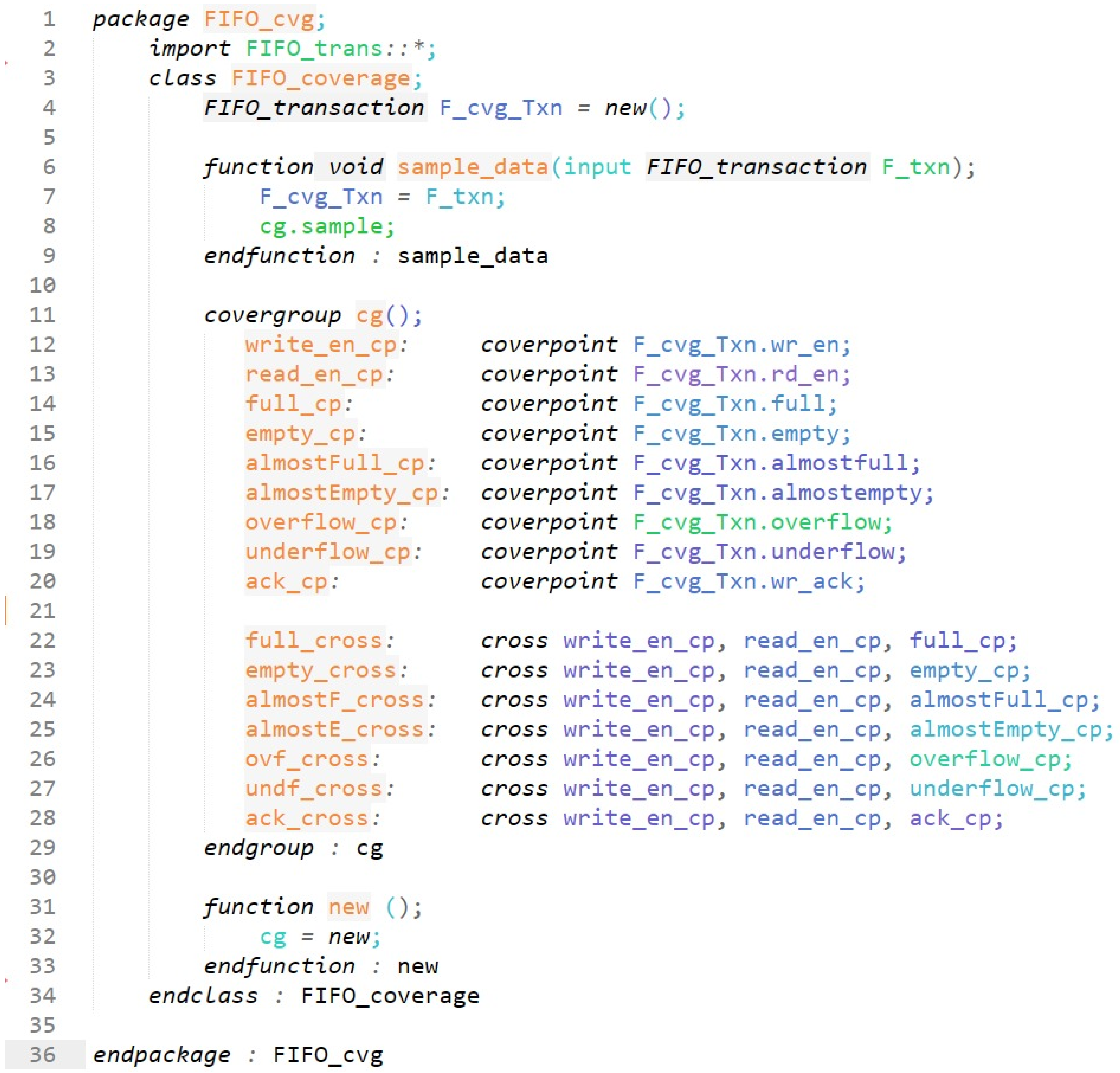
```
    initial begin
        rst_n = 0;
        #20     rst_n = 1;

        for(int i=0;i<TESTS;i++) begin
            assert(F_rand.randomize());
            @(negedge clk);
            rst_n = F_rand.rst_n;   wr_en = F_rand.wr_en;   rd_en = F_rand.rd_en;
            data_in = F_rand.data_in;
        end

        test_finished = 1;
    end

endmodule : FIFO_tb
```



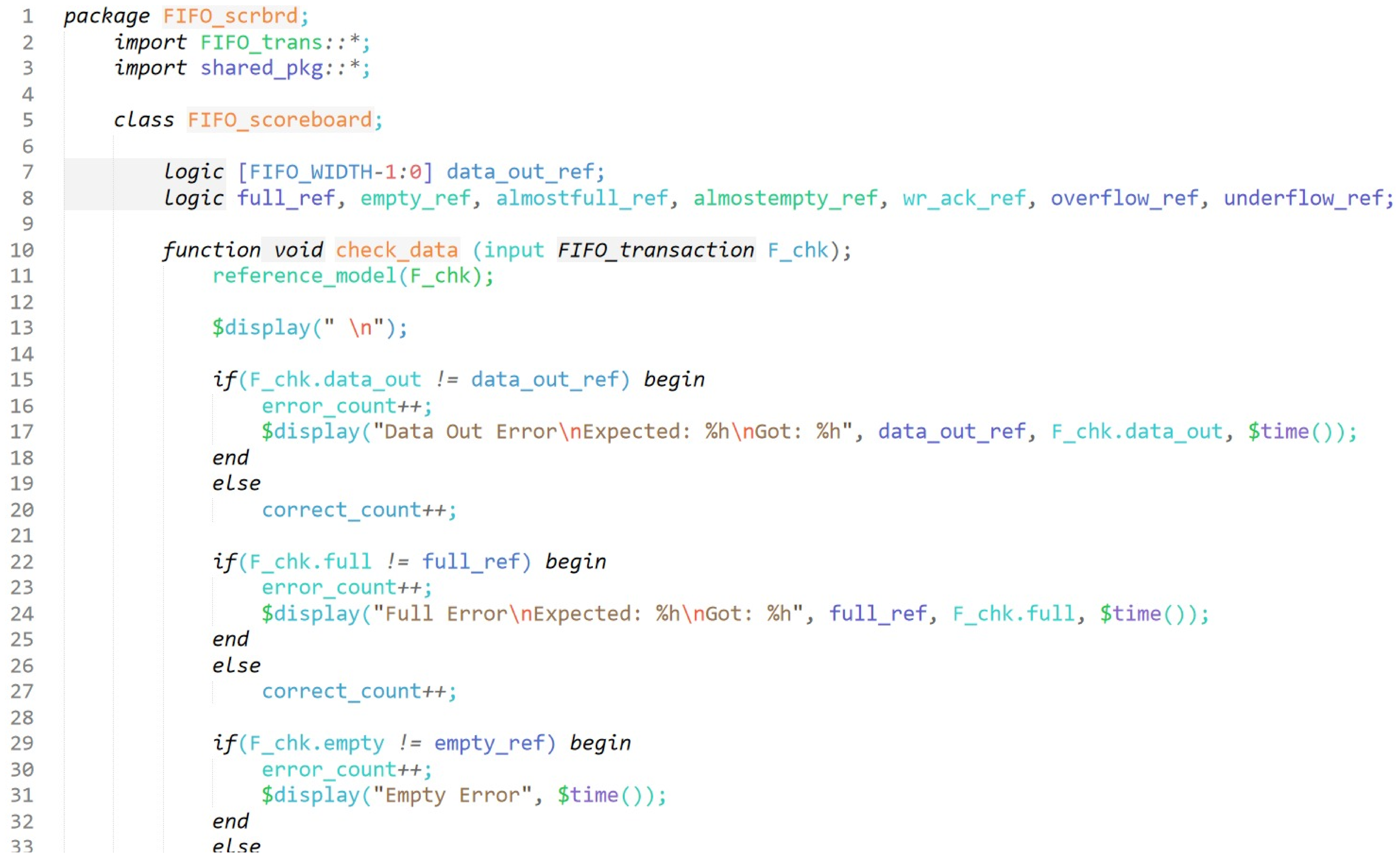
```
package shared_pkg;

    int error_count = 0;
    int correct_count = 0;

    bit test_finished = 0;

endpackage : shared_pkg
```



