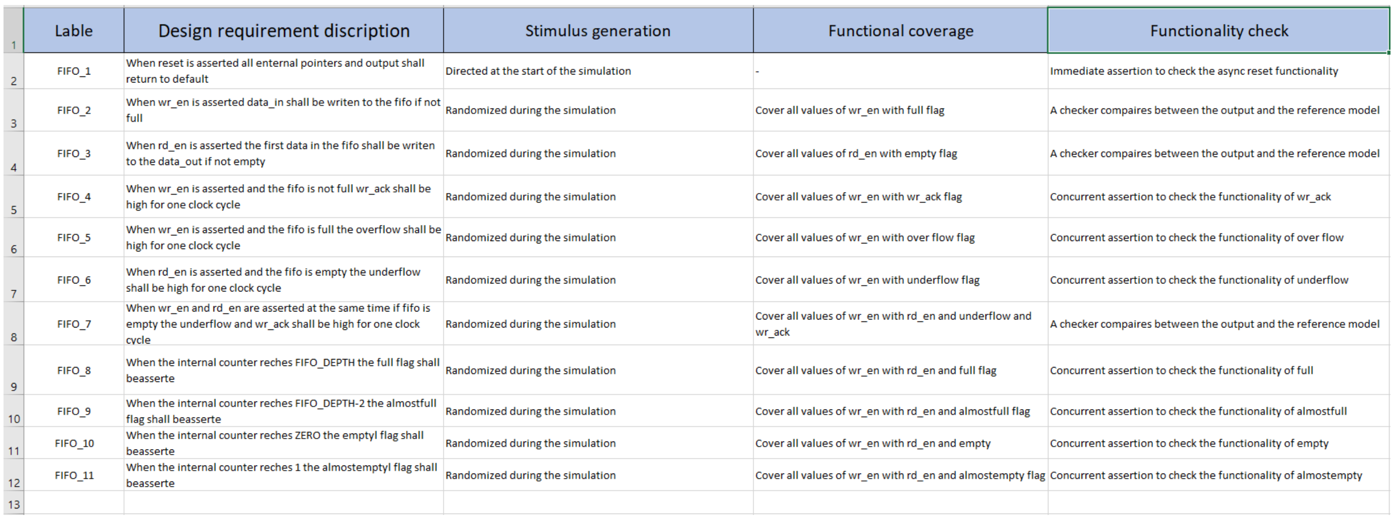
Project 2

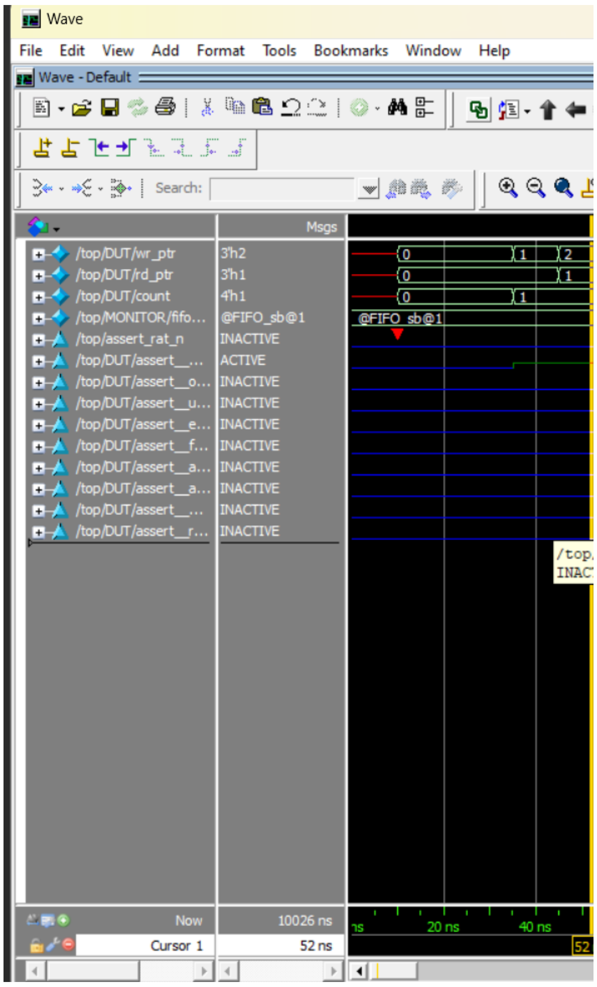
⇨ Verification plan



Lable	Design requirement discription	Stimulus generation	Functional coverage	Functionality check
FIFO_1	When reset is asserted all enternal pointers and output shall return to default	Directed at the start of the simulation	-	Immediate assertion to check the async reset functionality
FIFO_2	When wr_en is asserted data_in shall be writen to the fifo if not full	Randomized during the simulation	Cover all values of wr_en with full flag	A checker compaires between the output and the reference model
FIFO_3	When rd_en is asserted the first data in the fifo shall be writen to the data_out if not empty	Randomized during the simulation	Cover all values of rd_en with empty flag	A checker compaires between the output and the reference model
FIFO_4	When wr_en is asserted and the fifo is not full wr_ack shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with wr_ack flag	Concurrent assertion to check the functionality of wr_ack
FIFO_5	When wr_en is asserted and the fifo is full the overflow shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with over flow flag	Concurrent assertion to check the functionality of over flow
