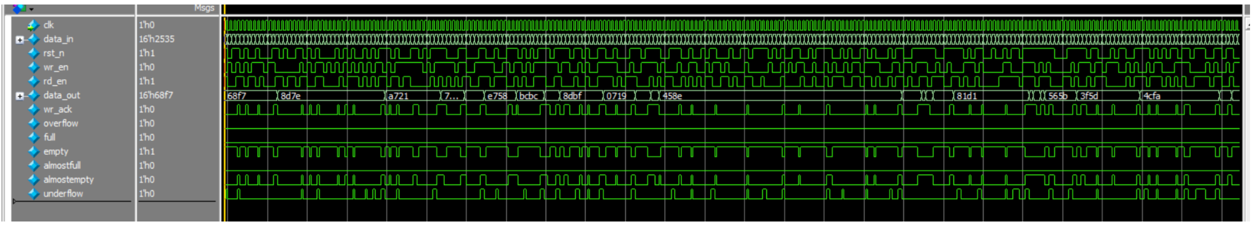
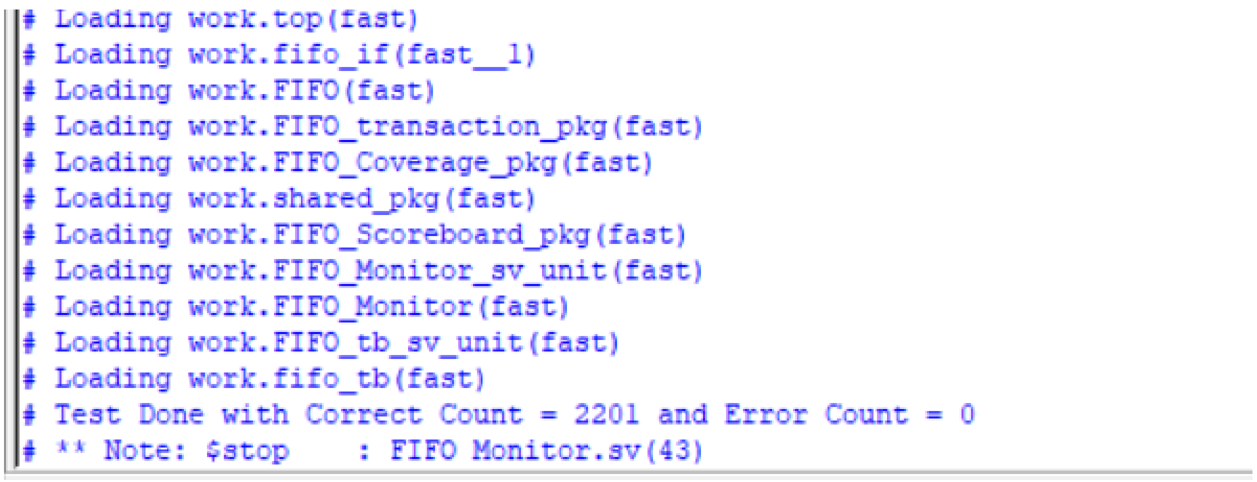
Project 1

Verification Plan:

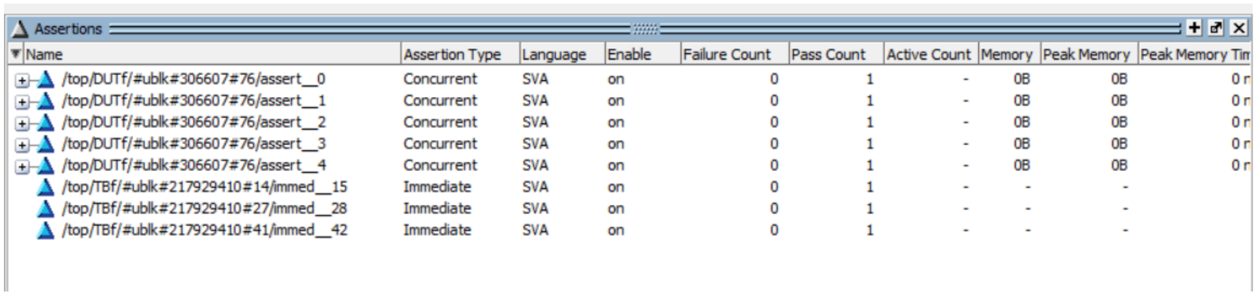
Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted, the output signals are all desserted except the empty flag	Directed at the start of the simulation, then randomized with constraint that drive the reset to be off most of the simulation time		A checker through a reference model task to make sure the output is correct
FIFO_2	When the wr_en signal is asserted, rd_en is desserted and data_in take randomized value while the memory isn't full: the memory with address = wr_ptr will take the data_in value, wr_ack will be asserted and counter is incremented	Randomization Under wr_only constraint that the value of wr_en to be on, rd_en to be off and reset signal to be desserted	Coverpoint for wr_en, rd_en and wr_ack	A checker through a reference model task and assertions for output flags to make sure the output is correct
FIFO_3	When the rd_en signal is asserted, wr_en is desserted while the memory isn't empty: the data_out = memory with address = rd_ptr	Randomization Under rd_only constraint that the value of wr_en to be on, rd_en to be off and reset signal to be desserted	Coverpoint for wr_en, rd_en and wr_ack	A checker through a reference model task and assertions for output flags to make sure the output is correct
FIFO_4	When the rd_en signal and wr_en are asserted, while the memory isn't empty or full: the data_out = memory with address = rd_ptr and the memory with address = wr_ptr will take the data_in value, wr_ack will be asserted and counter won't be affected	Randomization Under constraints that the wr_en to be on 70% of the time while rd_en is on for 30% of the time	Coverpoint for wr_en, rd_en and wr_ack	A checker through a reference model task and assertions for output flags to make sure the output is correct
FIFO_5	When the rd_en signal and wr_en are asserted, while the memory is empty: the memory with address = wr_ptr will take the data_in value, wr_ack will be asserted and counter will be incremented	Randomization Under constraints that the wr_en to be on 70% of the time while rd_en is on for 30% of the time	Coverpoint for wr_en, rd_en and empty and for wr_en, rd_en and almost empty	A checker through a reference model task and assertions for output flags to make sure the output is correct
FIFO_7	When the wr_en signal is asserted, rd_en is desserted and data_in take randomized value while the memory is full: overflow will be asserted, wr_ack will be desserted and counter isn't changed	Randomization Under constraints that the wr_en to be on 70% of the time while rd_en is on for 30% of the time	Coverpoint for wr_en, rd_en and overflow	A checker through a reference model task and assertions for output flags to make sure the output is correct
FIFO_8	When the rd_en signal is asserted, wr_en is desserted while the memory is empty: underflow is asserted and counter isn't changed	Randomization Under constraints that the wr_en to be on 70% of the time while rd_en is on for 30% of the time	Coverpoint for wr_en, rd_en and underflow	A checker through a reference model task and assertions for output flags to make sure the output is correct

Questasim snippets:



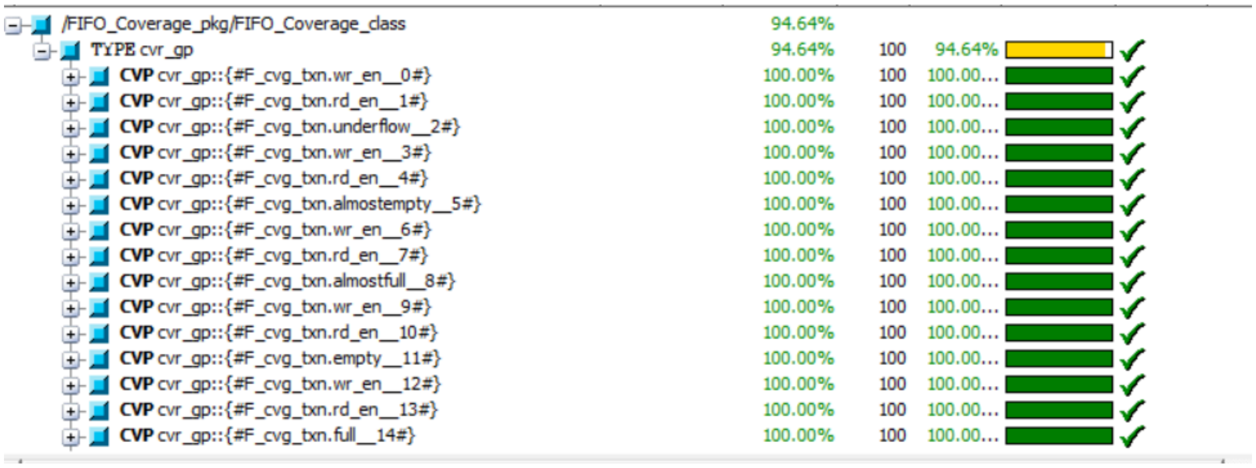


```
# Loading work.top(fast)
# Loading work.fifo_if(fast__1)
# Loading work.FIFO(fast)
# Loading work.FIFO_transaction_pkg(fast)
# Loading work.FIFO_Coverage_pkg(fast)
# Loading work.shared_pkg(fast)
# Loading work.FIFO_Scoreboard_pkg(fast)
# Loading work.FIFO_Monitor_sv_unit(fast)
# Loading work.FIFO_Monitor(fast)
# Loading work.FIFO_tb_sv_unit(fast)
# Loading work.fifo_tb(fast)
# Test Done with Correct Count = 2201 and Error Count = 0
# ** Note: $stop    : FIFO_Monitor.sv(43)
```