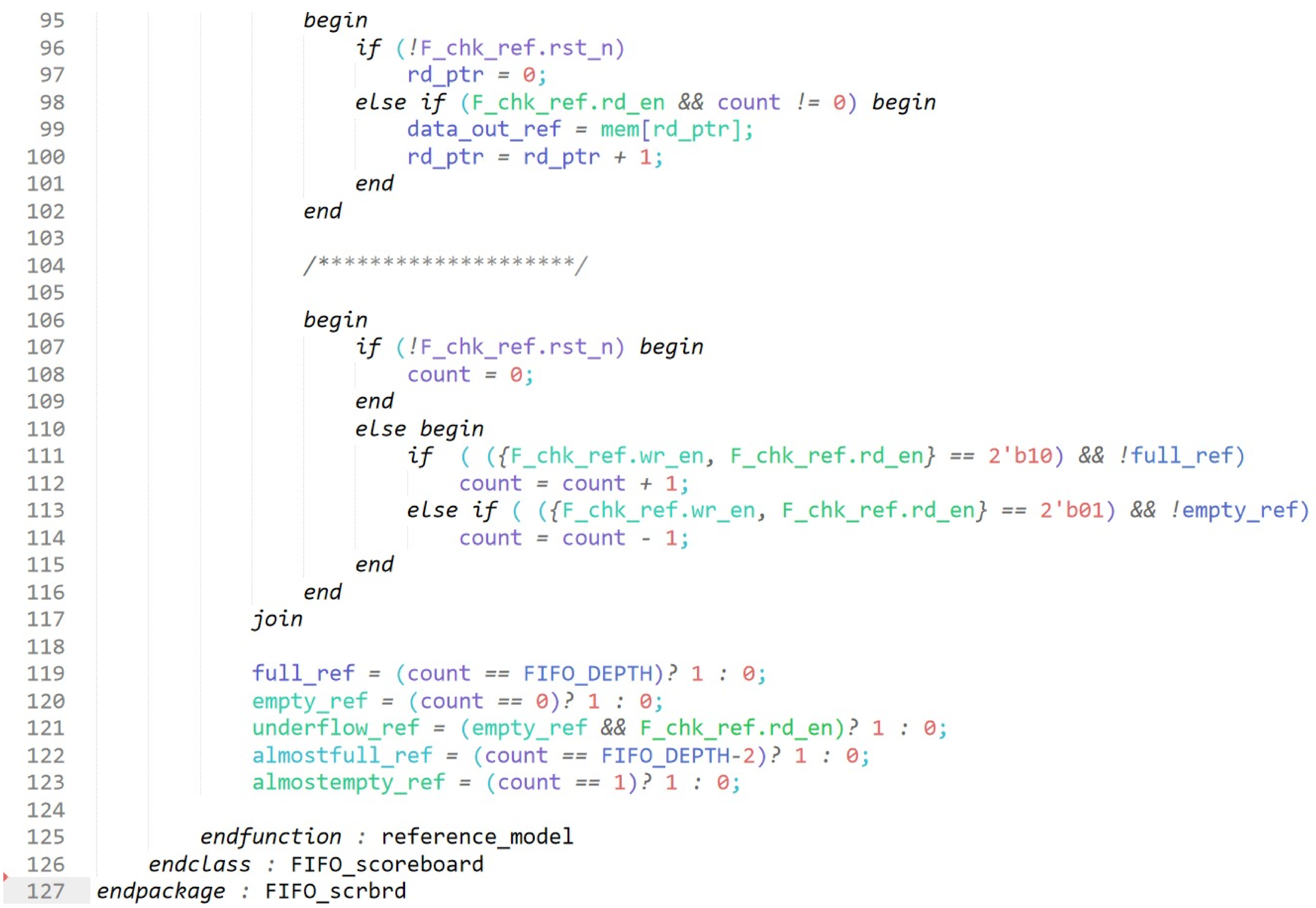
```
package FIFO_trans;

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

    class FIFO_transaction;
        /* Defining the Design I/O as Class Properties */
        rand bit [FIFO_WIDTH-1:0] data_in;
        rand bit wr_en, rd_en, rst_n;
        bit full, empty, almostfull, almostempty, wr_ack, overflow, underflow;
        bit [FIFO_WIDTH-1:0] data_out;

        int WR_EN_ON_DIST = 70, RD_EN_ON_DIST = 30;

        /* Constraint Blocks */
        constraint rstConstr {
            rst_n dist {0 := 4, 1 := 96};
        }

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

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

    endclass : FIFO_transaction

endpackage : FIFO_trans
```



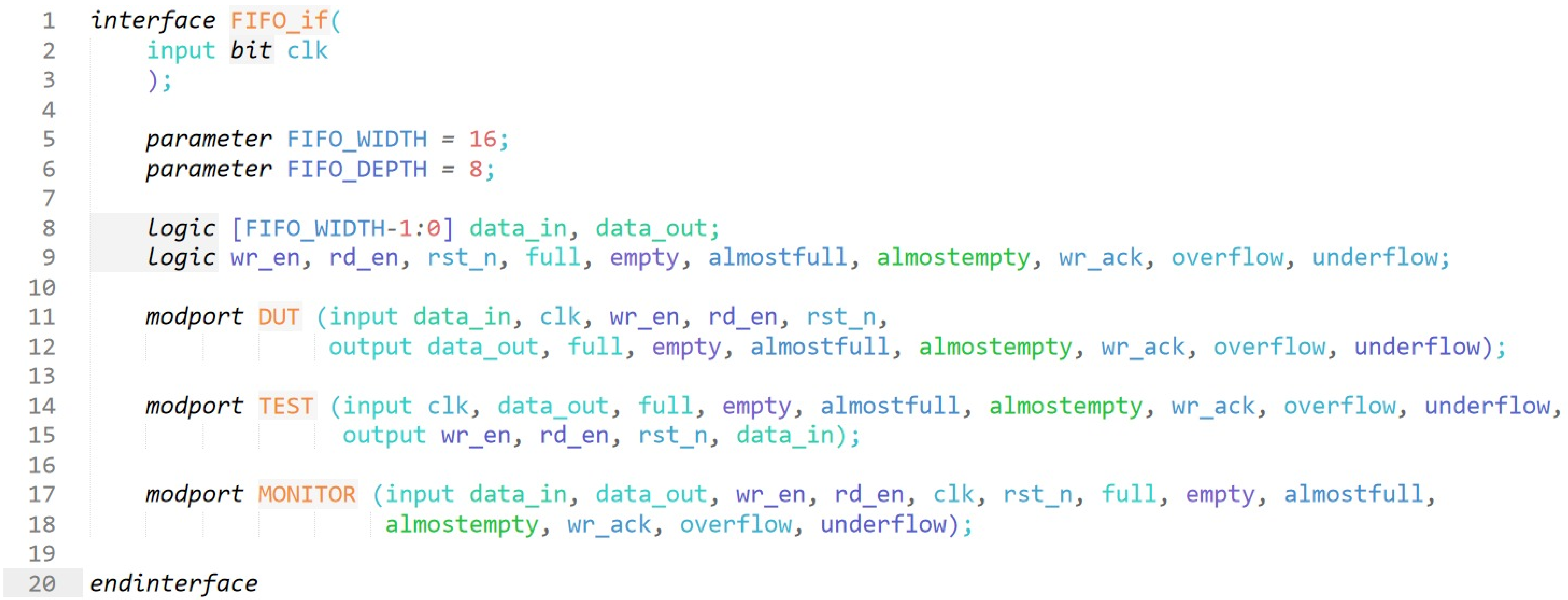
```
package FIFO_cvg;
    import FIFO_trans::*;
    class FIFO_coverage;
        FIFO_transaction F_cvg_Txn = new();

        function void sample_data(input FIFO_transaction F_txn);
            F_cvg_Txn = F_txn;
            cg.sample;
        endfunction : sample_data

        covergroup cg();
           write_en_cp:      coverpoint F_cvg_Txn.wr_en;
           read_en_cp:       coverpoint F_cvg_Txn.rd_en;
           full_cp:          coverpoint F_cvg_Txn.full;
           empty_cp:         coverpoint F_cvg_Txn.empty;
           almostFull_cp:    coverpoint F_cvg_Txn.almostfull;
           almostEmpty_cp:   coverpoint F_cvg_Txn.almostempty;
           overflow_cp:      coverpoint F_cvg_Txn.overflow;
           underflow_cp:     coverpoint F_cvg_Txn.underflow;
           ack_cp:           coverpoint F_cvg_Txn.wr_ack;

           full_cross:       cross write_en_cp, read_en_cp, full_cp;
           empty_cross:      cross write_en_cp, read_en_cp, empty_cp;
           almostF_cross:    cross write_en_cp, read_en_cp, almostFull_cp;
           almostE_cross:    cross write_en_cp, read_en_cp, almostEmpty_cp;
           ovf_cross:        cross write_en_cp, read_en_cp, overflow_cp;
           undf_cross:       cross write_en_cp, read_en_cp, underflow_cp;
           ack_cross:        cross write_en_cp, read_en_cp, ack_cp;
        endgroup : cg

        function new ();
            cg = new;
        endfunction : new
    endclass : FIFO_coverage

endpackage : FIFO_cvg
```