FIFO_6	When rd_en is asserted and the fifo is empty the underflow shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with underflow flag	Concurrent assertion to check the functionality of underflow
FIFO_7	When wr_en and rd_en are asserted at the same time if fifo is empty the underflow and wr_ack shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with rd_en and underflow and wr_ack	A checker compaires between the output and the reference model
FIFO_8	When the internal counter reches FIFO_DEPTH the full flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and full flag	Concurrent assertion to check the functionality of full
FIFO_9	When the internal counter reches FIFO_DEPTH-2 the almostfull flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and almostfull flag	Concurrent assertion to check the functionality of almostfull
FIFO_10	When the internal counter reches ZERO the emptyl flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and empty	Concurrent assertion to check the functionality of empty
FIFO_11	When the internal counter reches 1 the almostemptyl flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and almostempty flag	Concurrent assertion to check the functionality of almostempty

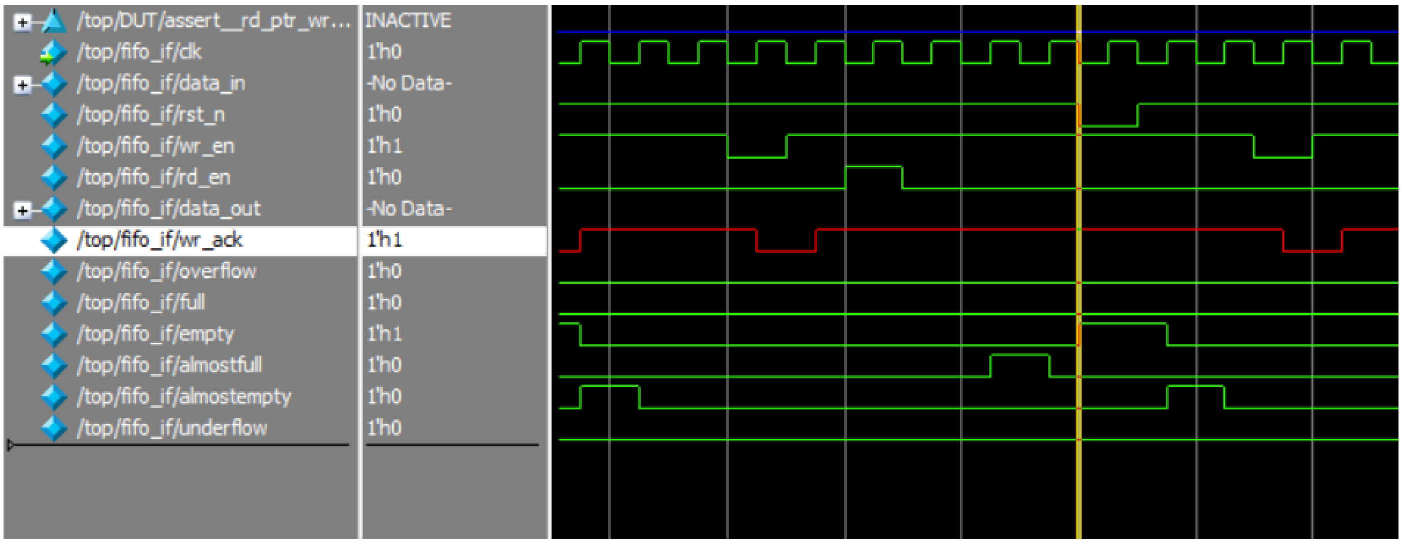
⇨ Bugs detected

- 1- when resetting data_out register doesn’t reset.