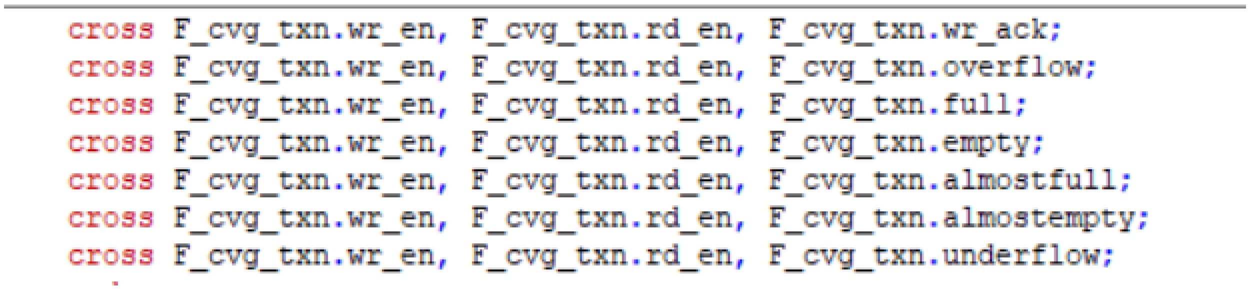
Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Tim
/top/DUTf/#ublk#306607#76/assert__0	Concurrent	SVA	on	0	1	-	0B	0B	0 r
/top/DUTf/#ublk#306607#76/assert__1	Concurrent	SVA	on	0	1	-	0B	0B	0 r
/top/DUTf/#ublk#306607#76/assert__2	Concurrent	SVA	on	0	1	-	0B	0B	0 r
/top/DUTf/#ublk#306607#76/assert__3	Concurrent	SVA	on	0	1	-	0B	0B	0 r
/top/DUTf/#ublk#306607#76/assert__4	Concurrent	SVA	on	0	1	-	0B	0B	0 r
/top/TBf/#ublk#217929410#14/immed__15	Immediate	SVA	on	0	1	-	-	-	
/top/TBf/#ublk#217929410#27/immed__28	Immediate	SVA	on	0	1	-	-	-	
/top/TBf/#ublk#217929410#41/immed__42	Immediate	SVA	on	0	1	-	-	-	

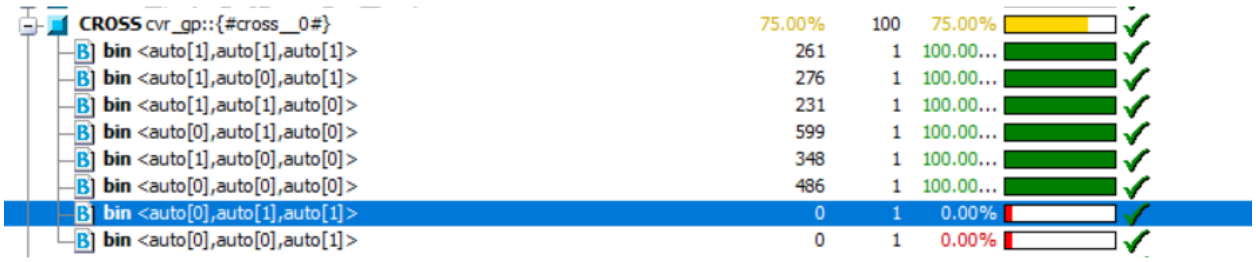


/FIFO_Coverage_pkg/FIFO_Coverage_class	94.64%				
TYPE cvr_gp	94.64%	100	94.64%		✓
CVP cvr_gp::{#F_cvg_txn.wr_en__0#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.rd_en__1#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.underflow__2#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.wr_en__3#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.rd_en__4#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.almostempty__5#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.wr_en__6#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.rd_en__7#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.almostfull__8#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.wr_en__9#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.rd_en__10#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.empty__11#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.wr_en__12#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.rd_en__13#}	100.00%	100	100.00...	🟩	✓
CVP cvr_gp::{#F_cvg_txn.full__14#}	100.00%	100	100.00...	🟩	✓

It's not 100% as it's impossible for the wr_en to be de-asserted while the wr_ack is asserted in the same instance



```
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.wr_ack;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.overflow;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.full;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.empty;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostfull;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostempty;
cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.underflow;
```



CROSS cvr_gp::{#cross__0#}	75.00%	100	75.00%		✓
bin <auto[1],auto[1],auto[1]>	261	1	100.00...	🟩	✓
bin <auto[1],auto[0],auto[1]>	276	1	100.00...	🟩	✓
bin <auto[1],auto[1],auto[0]>	231	1	100.00...	🟩	✓
bin <auto[0],auto[1],auto[0]>	599	1	100.00...	🟩	✓
bin <auto[1],auto[0],auto[0]>	348	1	100.00...	🟩	✓
bin <auto[0],auto[0],auto[0]>	486	1	100.00...	🟩	✓
bin <auto[0],auto[1],auto[1]>	0	1	0.00%	🟥	✓
bin <auto[0],auto[0],auto[1]>	0	1	0.00%	🟥	✓

Do file:

```
vlib work
vlog -f src_files.txt
vlog -work work -vopt -sv -stats=none
+incdir+path_to_assertions_dir +define+SIM FIFO.sv
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all
add wave -position insertpoint sim:/top/fifoif/*
run -all
```

Original Code:

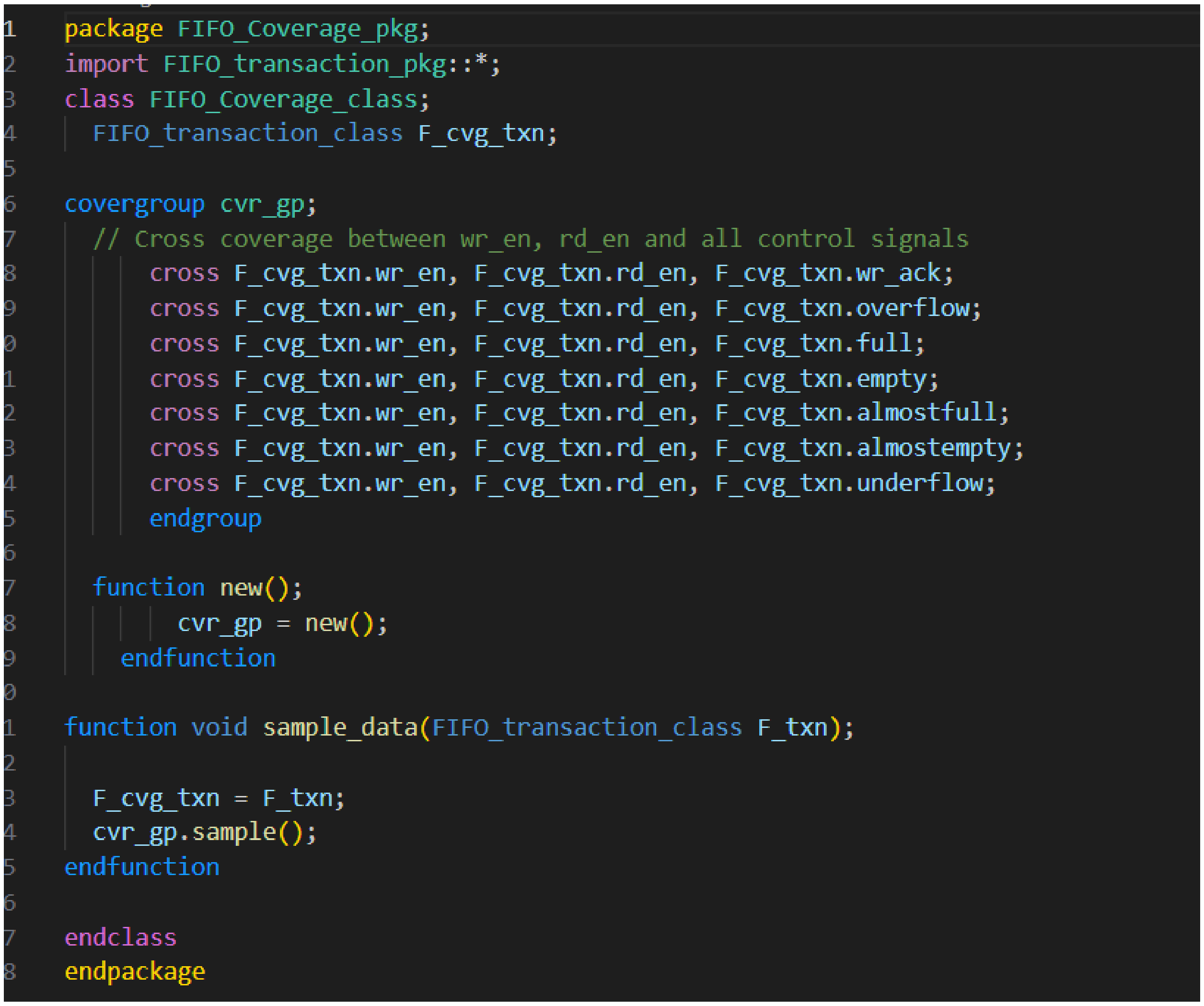
```
module FIFO(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull,
almostempty, wr_ack, overflow, underflow, data_out);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
input [FIFO_WIDTH-1:0] data_in;
input clk, rst_n, wr_en, rd_en;
output reg [FIFO_WIDTH-1:0] data_out;
output reg wr_ack, overflow;
output full, empty, almostfull, almostempty, underflow;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

reg [FIFO_WIDTH-1:0] mem [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;
    end
    else if (wr_en && count < 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 == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign underflow = (empty && rd_en)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
assign almostempty = (count == 1)? 1 : 0;

endmodule
```