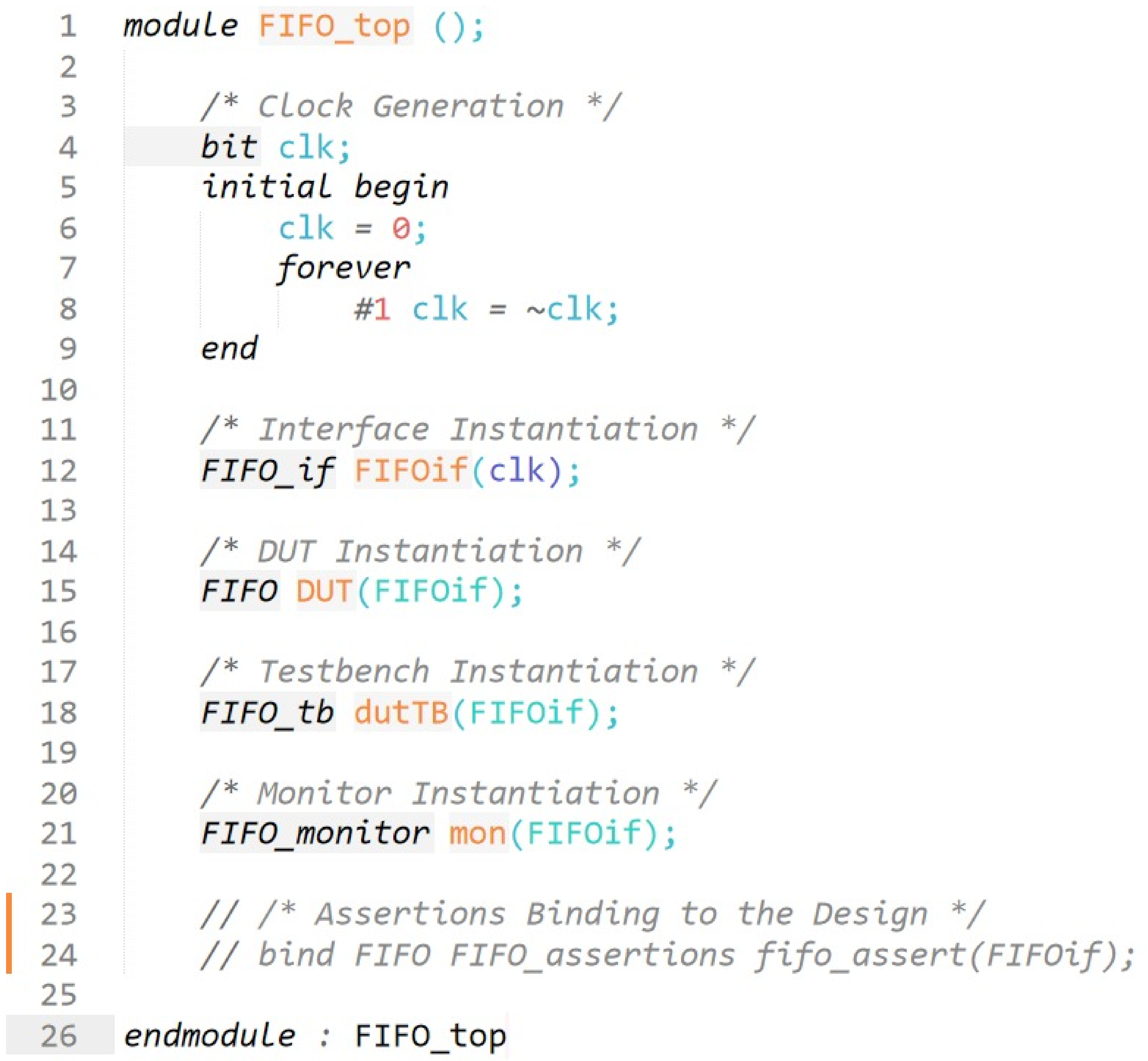
```
package FIFO_scrbrd;
    import FIFO_trans::*;
    import shared_pkg::*;

    class FIFO_scoreboard;

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

        function void check_data (input FIFO_transaction F_chk);
            reference_model(F_chk);

            $display(" \n");

            if(F_chk.data_out != data_out_ref) begin
                error_count++;
                $display("Data Out Error\nExpected: %h\nGot: %h", data_out_ref, F_chk.data_out, $time());
            end
            else
                correct_count++;

            if(F_chk.full != full_ref) begin
                error_count++;
                $display("Full Error\nExpected: %h\nGot: %h", full_ref, F_chk.full, $time());
            end
            else
                correct_count++;

            if(F_chk.empty != empty_ref) begin
                error_count++;
                $display("Empty Error", $time());
            end
            else
            else
                correct_count++;

            if(F_chk.almostfull != almostfull_ref) begin
                error_count++;
                $display("Almost Full Error", $time());
            end
            else
                correct_count++;

            if(F_chk.almostempty != almostempty_ref) begin
                error_count++;
                $display("Almost Empty Error", $time());
            end
            else
                correct_count++;

            if(F_chk.overflow != overflow_ref) begin
                error_count++;
                $display("Overflow Error", $time());
            end
            else
                correct_count++;

            if(F_chk.underflow != underflow_ref) begin
                error_count++;
                $display("Underflow Error", $time());
            end
            else
                correct_count++;

            if(F_chk.wr_ack != wr_ack_ref) begin
                error_count++;
            if(F_chk.wr_ack != wr_ack_ref) begin
                error_count++;
                $display("Write Ack Error", $time());
            end
            else
                correct_count++;
        endfunction : check_data

        function void reference_model (input FIFO_transaction F_chk_ref);
            int wr_ptr, rd_ptr;
            int count;
            bit [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];

            fork
                begin
                    if (!F_chk_ref.rst_n)
                        wr_ptr = 0;
                    else if (F_chk_ref.wr_en && count < FIFO_DEPTH) begin
                        mem[wr_ptr] = F_chk_ref.data_in;
                        wr_ack_ref = 1;
                        wr_ptr = wr_ptr + 1;
                    end
                    else begin
                        wr_ack_ref = 0;
                        if (full_ref & F_chk_ref.wr_en)
                            overflow_ref = 1;
                        else
                            overflow_ref = 0;
                    end
                end

                begin
                    if (!F_chk_ref.rst_n)
```