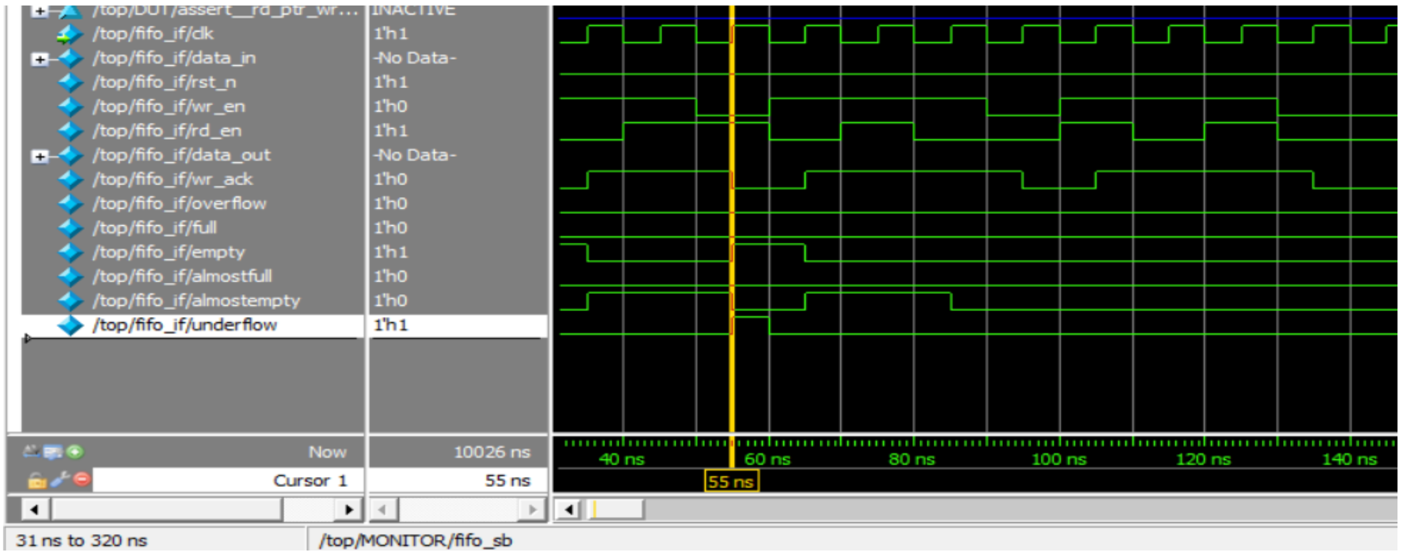
```
/top/DUT/wr_ptr          3'h2
/top/DUT/rd_ptr          3'h1
/top/DUT/count           4'h1
/top/MONITOR/fifo...     @FIFO_sb@1
/top/assert_rat_n        INACTIVE
/top/DUT/assert___...    ACTIVE
/top/DUT/assert__o...    INACTIVE
/top/DUT/assert__u...    INACTIVE
/top/DUT/assert__e...    INACTIVE
/top/DUT/assert__f...    INACTIVE
/top/DUT/assert__a...    INACTIVE
/top/DUT/assert__a...    INACTIVE
/top/DUT/assert___...    INACTIVE
/top/DUT/assert__r...    INACTIVE

Now        10026 ns
Cursor 1   52 ns
```

- 2- wr_ack is not affected by the reset.
- 2.1- and doesn’t go low if it was high and the full flag is asserted before wr_en become low in the next rising edge of the clock.



```
/top/DUT/assert__rd_ptr_wr...  INACTIVE
/top/fifo_if/clk               1'h0
/top/fifo_if/data_in           -No Data-
/top/fifo_if/rst_n             1'h0
/top/fifo_if/wr_en             1'h1
/top/fifo_if/rd_en             1'h0
/top/fifo_if/data_out          -No Data-
/top/fifo_if/wr_ack            1'h1
/top/fifo_if/overflow          1'h0
/top/fifo_if/full              1'h0
/top/fifo_if/empty             1'h1
/top/fifo_if/almostfull        1'h0
/top/fifo_if/almostempty       1'h0
/top/fifo_if/underflow         1'h0
```

- 3- underflow is not sequential.