Code after fixing bugs and adding assertions:

```
module FIFO(fifo_if.DUT fifoif);

    parameter FIFO_DEPTH = 8;
    localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

     always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
        if (!fifoif.rst_n) begin    //resetting all values to 0
            wr_ptr <= 0;
            fifoif.wr_ack <= 0;
            fifoif.overflow <= 0;
        end
        else if (fifoif.wr_en && count < FIFO_DEPTH) begin   //modified
the condition to check if the FIFO is fifoif.full or not
            mem[wr_ptr] <= fifoif.data_in;
            fifoif.wr_ack <= 1;
            wr_ptr <= wr_ptr + 1;
            fifoif.overflow <= 0;
        end
        else begin
            fifoif.wr_ack <= 0;
            if (fifoif.full && fifoif.wr_en)
                fifoif.overflow <= 1;
            else
                fifoif.overflow <= 0;
        end
    end

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

always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
    if (!fifoif.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({fifoif.wr_en, fifoif.rd_en} == 2'b10) && count <
FIFO_DEPTH)
            count <= count + 1;
        else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b01) && count != 0)
            count <= count - 1;
            else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && count ==
FIFO_DEPTH)  //read only
            count <= count - 1;
            else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && count ==
0)  //write only
            count <= count + 1;
    end
end

assign fifoif.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifoif.empty = (count == 0)? 1 : 0;
assign fifoif.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;  // Trigger
when one slot is left
assign fifoif.almostempty = (count == 1)? 1 : 0;


// Assertions (guarded with `ifdef SIM)
`ifdef SIM
    initial begin
```

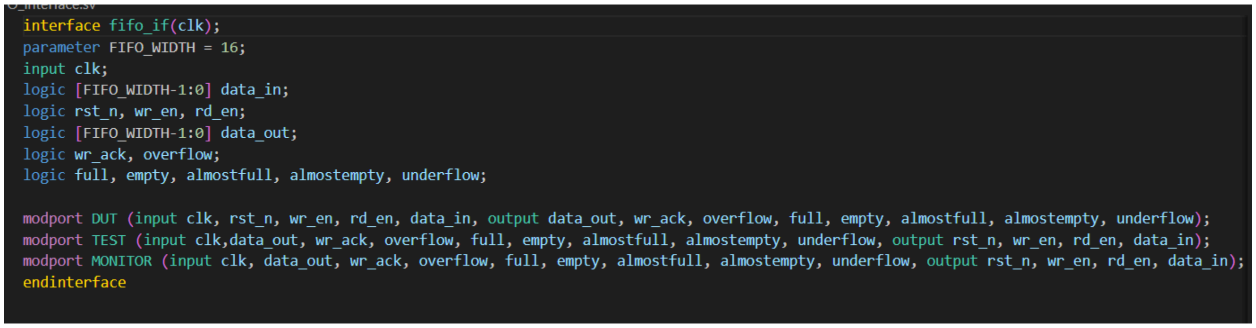
```
        // Assert full flag
        assert property (@(posedge fifoif.clk)
            disable iff (!fifoif.rst_n)
            fifoif.full == (count == FIFO_DEPTH)
        ) else $fatal("FIFO is incorrectly reporting full status!");

        // Assert empty flag
        assert property (@(posedge fifoif.clk)
            disable iff (!fifoif.rst_n)
            fifoif.empty == (count == 0)
        ) else $fatal("FIFO is incorrectly reporting empty status!");

        // Assert almost full flag
        assert property (@(posedge fifoif.clk)
            disable iff (!fifoif.rst_n)
            fifoif.almostfull == (count == FIFO_DEPTH - 1)
        ) else $fatal("FIFO is incorrectly reporting almost full
status!");

        // Assert almost empty flag
        assert property (@(posedge fifoif.clk)
            disable iff (!fifoif.rst_n)
            fifoif.almostempty == (count == 1)
        ) else $fatal("FIFO is incorrectly reporting almost empty
status!");

        // Assert underflow flag
        assert property (@(posedge fifoif.clk)
            disable iff (!fifoif.rst_n)
            fifoif.underflow == (fifoif.empty && fifoif.rd_en)
        ) else $fatal("FIFO is incorrectly reporting underflow status!");
    end
`endif

endmodule
```

Coverage file:



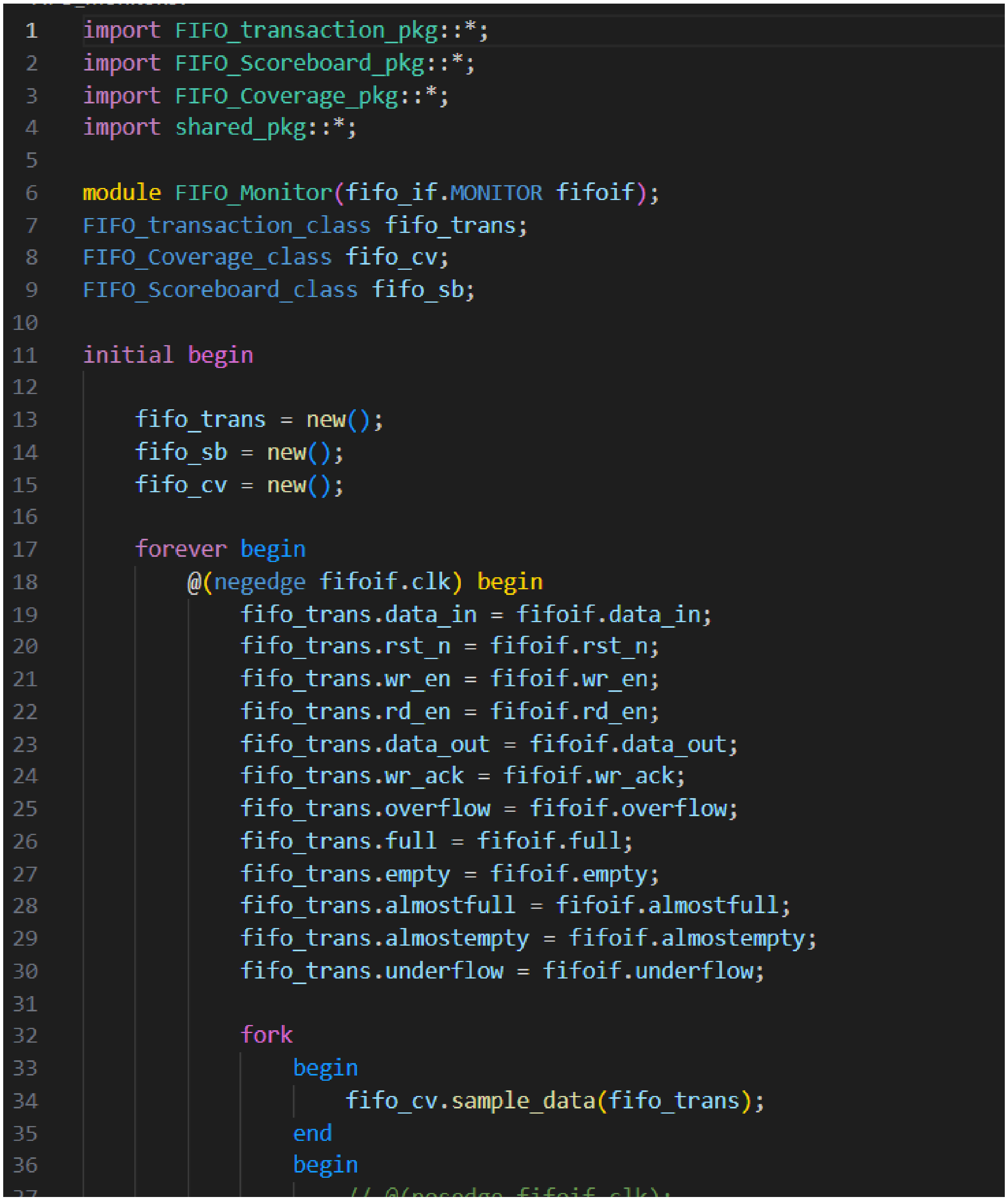
```systemverilog
package FIFO_Coverage_pkg;
import FIFO_transaction_pkg::*;
class FIFO_Coverage_class;
  FIFO_transaction_class F_cvg_txn;

covergroup cvr_gp;
  // Cross coverage between wr_en, rd_en and all control signals
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.wr_ack;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.overflow;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.full;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.empty;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostfull;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostempty;
      cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.underflow;
      endgroup

  function new();
        cvr_gp = new();
    endfunction

function void sample_data(FIFO_transaction_class F_txn);

  F_cvg_txn = F_txn;
  cvr_gp.sample();
endfunction

endclass
endpackage
```

Interface:

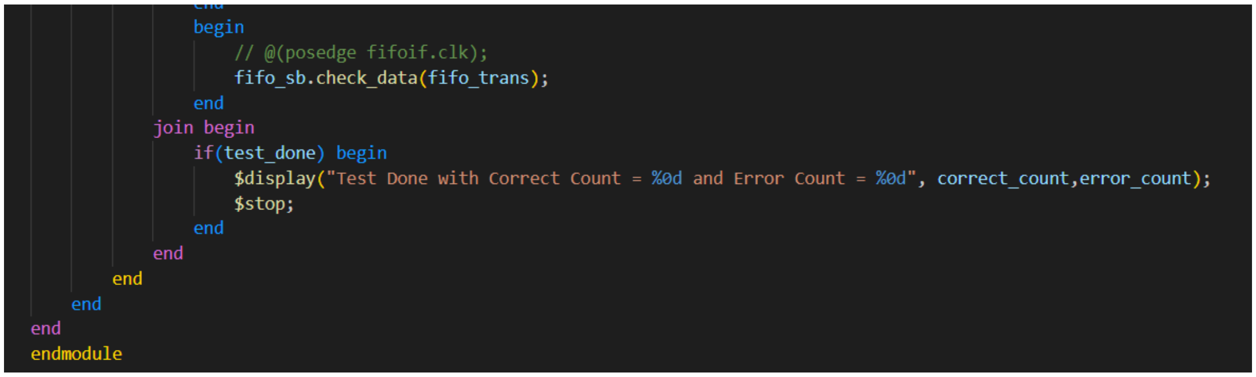


```systemverilog
interface fifo_if(clk);
parameter FIFO_WIDTH = 16;
input clk;
logic [FIFO_WIDTH-1:0] data_in;
logic rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0] data_out;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;

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

Monitor file:

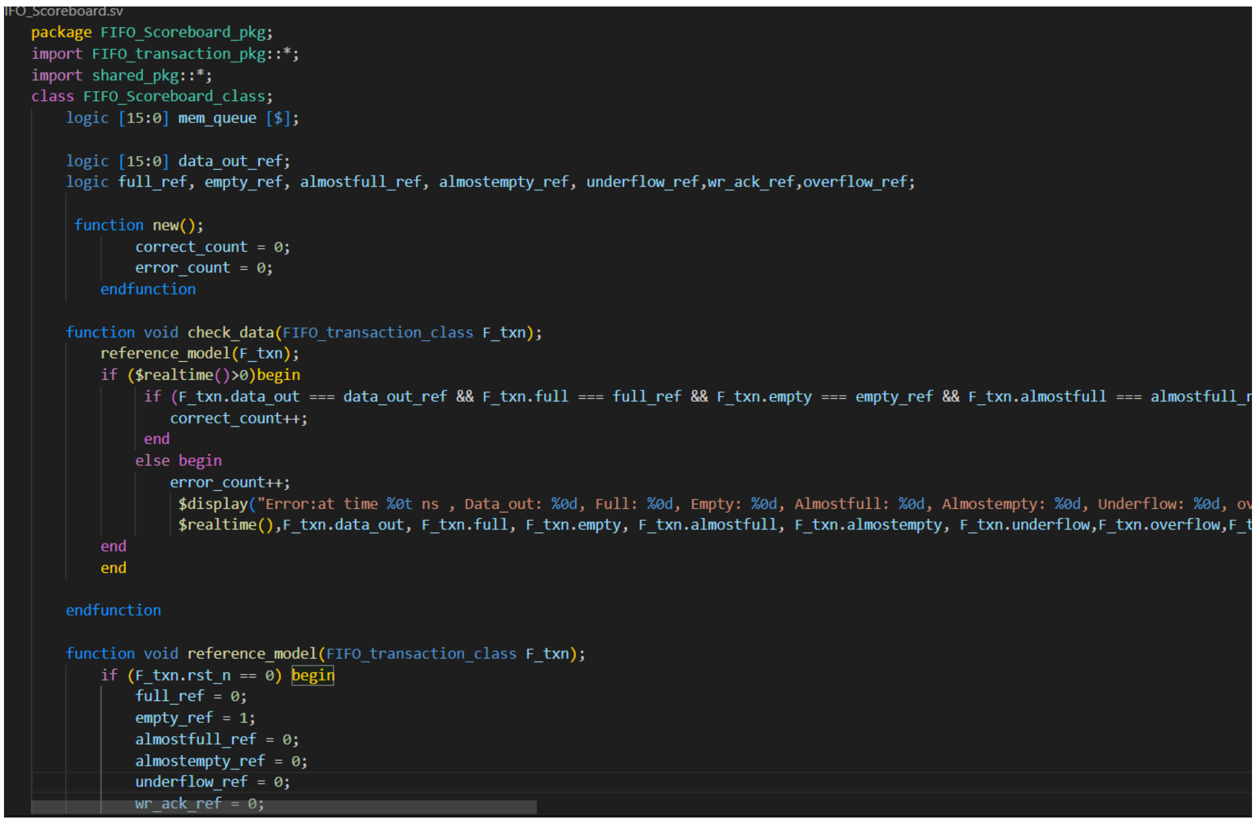


```systemverilog
import FIFO_transaction_pkg::*;
import FIFO_Scoreboard_pkg::*;
import FIFO_Coverage_pkg::*;
import shared_pkg::*;

module FIFO_Monitor(fifo_if.MONITOR fifoif);
FIFO_transaction_class fifo_trans;
FIFO_Coverage_class fifo_cv;
FIFO_Scoreboard_class fifo_sb;

initial begin

    fifo_trans = new();
    fifo_sb = new();
    fifo_cv = new();

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

            fork
                begin
                    fifo_cv.sample_data(fifo_trans);
                end
                begin
```



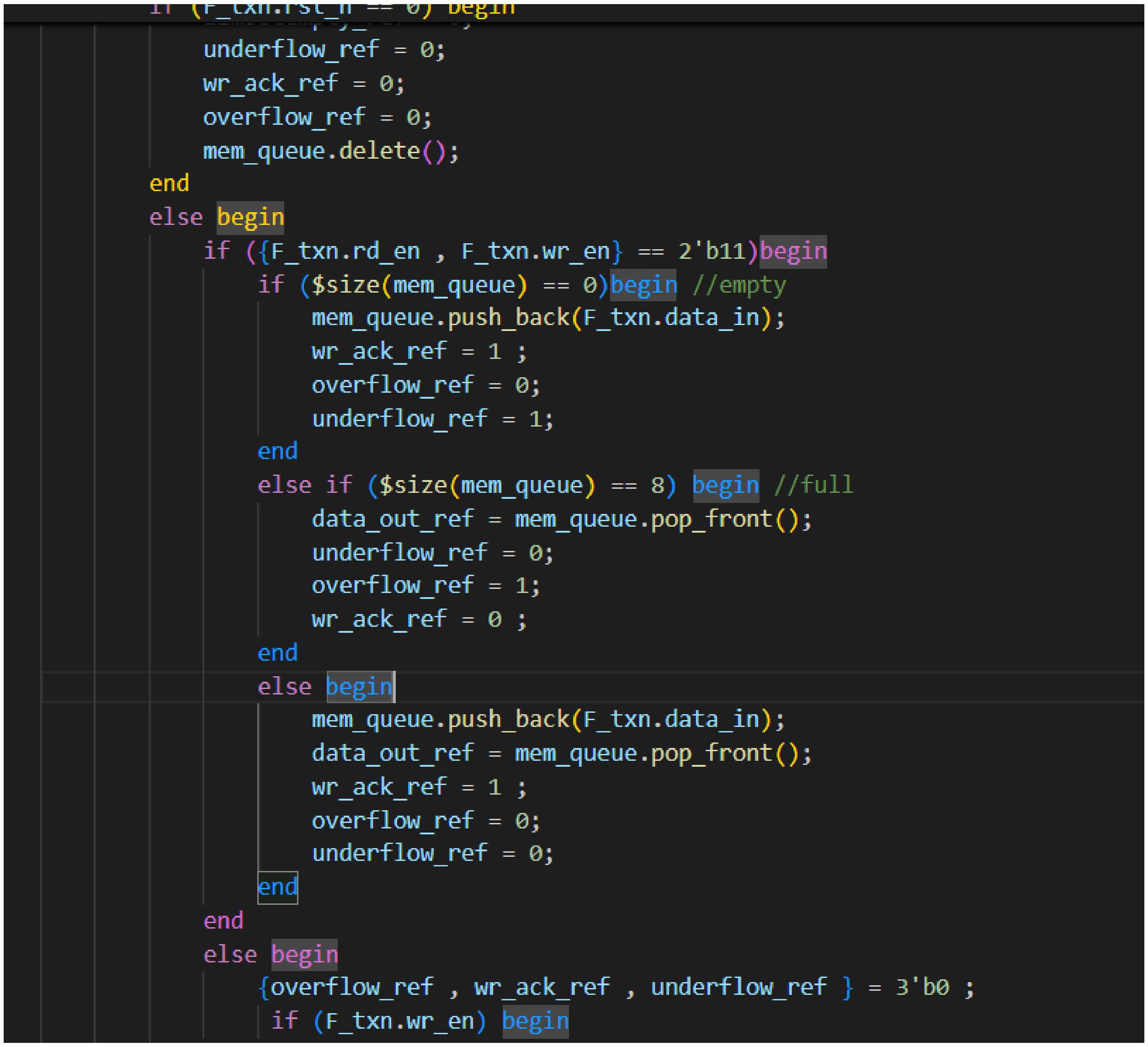
```
                begin
                    // @(posedge fifoif.clk);
                    fifo_sb.check_data(fifo_trans);
                end
            join begin
                if(test_done) begin
                    $display("Test Done with Correct Count = %0d and Error Count = %0d", correct_count,error_count);
                    $stop;
                end
            end
        end
    end
end
endmodule
```

Scoreboard:

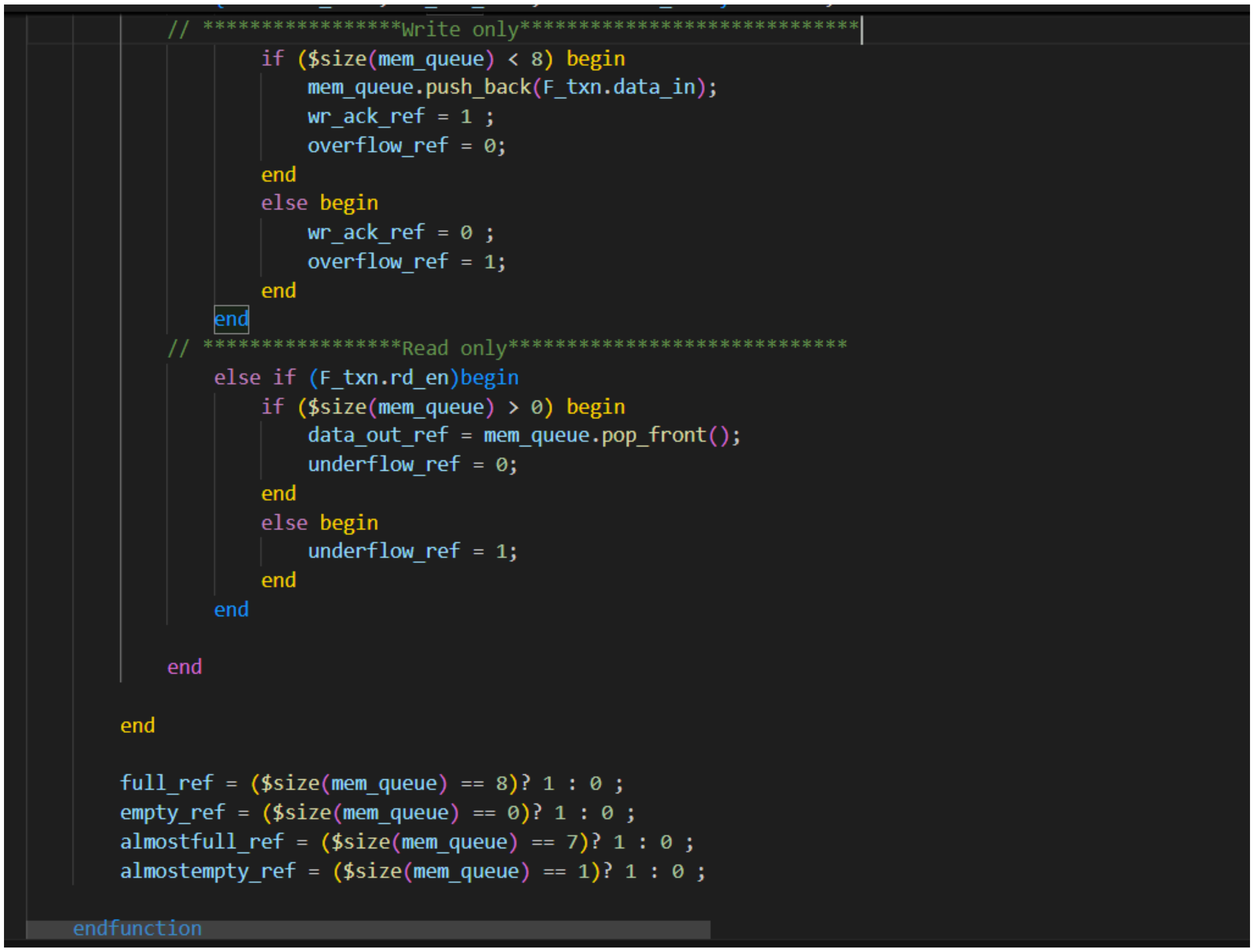


IFO_Scoreboard.sv