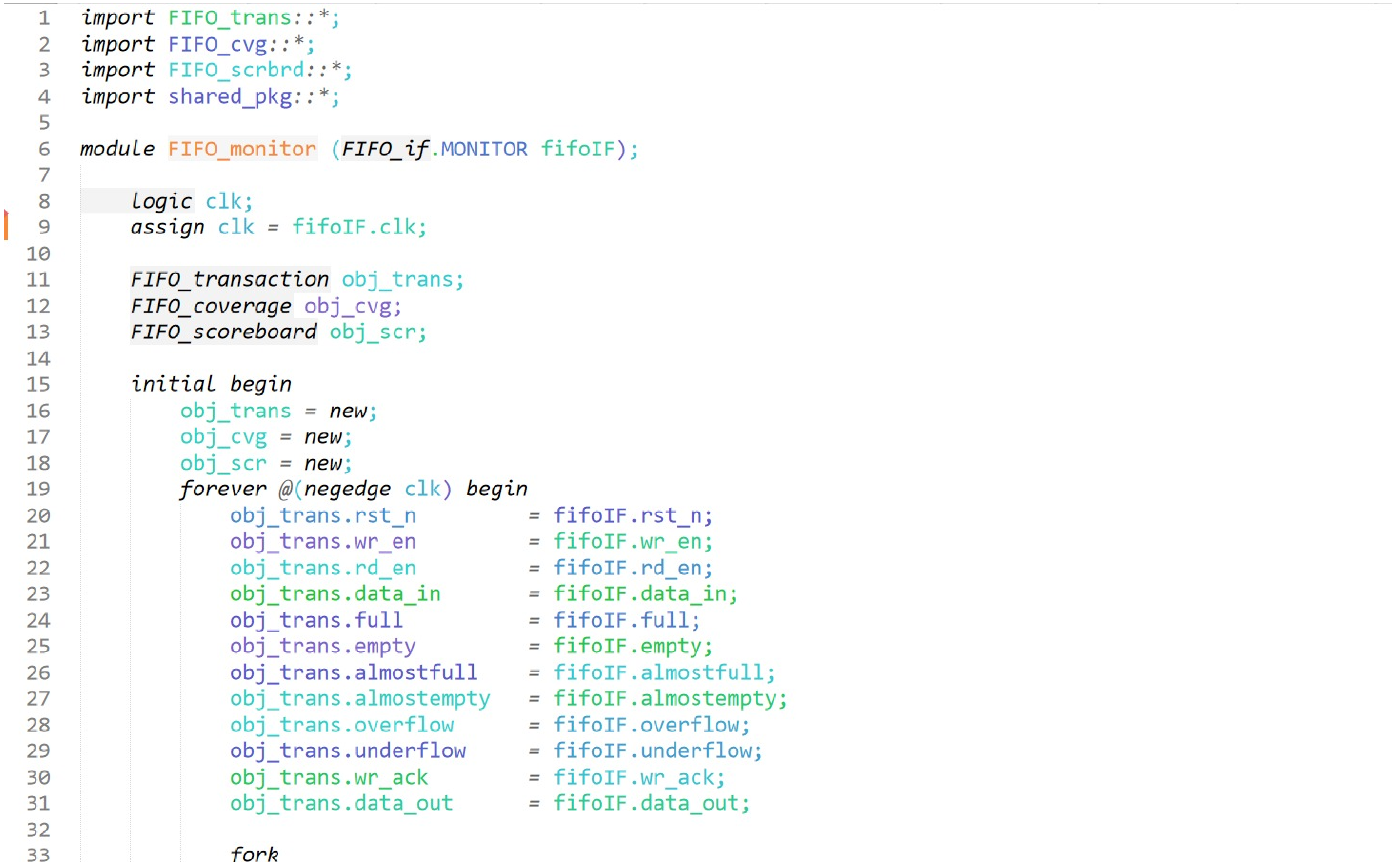
```
                begin
                    if (!F_chk_ref.rst_n)
                        rd_ptr = 0;
                    else if (F_chk_ref.rd_en && count != 0) begin
                        data_out_ref = mem[rd_ptr];
                        rd_ptr = rd_ptr + 1;
                    end
                end

                /********************/

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

            full_ref = (count == FIFO_DEPTH)? 1 : 0;
            empty_ref = (count == 0)? 1 : 0;
            underflow_ref = (empty_ref && F_chk_ref.rd_en)? 1 : 0;
            almostfull_ref = (count == FIFO_DEPTH-2)? 1 : 0;
            almostempty_ref = (count == 1)? 1 : 0;

        endfunction : reference_model
    endclass : FIFO_scoreboard
endpackage : FIFO_scrbrd
```



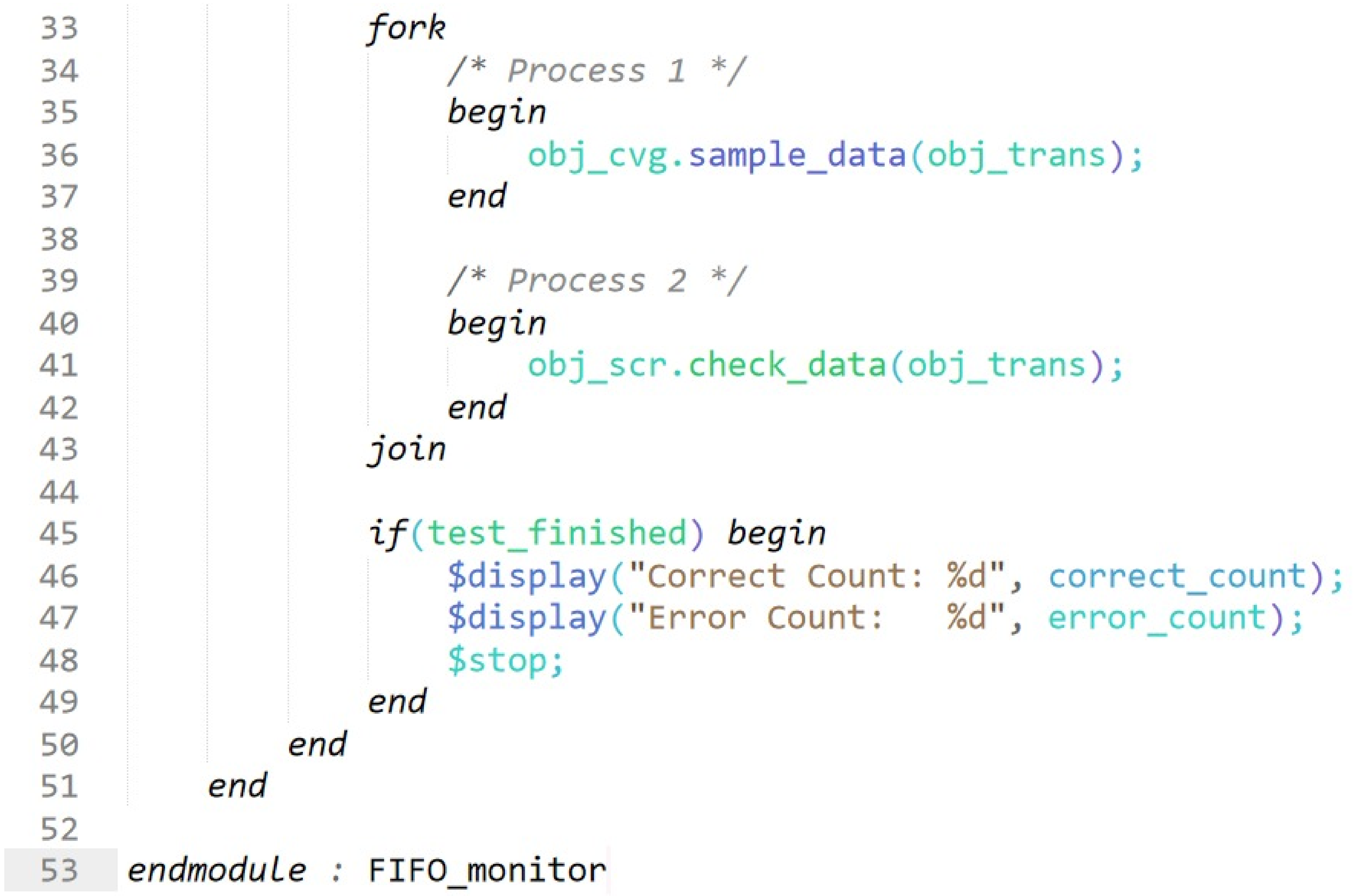
```systemverilog
interface FIFO_if(
    input bit clk
    );

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

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

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

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

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

endinterface
```