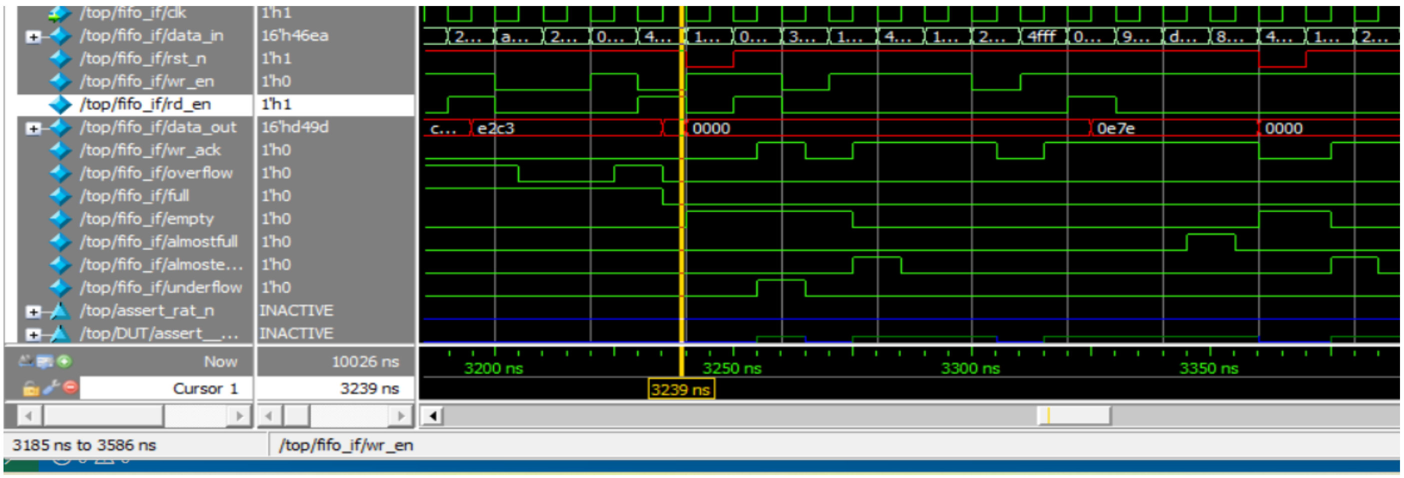


```
/top/DUT/assert__rd_ptr_wr...  INACTIVE
/top/fifo_if/clk               1'h1
/top/fifo_if/data_in           -No Data-
/top/fifo_if/rst_n             1'h1
/top/fifo_if/wr_en             1'h0
/top/fifo_if/rd_en             1'h1
/top/fifo_if/data_out          -No Data-
/top/fifo_if/wr_ack            1'h0
/top/fifo_if/overflow          1'h0
/top/fifo_if/full              1'h0
/top/fifo_if/empty             1'h1
/top/fifo_if/almostfull        1'h0
/top/fifo_if/almostempty       1'h0
/top/fifo_if/underflow         1'h1
Now        10026 ns
Cursor 1   55 ns
31 ns to 320 ns    /top/MONITOR/fifo_sb
```

- 4- overflow shall be low if write operation is accepted.
- 5- if write or read operation is rejected the internal counter is not handled.
- 6- almost full when counter is FIFO_DEPTH-1 not -2