```
package FIFO_Scoreboard_pkg;
import FIFO_transaction_pkg::*;
import shared_pkg::*;
class FIFO_Scoreboard_class;
    logic [15:0] mem_queue [$];

    logic [15:0] data_out_ref;
    logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref,wr_ack_ref,overflow_ref;

     function new();
            correct_count = 0;
            error_count = 0;
        endfunction

    function void check_data(FIFO_transaction_class F_txn);
        reference_model(F_txn);
        if ($realtime()>0)begin
             if (F_txn.data_out === data_out_ref && F_txn.full === full_ref && F_txn.empty === empty_ref && F_txn.almostfull === almostfull_r
                correct_count++;
             end
            else begin
                error_count++;
                 $display("Error:at time %0t ns , Data_out: %0d, Full: %0d, Empty: %0d, Almostfull: %0d, Almostempty: %0d, Underflow: %0d, ov
                 $realtime(),F_txn.data_out, F_txn.full, F_txn.empty, F_txn.almostfull, F_txn.almostempty, F_txn.underflow,F_txn.overflow,F_t
        end
        end

    endfunction

    function void reference_model(FIFO_transaction_class F_txn);
        if (F_txn.rst_n == 0) begin
            full_ref = 0;
            empty_ref = 1;
            almostfull_ref = 0;
            almostempty_ref = 0;
            underflow_ref = 0;
            wr_ack_ref = 0;
```

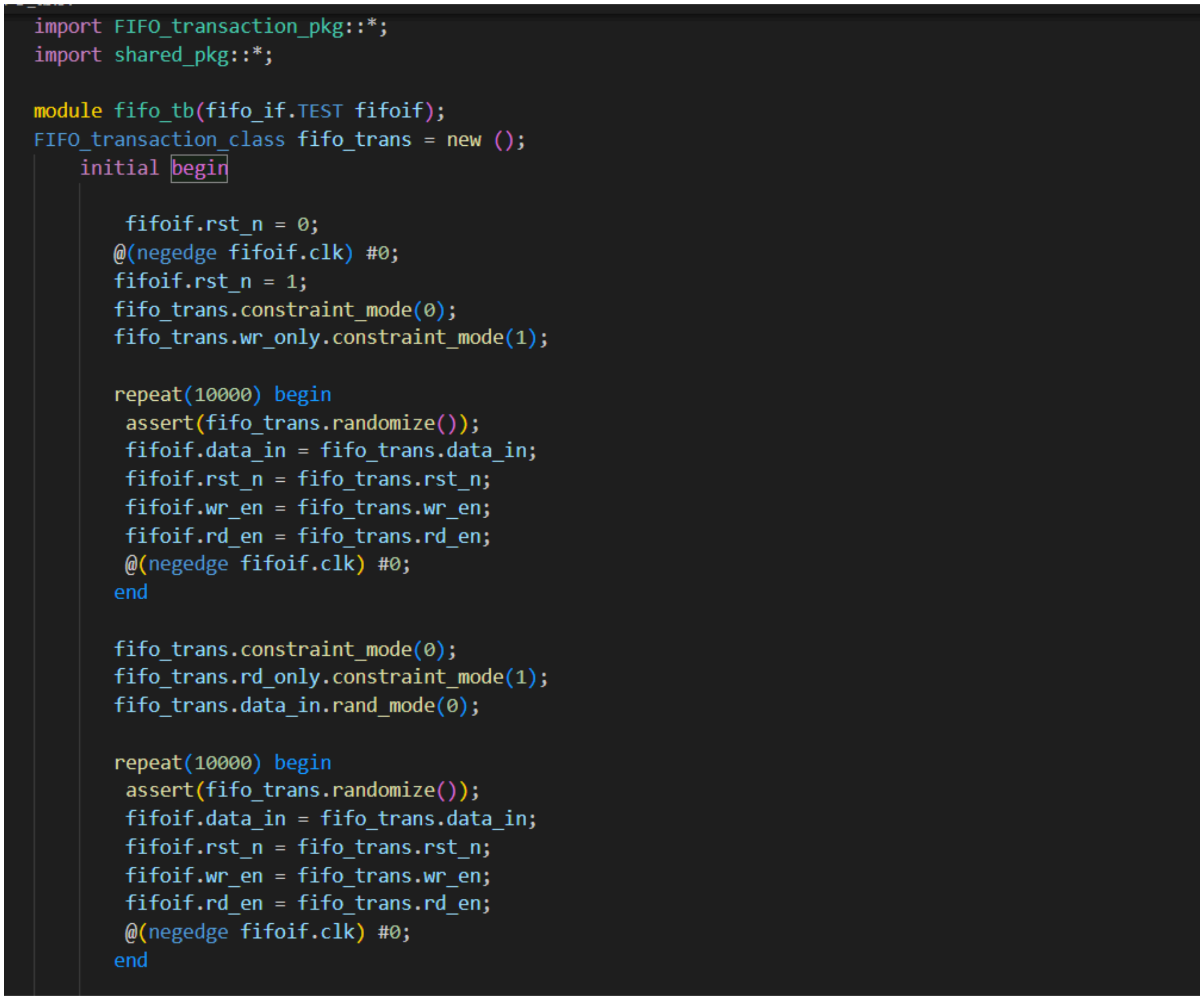