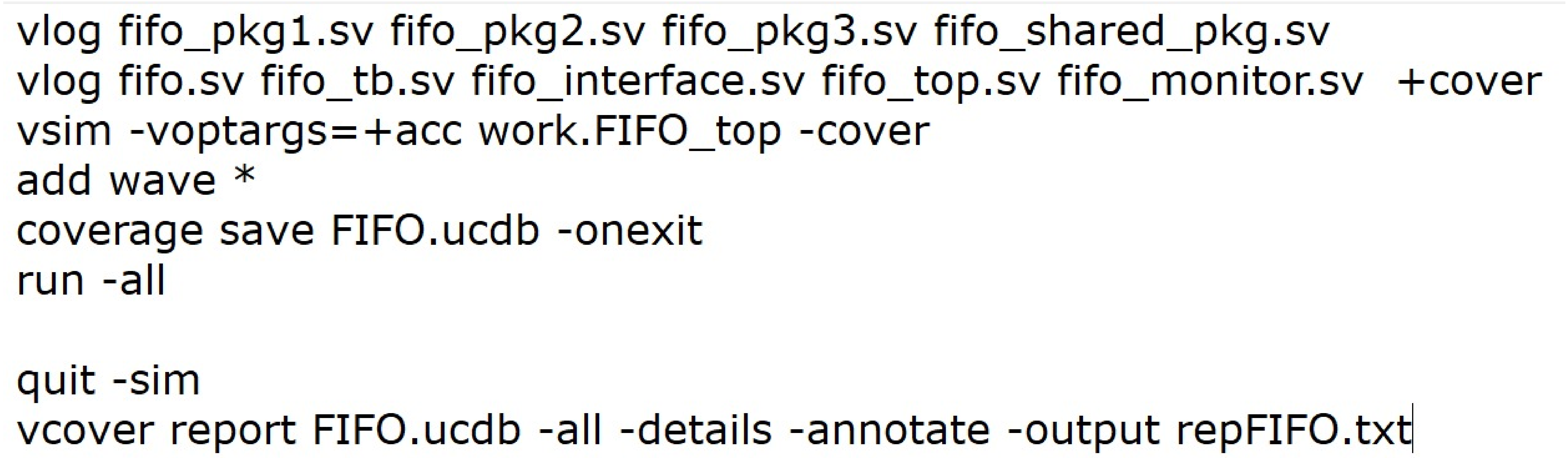
```
module FIFO_top ();

    /* Clock Generation */
    bit clk;
    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    /* Interface Instantiation */
    FIFO_if FIFOif(clk);

    /* DUT Instantiation */
    FIFO DUT(FIFOif);

    /* Testbench Instantiation */
    FIFO_tb dutTB(FIFOif);

    /* Monitor Instantiation */
    FIFO_monitor mon(FIFOif);

    // /* Assertions Binding to the Design */
    // bind FIFO FIFO_assertions fifo_assert(FIFOif);

endmodule : FIFO_top
```



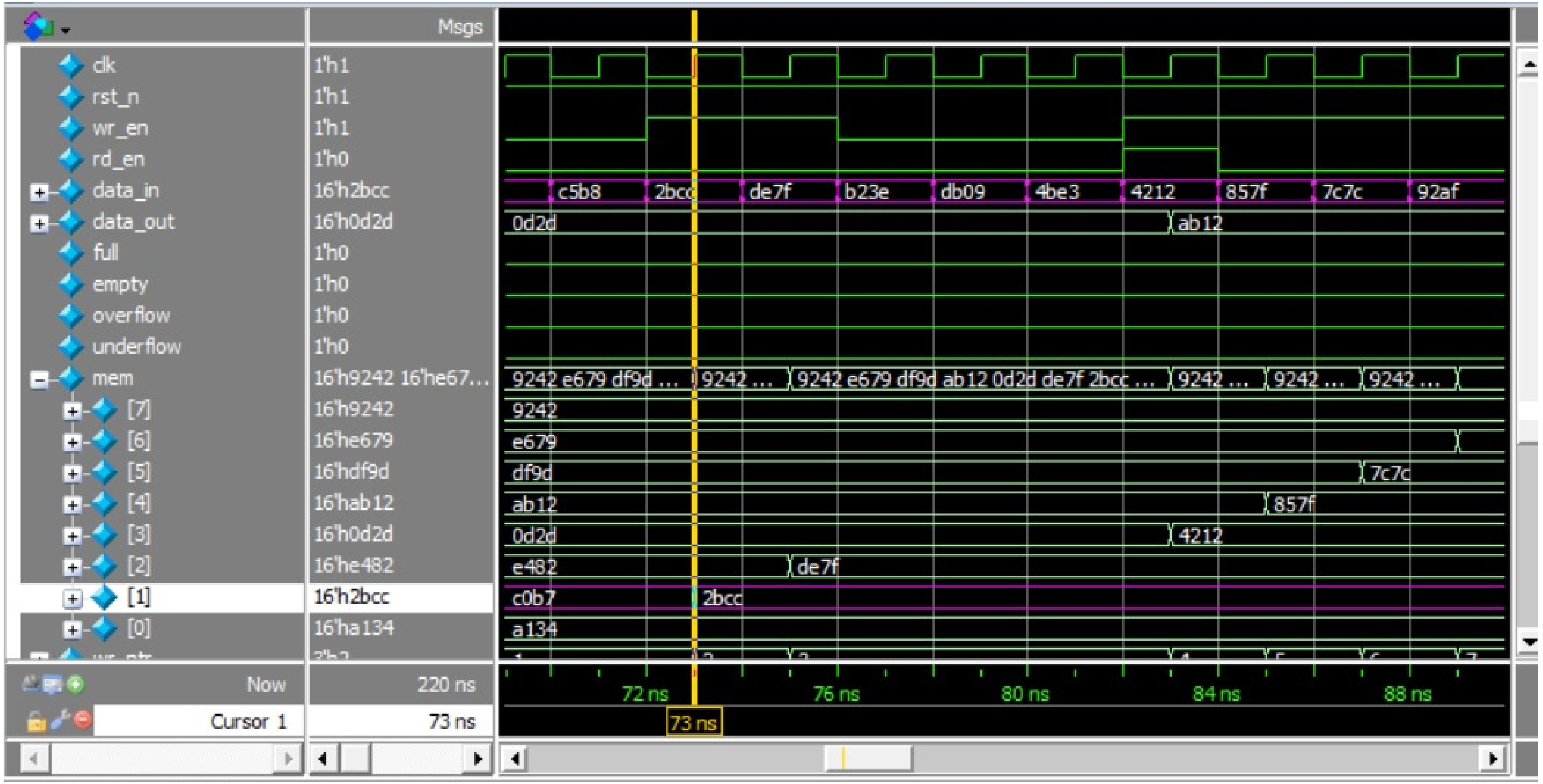
```systemverilog
import FIFO_trans::*;
import FIFO_cvg::*;
import FIFO_scrbrd::*;
import shared_pkg::*;

module FIFO_monitor (FIFO_if.MONITOR fifoIF);

    logic clk;
    assign clk = fifoIF.clk;

    FIFO_transaction obj_trans;
    FIFO_coverage obj_cvg;
    FIFO_scoreboard obj_scr;

    initial begin
        obj_trans = new;
        obj_cvg = new;
        obj_scr = new;
        forever @(negedge clk) begin
            obj_trans.rst_n         = fifoIF.rst_n;
            obj_trans.wr_en         = fifoIF.wr_en;
            obj_trans.rd_en         = fifoIF.rd_en;
            obj_trans.data_in       = fifoIF.data_in;
            obj_trans.full          = fifoIF.full;
            obj_trans.empty         = fifoIF.empty;
            obj_trans.almostfull    = fifoIF.almostfull;
            obj_trans.almostempty   = fifoIF.almostempty;
            obj_trans.overflow      = fifoIF.overflow;
            obj_trans.underflow     = fifoIF.underflow;
            obj_trans.wr_ack        = fifoIF.wr_ack;
            obj_trans.data_out      = fifoIF.data_out;

            fork
```