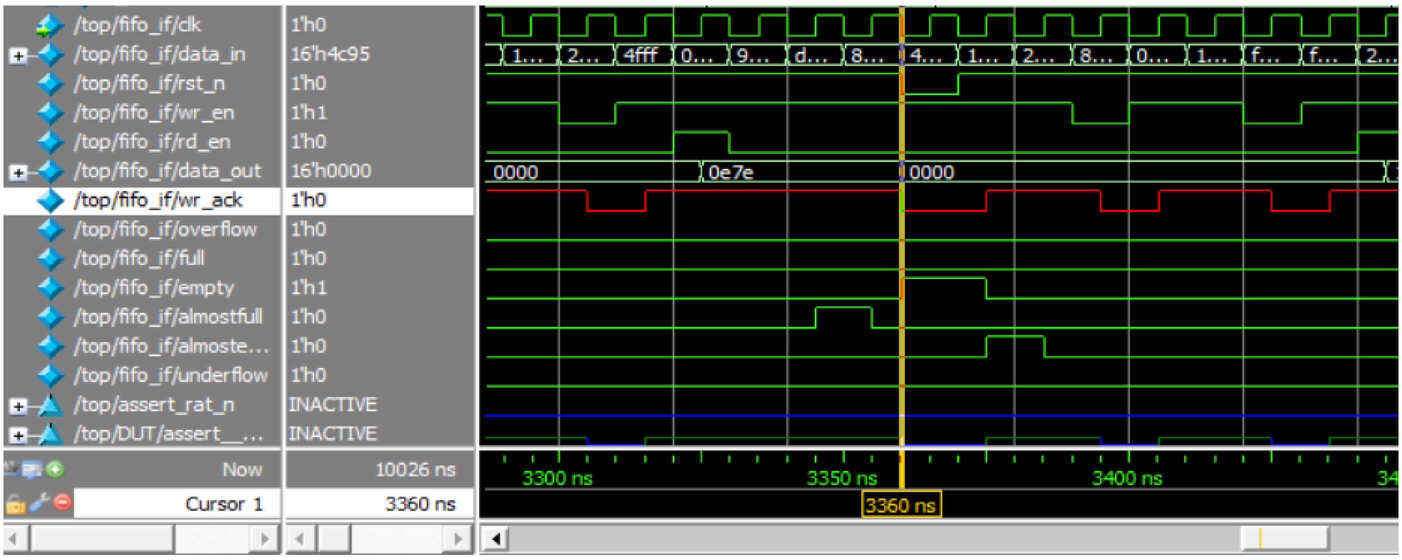
⇨ Fixed bugs

- Bug 1



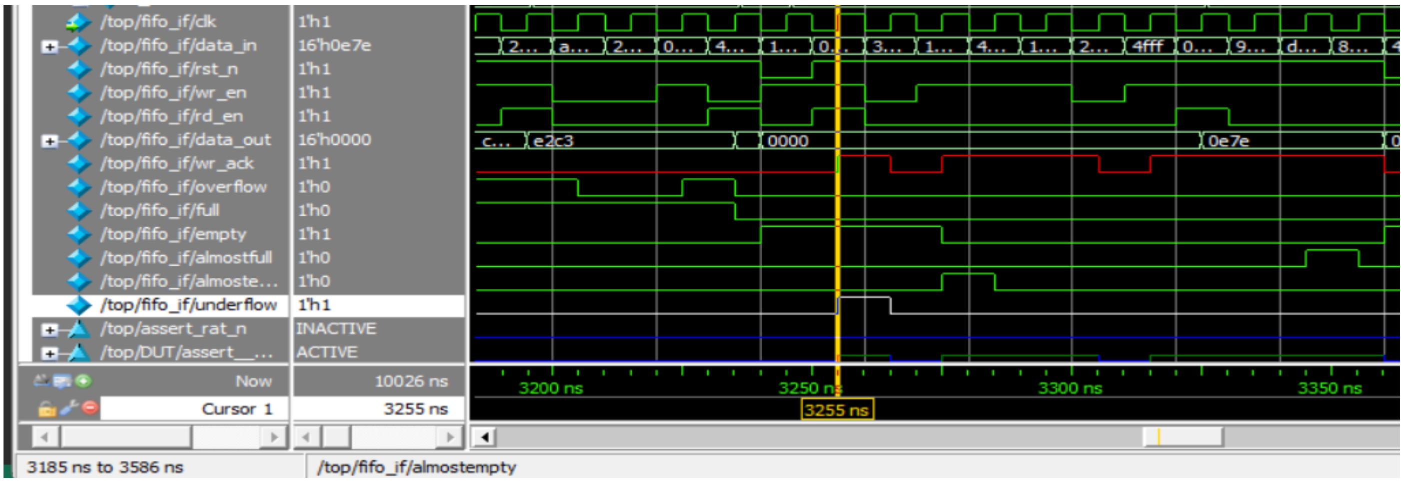
```
/top/fifo_if/clk            1'h1
/top/fifo_if/data_in        16'h46ea
/top/fifo_if/rst_n          1'h1
/top/fifo_if/wr_en          1'h0
/top/fifo_if/rd_en          1'h1
/top/fifo_if/data_out       16'hd49d
/top/fifo_if/wr_ack         1'h0
/top/fifo_if/overflow       1'h0
/top/fifo_if/full           1'h0
/top/fifo_if/empty          1'h0
/top/fifo_if/almostfull     1'h0
/top/fifo_if/almoste...     1'h0
/top/fifo_if/underflow      1'h0
/top/assert_rat_n           INACTIVE
/top/DUT/assert__...        INACTIVE
Now        10026 ns
Cursor 1   3239 ns
3185 ns to 3586 ns    /top/fifo_if/wr_en
```