```
if (F_txn.rst_n == 0) begin
    underflow_ref = 0;
    wr_ack_ref = 0;
    overflow_ref = 0;
    mem_queue.delete();
end
else begin
    if ({F_txn.rd_en , F_txn.wr_en} == 2'b11)begin
        if ($size(mem_queue) == 0)begin //empty
            mem_queue.push_back(F_txn.data_in);
            wr_ack_ref = 1 ;
            overflow_ref = 0;
            underflow_ref = 1;
        end
        else if ($size(mem_queue) == 8) begin //full
            data_out_ref = mem_queue.pop_front();
            underflow_ref = 0;
            overflow_ref = 1;
            wr_ack_ref = 0 ;
        end
        else begin
            mem_queue.push_back(F_txn.data_in);
            data_out_ref = mem_queue.pop_front();
            wr_ack_ref = 1 ;
            overflow_ref = 0;
            underflow_ref = 0;
        end
    end
    else begin
        {overflow_ref , wr_ack_ref , underflow_ref } = 3'b0 ;
         if (F_txn.wr_en) begin
```

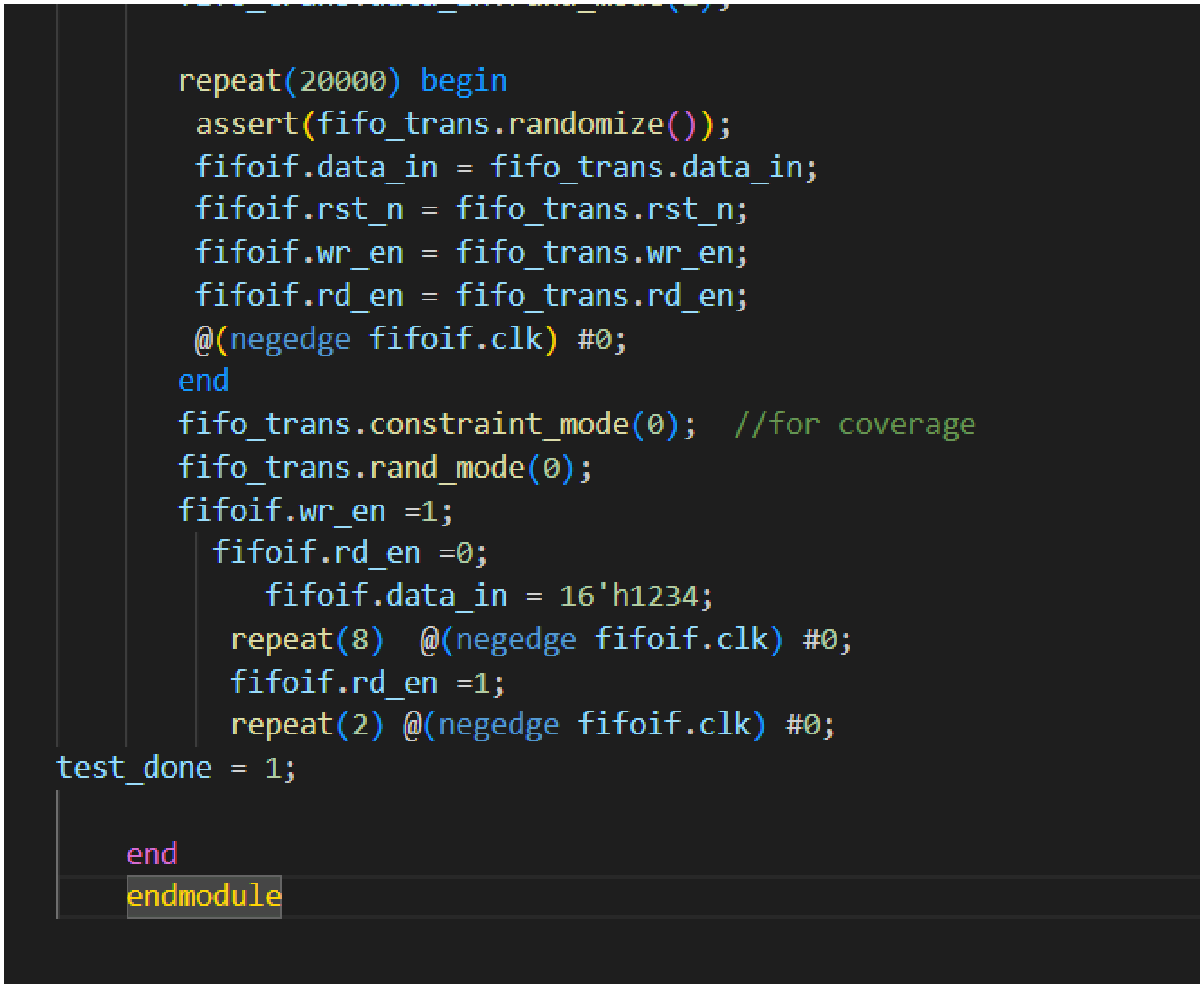


```
        // *****************Write only****************************
                if ($size(mem_queue) < 8) begin
                    mem_queue.push_back(F_txn.data_in);
                    wr_ack_ref = 1 ;
                    overflow_ref = 0;
                end
                else begin
                    wr_ack_ref = 0 ;
                    overflow_ref = 1;
                end
            end
        // *****************Read only****************************
            else if (F_txn.rd_en)begin
                if ($size(mem_queue) > 0) begin
                    data_out_ref = mem_queue.pop_front();
                    underflow_ref = 0;
                end
                else begin
                    underflow_ref = 1;
                end
            end

        end

    end

    full_ref = ($size(mem_queue) == 8)? 1 : 0 ;
    empty_ref = ($size(mem_queue) == 0)? 1 : 0 ;
    almostfull_ref = ($size(mem_queue) == 7)? 1 : 0 ;
    almostempty_ref = ($size(mem_queue) == 1)? 1 : 0 ;

endfunction
```

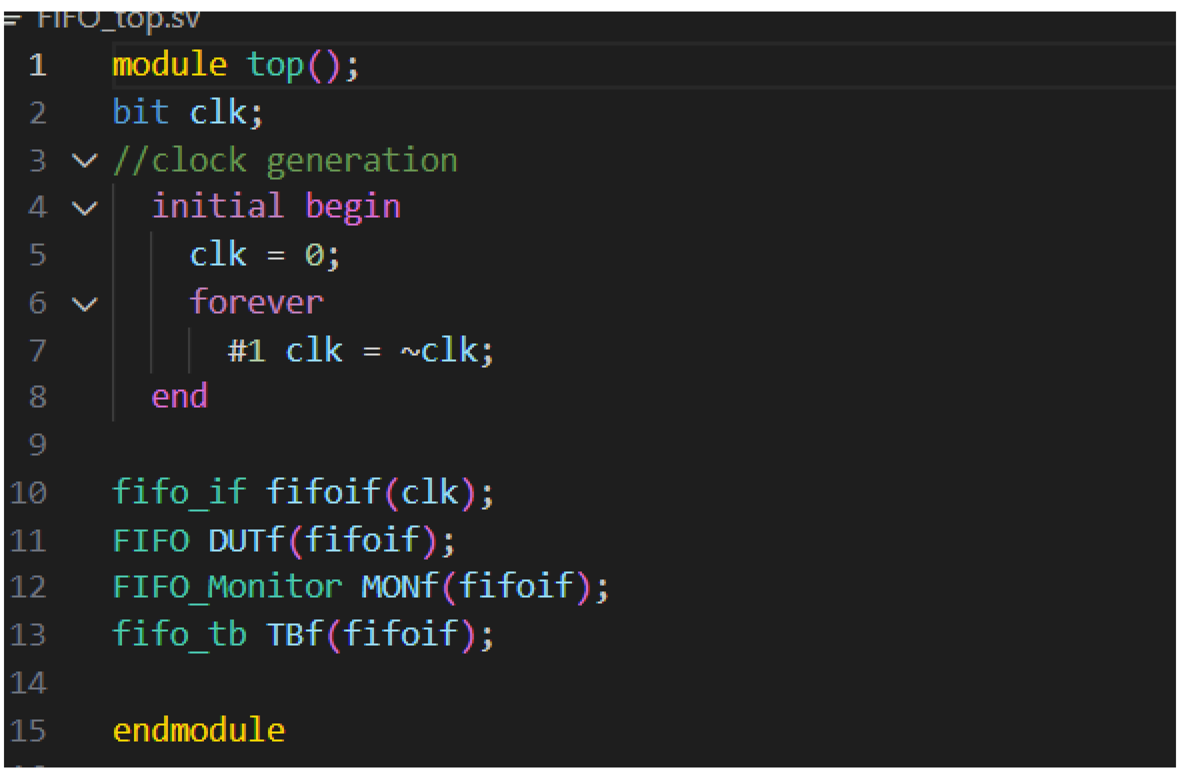
Testbench:



```
import FIFO_transaction_pkg::*;
import shared_pkg::*;

module fifo_tb(fifo_if.TEST fifoif);
FIFO_transaction_class fifo_trans = new ();
    initial begin

        fifoif.rst_n = 0;
       @(negedge fifoif.clk) #0;
       fifoif.rst_n = 1;
       fifo_trans.constraint_mode(0);
       fifo_trans.wr_only.constraint_mode(1);

       repeat(10000) begin
        assert(fifo_trans.randomize());
        fifoif.data_in = fifo_trans.data_in;
        fifoif.rst_n = fifo_trans.rst_n;
        fifoif.wr_en = fifo_trans.wr_en;
        fifoif.rd_en = fifo_trans.rd_en;
        @(negedge fifoif.clk) #0;
       end

       fifo_trans.constraint_mode(0);
       fifo_trans.rd_only.constraint_mode(1);
       fifo_trans.data_in.rand_mode(0);

       repeat(10000) begin
        assert(fifo_trans.randomize());
        fifoif.data_in = fifo_trans.data_in;
        fifoif.rst_n = fifo_trans.rst_n;
        fifoif.wr_en = fifo_trans.wr_en;
        fifoif.rd_en = fifo_trans.rd_en;
        @(negedge fifoif.clk) #0;
       end
```


```
    repeat(20000) begin
     assert(fifo_trans.randomize());
     fifoif.data_in = fifo_trans.data_in;
     fifoif.rst_n = fifo_trans.rst_n;
     fifoif.wr_en = fifo_trans.wr_en;
     fifoif.rd_en = fifo_trans.rd_en;
     @(negedge fifoif.clk) #0;
    end
    fifo_trans.constraint_mode(0);  //for coverage
    fifo_trans.rand_mode(0);
    fifoif.wr_en =1;
      fifoif.rd_en =0;
         fifoif.data_in = 16'h1234;
       repeat(8)  @(negedge fifoif.clk) #0;
       fifoif.rd_en =1;
       repeat(2) @(negedge fifoif.clk) #0;
test_done = 1;

    end
    endmodule
```

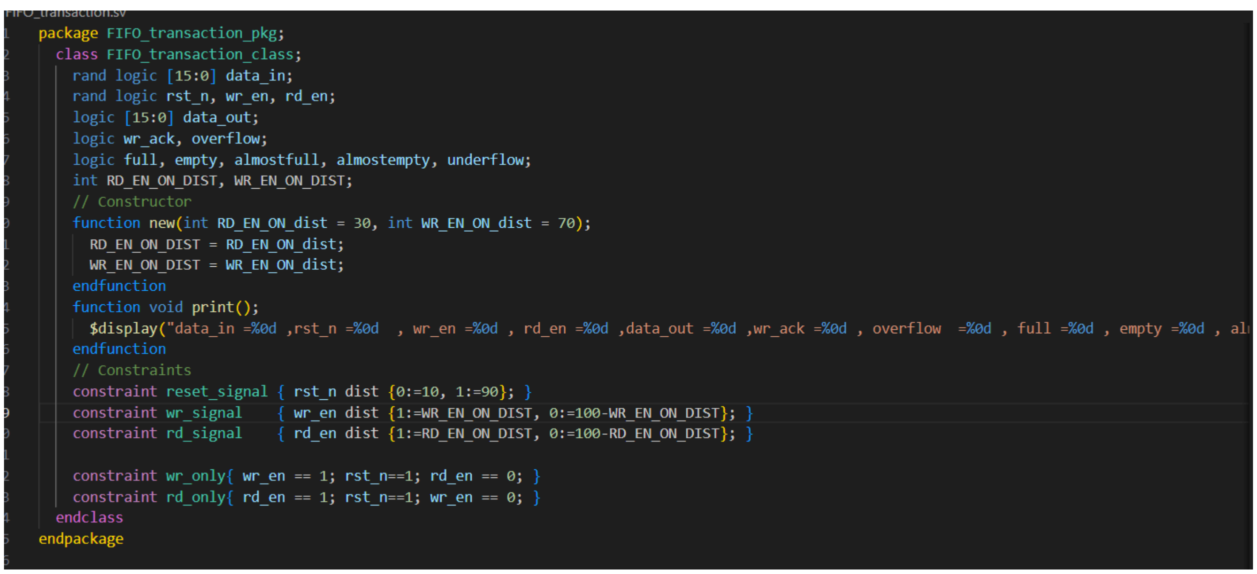
Top module:



FIFO_top.sv