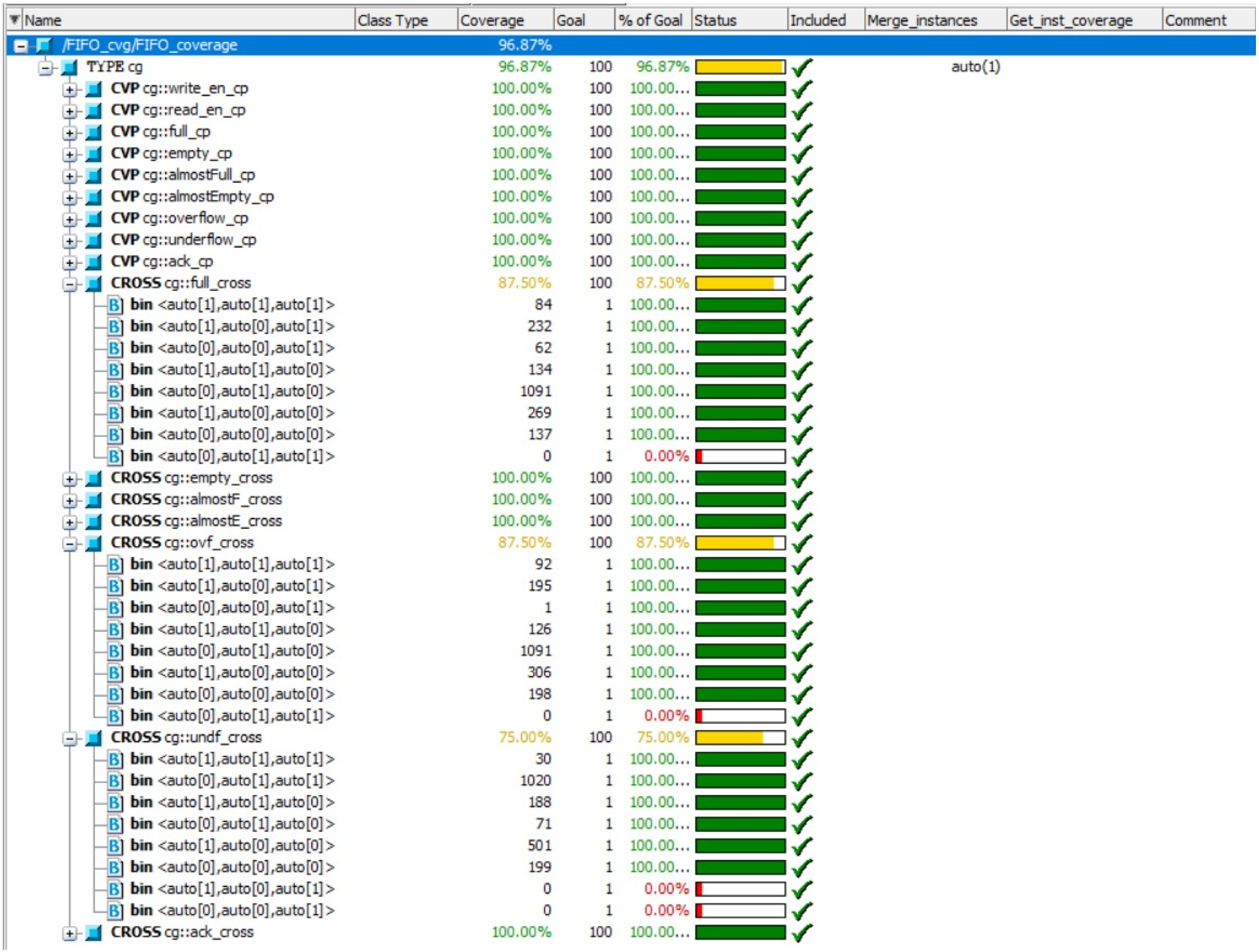
```
            fork
                /* Process 1 */
                begin
                    obj_cvg.sample_data(obj_trans);
                end

                /* Process 2 */
                begin
                    obj_scr.check_data(obj_trans);
                end
            join

            if(test_finished) begin
                $display("Correct Count: %d", correct_count);
                $display("Error Count:   %d", error_count);
                $stop;
            end
        end
    end

endmodule : FIFO_monitor
```



```
vlog fifo_pkg1.sv fifo_pkg2.sv fifo_pkg3.sv fifo_shared_pkg.sv
vlog fifo.sv fifo_tb.sv fifo_interface.sv fifo_top.sv fifo_monitor.sv  +cover
vsim -voptargs=+acc work.FIFO_top -cover
add wave *
coverage save FIFO.ucdb -onexit
run -all

quit -sim
vcover report FIFO.ucdb -all -details -annotate -output repFIFO.txt
```



```
                     Msgs
clk                  1'h1
rst_n                1'h1
wr_en                1'h1
rd_en                1'h0
data_in              16'h2bcc
data_out             16'h0d2d
full                 1'h0
empty                1'h0
overflow             1'h0
underflow            1'h0
mem                  16'h9242 16'he67...
  [7]                16'h9242
  [6]                16'he679
  [5]                16'hdf9d
  [4]                16'hab12
  [3]                16'h0d2d
  [2]                16'he482
  [1]                16'h2bcc
  [0]                16'ha134
wr_ptr               3'h2
Now                  220 ns
Cursor 1             73 ns
```