- Bug 2



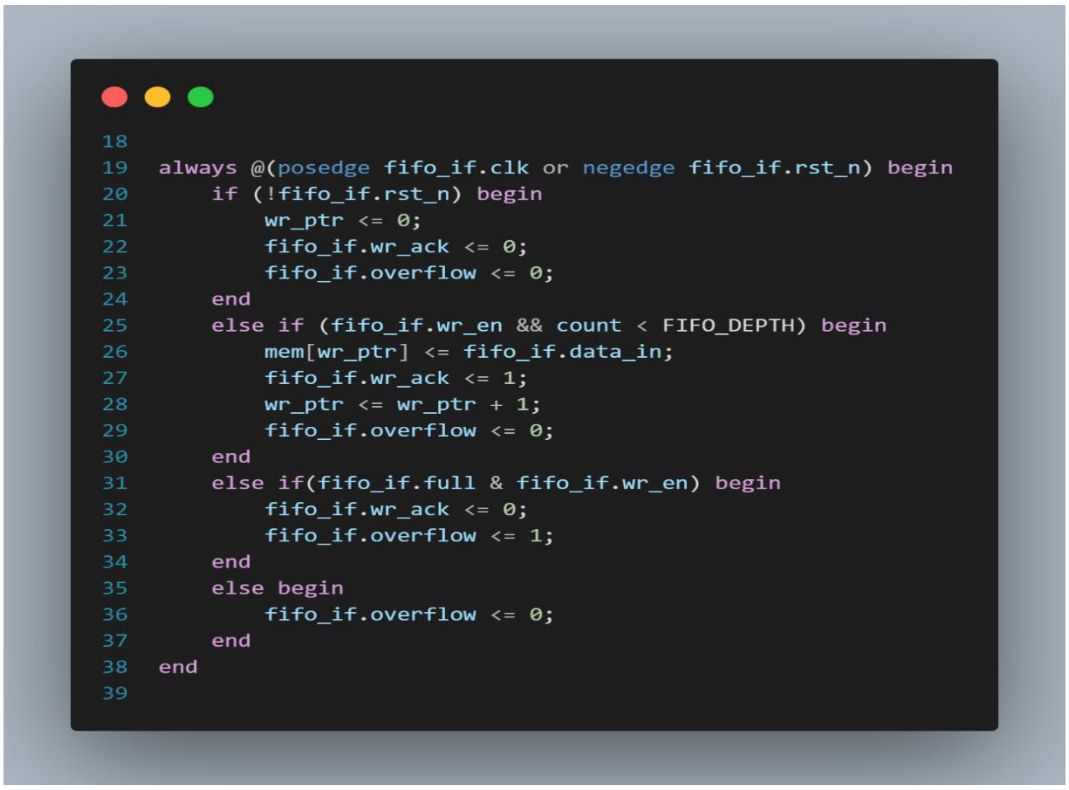
```
/top/fifo_if/clk          1'h0
/top/fifo_if/data_in      16'h4c95
/top/fifo_if/rst_n        1'h0
/top/fifo_if/wr_en        1'h1
/top/fifo_if/rd_en        1'h0
/top/fifo_if/data_out     16'h0000
/top/fifo_if/wr_ack       1'h0
/top/fifo_if/overflow     1'h0
/top/fifo_if/full         1'h0
/top/fifo_if/empty        1'h1
/top/fifo_if/almostfull   1'h0
/top/fifo_if/almoste...   1'h0
/top/fifo_if/underflow    1'h0
/top/assert_rat_n         INACTIVE
/top/DUT/assert_...       INACTIVE
Now                       10026 ns
Cursor 1                  3360 ns
```