```
module top();
bit clk;
//clock generation
  initial begin
    clk = 0;
    forever
      #1 clk = ~clk;
  end

fifo_if fifoif(clk);
FIFO DUTf(fifoif);
FIFO_Monitor MONf(fifoif);
fifo_tb TBf(fifoif);

endmodule
```

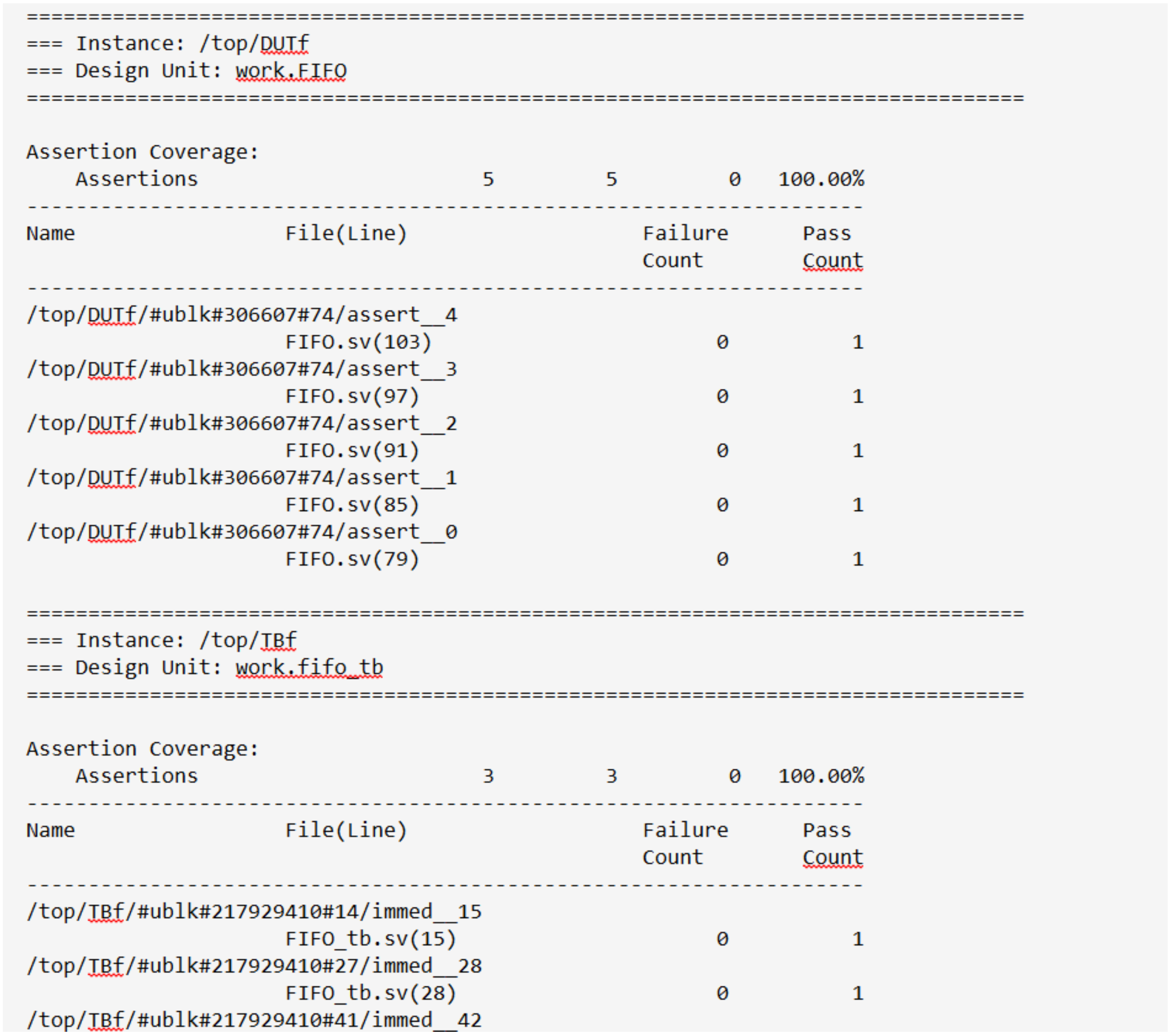
Transaction:



FIFO_transaction.sv

```
package FIFO_transaction_pkg;
  class FIFO_transaction_class;
    rand logic [15:0] data_in;
    rand logic rst_n, wr_en, rd_en;
    logic [15:0] data_out;
    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;
    int RD_EN_ON_DIST, WR_EN_ON_DIST;
    // Constructor
    function new(int RD_EN_ON_dist = 30, int WR_EN_ON_dist = 70);
      RD_EN_ON_DIST = RD_EN_ON_dist;
      WR_EN_ON_DIST = WR_EN_ON_dist;
    endfunction
    function void print();
      $display("data_in =%0d ,rst_n =%0d  , wr_en =%0d , rd_en =%0d ,data_out =%0d ,wr_ack =%0d , overflow  =%0d , full =%0d , empty =%0d , al
    endfunction
    // Constraints
    constraint reset_signal { rst_n dist {0:=10, 1:=90}; }
    constraint wr_signal    { wr_en dist {1:=WR_EN_ON_DIST, 0:=100-WR_EN_ON_DIST}; }
    constraint rd_signal    { rd_en dist {1:=RD_EN_ON_DIST, 0:=100-RD_EN_ON_DIST}; }

    constraint wr_only{ wr_en == 1; rst_n==1; rd_en == 0; }
    constraint rd_only{ rd_en == 1; rst_n==1; wr_en == 0; }
  endclass
endpackage
```

Assertion Coverage Report:

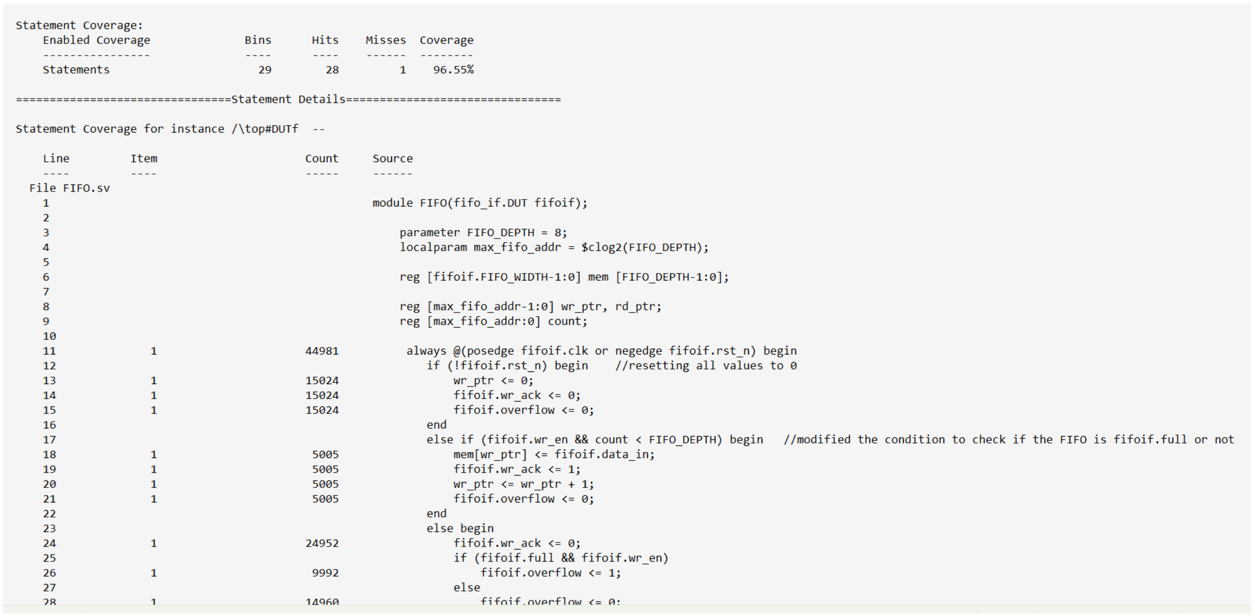


```
=================================================================================
=== Instance: /top/DUTf
=== Design Unit: work.FIFO
=================================================================================

Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUTf/#ublk#306607#74/assert__4
                     FIFO.sv(103)                       0          1
/top/DUTf/#ublk#306607#74/assert__3
                     FIFO.sv(97)                        0          1
/top/DUTf/#ublk#306607#74/assert__2
                     FIFO.sv(91)                        0          1
/top/DUTf/#ublk#306607#74/assert__1
                     FIFO.sv(85)                        0          1
/top/DUTf/#ublk#306607#74/assert__0
                     FIFO.sv(79)                        0          1