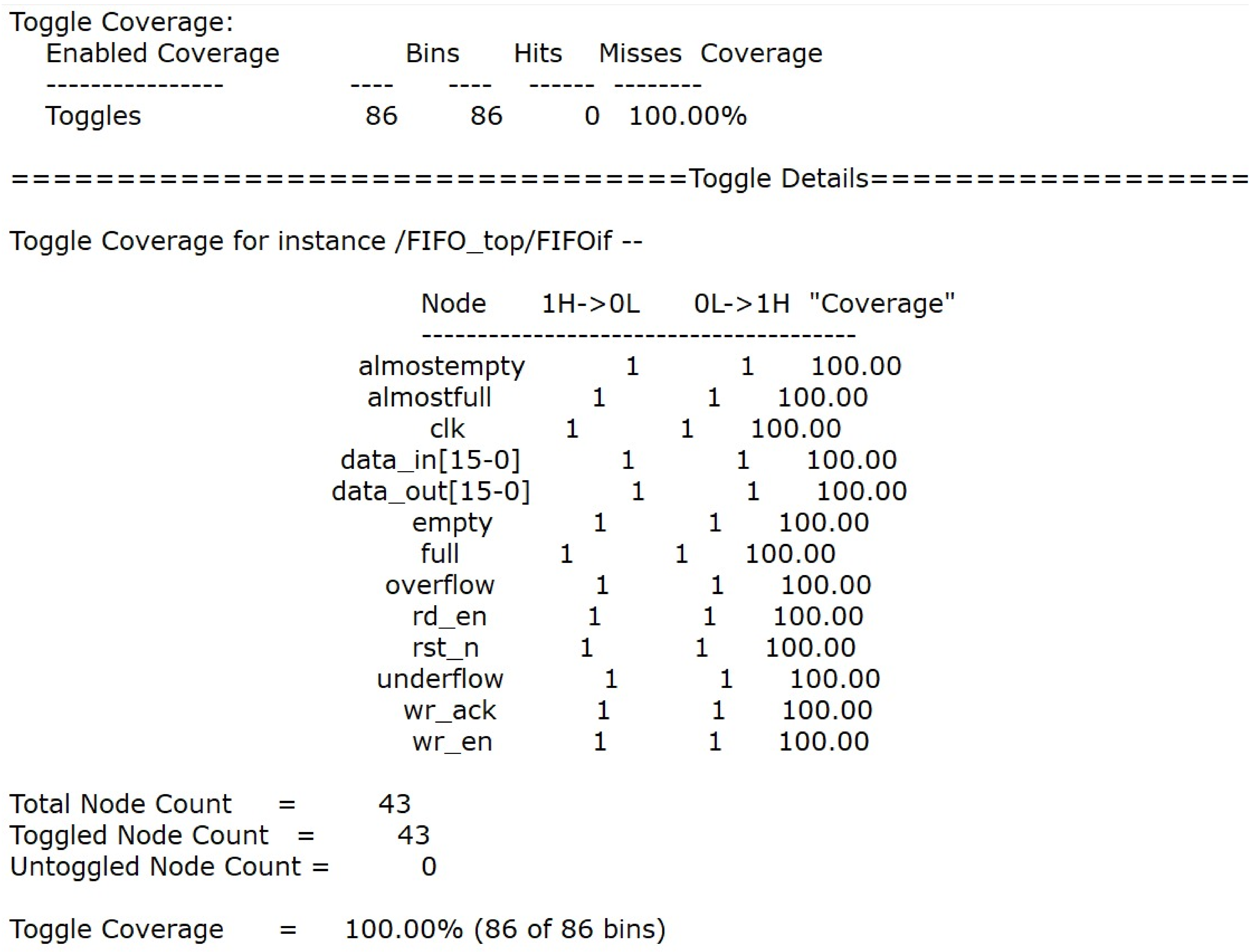
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_cvg/FIFO_coverage		96.87%							
TYPE cg		96.87%	100	96.87%	🟨	✓		auto(1)	
CVP cg::write_en_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::read_en_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::full_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::empty_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::almostFull_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::almostEmpty_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::overflow_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::underflow_cp		100.00%	100	100.00...	🟩	✓			
CVP cg::ack_cp		100.00%	100	100.00...	🟩	✓			
CROSS cg::full_cross		87.50%	100	87.50%	🟨	✓			
bin <auto[1],auto[1],auto[1]>		84	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[1]>		232	1	100.00...	🟩	✓			
bin <auto[0],auto[0],auto[1]>		62	1	100.00...	🟩	✓			
bin <auto[1],auto[1],auto[0]>		134	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[0]>		1091	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[0]>		269	1	100.00...	🟩	✓			
bin <auto[0],auto[0],auto[0]>		137	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[1]>		0	1	0.00%	🟥	✓			
CROSS cg::empty_cross		100.00%	100	100.00...	🟩	✓			
CROSS cg::almostF_cross		100.00%	100	100.00...	🟩	✓			
CROSS cg::almostE_cross		100.00%	100	100.00...	🟩	✓			
CROSS cg::ovf_cross		87.50%	100	87.50%	🟨	✓			
bin <auto[1],auto[1],auto[1]>		92	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[1]>		195	1	100.00...	🟩	✓			
bin <auto[0],auto[0],auto[1]>		1	1	100.00...	🟩	✓			
bin <auto[1],auto[1],auto[0]>		126	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[0]>		1091	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[0]>		306	1	100.00...	🟩	✓			
bin <auto[0],auto[0],auto[0]>		198	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[1]>		0	1	0.00%	🟥	✓			
CROSS cg::undf_cross		75.00%	100	75.00%	🟨	✓			
bin <auto[1],auto[1],auto[1]>		30	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[1]>		1020	1	100.00...	🟩	✓			
bin <auto[1],auto[1],auto[0]>		188	1	100.00...	🟩	✓			
bin <auto[0],auto[1],auto[0]>		71	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[0]>		501	1	100.00...	🟩	✓			
bin <auto[0],auto[0],auto[0]>		199	1	100.00...	🟩	✓			
bin <auto[1],auto[0],auto[1]>		0	1	0.00%	🟥	✓			
bin <auto[0],auto[0],auto[1]>		0	1	0.00%	🟥	✓			
CROSS cg::ack_cross		100.00%	100	100.00...	🟩	✓			

Justification:

1. Full_cross: "full" flag will never be high while wr_en = 0 and rd_En = 1 (count < DEPTH).

2. OVF_cross: "overflow" flag will never be high while wr_en = 0 & rd_en = 1 (full = 0).

3. UNDF_cross: "underflow" flag will never be high while wr_en = 1 and rd_en = 0 or while wr_en = 0 and rd_en = 0.

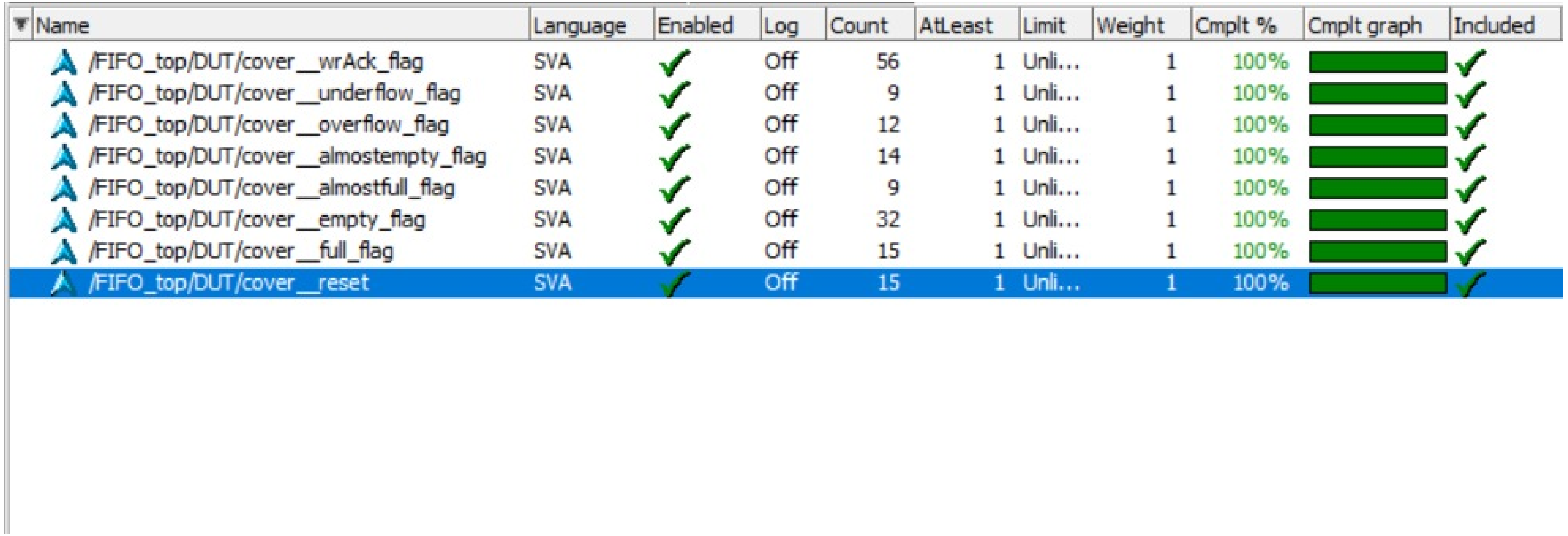


```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

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

Toggle Coverage for instance /FIFO_top/FIFOif --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ---------------------------------------
                                   almostempty           1           1      100.00
                                    almostfull           1           1      100.00
                                           clk           1           1      100.00
                                 data_in[15-0]           1           1      100.00
                                data_out[15-0]           1           1      100.00
                                         empty           1           1      100.00
                                          full           1           1      100.00
                                      overflow           1           1      100.00
                                         rd_en           1           1      100.00
                                         rst_n           1           1      100.00
                                     underflow           1           1      100.00
                                        wr_ack           1           1      100.00
                                         wr_en           1           1      100.00

Total Node Count     =         43
Toggled Node Count   =         43
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (86 of 86 bins)
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        106       106         0   100.00%

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

Toggle Coverage for instance /FIFO_top/DUT --

                                              Node      1H->0L      0L->1H  "Coverage"
                                              ---------------------------------------
                                       almostempty           1           1      100.00
                                        almostfull           1           1      100.00
                                               clk           1           1      100.00
                                        count[3-0]           1           1      100.00
                                     data_in[15-0]           1           1      100.00
                                    data_out[15-0]           1           1      100.00
                                             empty           1           1      100.00
                                              full           1           1      100.00
                                          overflow           1           1      100.00
                                             rd_en           1           1      100.00
                                       rd_ptr[2-0]           1           1      100.00
                                             rst_n           1           1      100.00
                                         underflow           1           1      100.00
                                            wr_ack           1           1      100.00
                                             wr_en           1           1      100.00
                                       wr_ptr[2-0]           1           1      100.00

Total Node Count     =         53
Toggled Node Count   =         53
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (106 of 106 bins)
```

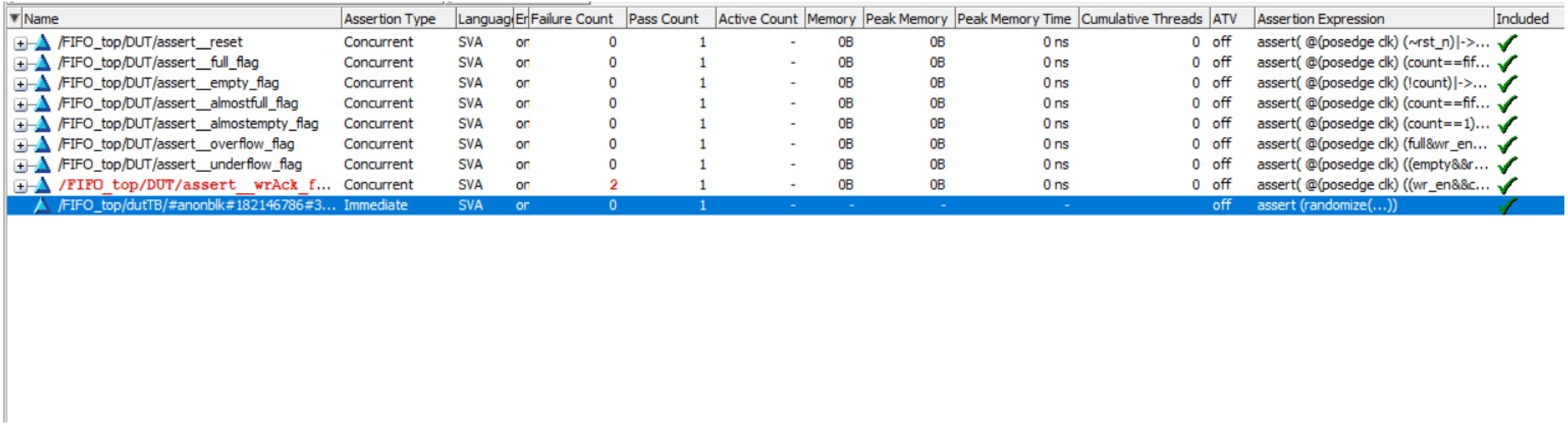
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      24        24         0   100.00%

=================================Statement Details==================================

Statement Coverage for instance /FIFO_top/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File fifo.sv
    8                                                module FIFO(FIFO_if.DUT fifoIF);

    9

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

    11                                                   logic wr_en, rd_en, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow;

    12

    13                                                   assign clk                                          = fifoIF.clk;

    14              1                         13         assign rst_n                                    = fifoIF.rst_n;

    15              1                         35         assign wr_en                                    = fifoIF.wr_en;

    16              1                         37         assign rd_en                                    = fifoIF.rd_en;

    17              1                        101         assign data_in                                  = fifoIF.data_in;

    18                                                   assign fifoIF.full                        = full;
```

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Branches                        21        21         0   100.00%

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

Branch Coverage for instance /FIFO_top/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File fifo.sv
------------------------------------IF Branch------------------------------------
    35                                       107     Count coming in to IF
    35              1                         12                         if (!rst_n) begin

    38              1                         54                         else if (wr_en && (count < fifoIF.FIFO_DEPTH)

    43              1                         41                         else begin

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    45                                        41     Count coming in to IF
    45              1                         12                                     if (full & wr_en)

    47              1                         29                                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    53                                        89     Count coming in to IF
```

```
Directive Coverage:
    Directives                     8        8        0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/FIFO_top/DUT/cover__wrAck_flag         FIFO   Verilog  SVA  fifo.sv(137)     56 Covered
/FIFO_top/DUT/cover__underflow_flag     FIFO   Verilog  SVA  fifo.sv(136)      9 Covered
/FIFO_top/DUT/cover__overflow_flag      FIFO   Verilog  SVA  fifo.sv(135)     12 Covered
/FIFO_top/DUT/cover__almostempty_flag   FIFO   Verilog  SVA  fifo.sv(134)     14 Covered
/FIFO_top/DUT/cover__almostfull_flag    FIFO   Verilog  SVA  fifo.sv(133)      9 Covered
/FIFO_top/DUT/cover__empty_flag         FIFO   Verilog  SVA  fifo.sv(132)     32 Covered
/FIFO_top/DUT/cover__full_flag          FIFO   Verilog  SVA  fifo.sv(131)     15 Covered
/FIFO_top/DUT/cover__reset              FIFO   Verilog  SVA  fifo.sv(130)     15 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      24        24         0   100.00%
```



Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/FIFO_top/DUT/cover__wrAck_flag	SVA	✓	Off	56	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__underflow_flag	SVA	✓	Off	9	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__overflow_flag	SVA	✓	Off	12	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__almostempty_flag	SVA	✓	Off	14	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__almostfull_flag	SVA	✓	Off	9	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__empty_flag	SVA	✓	Off	32	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__full_flag	SVA	✓	Off	15	1	Unli...	1	100%	🟩	✓
/FIFO_top/DUT/cover__reset	SVA	✓	Off	15	1	Unli...	1	100%	🟩	✓



Name	Assertion Type	Languag	Er	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/FIFO_top/DUT/assert__reset	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|->...	✓
/FIFO_top/DUT/assert__full_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==fif...	✓
/FIFO_top/DUT/assert__empty_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (!count)|->...	✓
/FIFO_top/DUT/assert__almostfull_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==fif...	✓
/FIFO_top/DUT/assert__almostempty_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==1)...	✓
/FIFO_top/DUT/assert__overflow_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (full&wr_en...	✓
/FIFO_top/DUT/assert__underflow_flag	Concurrent	SVA	or	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) ((empty&&r...	✓
/FIFO_top/DUT/assert__wrAck_f...	Concurrent	SVA	or	2	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) ((wr_en&&c...	✓
/FIFO_top/dutTB/#anonblk#182146786#3...	Immediate	SVA	or	0	1	-	-	-	-		off	assert (randomize(...))	✓