=================================================================================
=== Instance: /top/TBf
=== Design Unit: work.fifo_tb
=================================================================================

Assertion Coverage:
    Assertions                       3         3         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/TBf/#ublk#217929410#14/immed__15
                     FIFO_tb.sv(15)                     0          1
/top/TBf/#ublk#217929410#27/immed__28
                     FIFO_tb.sv(28)                     0          1
/top/TBf/#ublk#217929410#41/immed__42
```

```
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUTf/#ublk#306607#74/assert__4
                     FIFO.sv(103)                       0           1
/top/DUTf/#ublk#306607#74/assert__3
                     FIFO.sv(97)                        0           1
/top/DUTf/#ublk#306607#74/assert__2
                     FIFO.sv(91)                        0           1
/top/DUTf/#ublk#306607#74/assert__1
                     FIFO.sv(85)                        0           1
/top/DUTf/#ublk#306607#74/assert__0
                     FIFO.sv(79)                        0           1
/top/TBf/#ublk#217929410#14/immed__15
                     FIFO_tb.sv(15)                     0           1
/top/TBf/#ublk#217929410#27/immed__28
                     FIFO_tb.sv(28)                     0           1
/top/TBf/#ublk#217929410#41/immed__42
                     FIFO_tb.sv(42)                     0           1

Total Coverage By Instance (filtered view): 97.32%
```

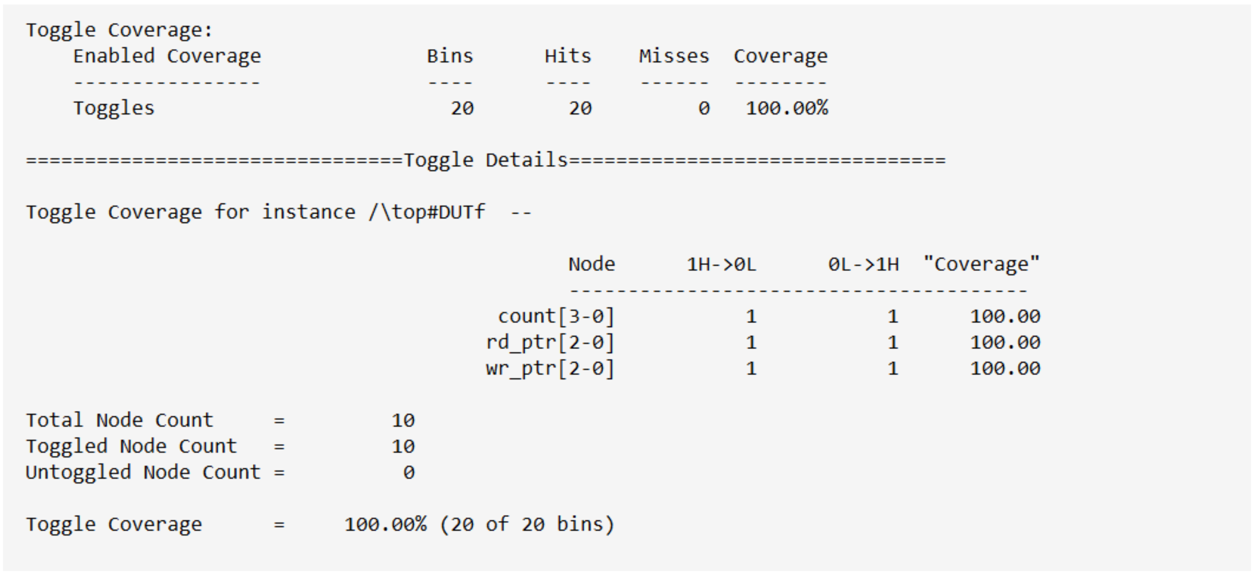
Code and Functional Coverage:

```
================================Condition Details================================

Condition Coverage for instance /\top#DUTf  --

  File FIFO.sv
----------------Focused Condition View-------------------
Line       17 Item    1  (fifoif.wr_en && (count < 8))
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
  fifoif.wr_en         Y
   (count < 8)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  fifoif.wr_en_0        -
  Row   2:          1  fifoif.wr_en_1        (count < 8)
  Row   3:          1  (count < 8)_0         fifoif.wr_en
  Row   4:          1  (count < 8)_1         fifoif.wr_en

----------------Focused Condition View-------------------
Line       25 Item    1  (fifoif.full && fifoif.wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
   fifoif.full         Y
  fifoif.wr_en         Y
```

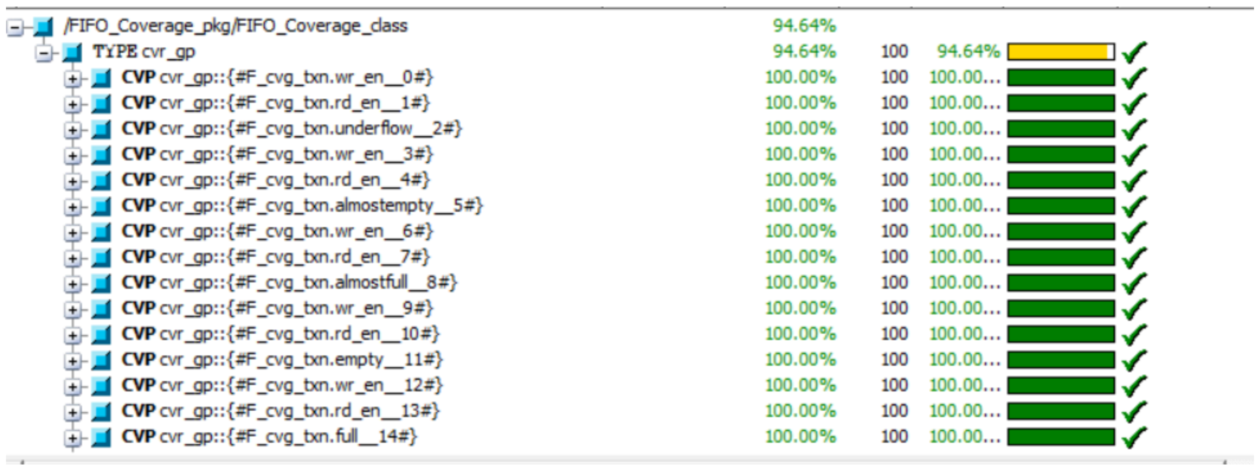


```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      29        28         1    96.55%

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

Statement Coverage for instance /\top#DUTf  --

    Line         Item                      Count     Source 
    ----         ----                      -----     ------ 
  File FIFO.sv
    1                                                module FIFO(fifo_if.DUT fifoif);
    2                                                
    3                                                    parameter FIFO_DEPTH = 8;
    4                                                    localparam max_fifo_addr = $clog2(FIFO_DEPTH);
    5                                                
    6                                                    reg [fifoif.FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
    7                                                
    8                                                    reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    9                                                    reg [max_fifo_addr:0] count;
    10                                               
    11              1                      44981          always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
    12                                                       if (!fifoif.rst_n) begin    //resetting all values to 0
    13              1                      15024                 wr_ptr <= 0;
    14              1                      15024                 fifoif.wr_ack <= 0;
    15              1                      15024                 fifoif.overflow <= 0;
    16                                                       end
    17                                                       else if (fifoif.wr_en && count < FIFO_DEPTH) begin   //modified the condition to check if the FIFO is fifoif.full or not
    18              1                       5005                 mem[wr_ptr] <= fifoif.data_in;
    19              1                       5005                 fifoif.wr_ack <= 1;
    20              1                       5005                 wr_ptr <= wr_ptr + 1;
    21              1                       5005                 fifoif.overflow <= 0;
    22                                                       end
    23                                                       else begin
    24              1                      24952                 fifoif.wr_ack <= 0;
    25                                                           if (fifoif.full && fifoif.wr_en)
    26              1                       9992                     fifoif.overflow <= 1;
    27                                                           else
    28              1                      14960                     fifoif.overflow <= 0;
```

Statement coverage can be better by increasing the loops of randomization to reach the condition (rd_en = 1, wr_en =1, and count = FIFO_DEPTH)



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

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

Toggle Coverage for instance /\top#DUTf  --

                                              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

Total Node Count     =         10 
Toggled Node Count   =         10 
Untoggled Node Count =          0 

Toggle Coverage      =     100.00% (20 of 20 bins)
```



```
/FIFO_Coverage_pkg/FIFO_Coverage_class                 94.64%
  TYPE cvr_gp                                          94.64%   100   94.64%   ✓
    CVP cvr_gp::{#F_cvg_txn.wr_en__0#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.rd_en__1#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.underflow__2#}             100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.wr_en__3#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.rd_en__4#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.almostempty__5#}           100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.wr_en__6#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.rd_en__7#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.almostfull__8#}            100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.wr_en__9#}                 100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.rd_en__10#}                100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.empty__11#}                100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.wr_en__12#}                100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.rd_en__13#}                100.00%  100   100.00...  ✓
    CVP cvr_gp::{#F_cvg_txn.full__14#}                 100.00%  100   100.00...  ✓
```

Do file for assertions:

```
vlib work
vlog -f src_files.txt +cover
vlog -work work -vopt -sv -stats=none
+incdir+path_to_assertions_dir +define+SIM FIFO.sv
vsim -voptargs=+acc work.top -cover
run -all
coverage save top.ucdb -du FIFO -onexit
coverage report -detail -assert -cvg -directive
-comments -output Assertion_Fcoverage_reports.txt
{}
quit -sim
```

Do file for Coverage and functional Coverage:

```
vlog -f src_files.txt +cover
vsim -voptargs=+acc work.top -cover
run -all
coverage save top1.ucdb -du FIFO -onexit
quit -sim
vcover report top1.ucdb -details -annotate -all
-output Code_coverage_reports_final.txt
```