- Bug 3



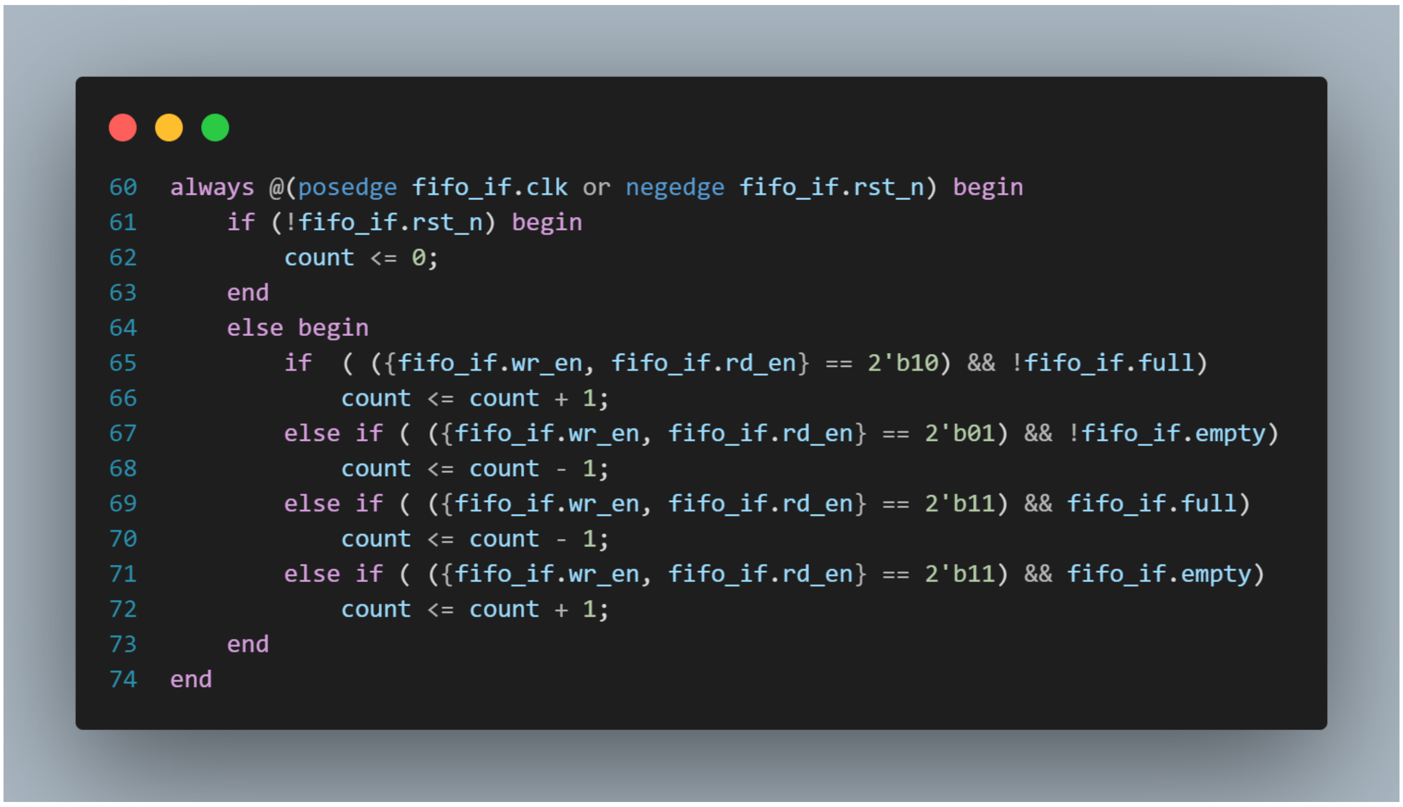
```
/top/fifo_if/clk          1'h1
/top/fifo_if/data_in      16'h0e7e
/top/fifo_if/rst_n        1'h1
/top/fifo_if/wr_en        1'h1
/top/fifo_if/rd_en        1'h1
/top/fifo_if/data_out     16'h0000
/top/fifo_if/wr_ack       1'h1
/top/fifo_if/overflow     1'h0
/top/fifo_if/full         1'h0
/top/fifo_if/empty        1'h1
/top/fifo_if/almostfull   1'h0
/top/fifo_if/almoste...   1'h0
/top/fifo_if/underflow    1'h1
/top/assert_rat_n         INACTIVE
/top/DUT/assert__...      ACTIVE
Now                       10026 ns
Cursor 1                  3255 ns
3185 ns to 3586 ns        /top/fifo_if/almostempty
```

- Bug 4



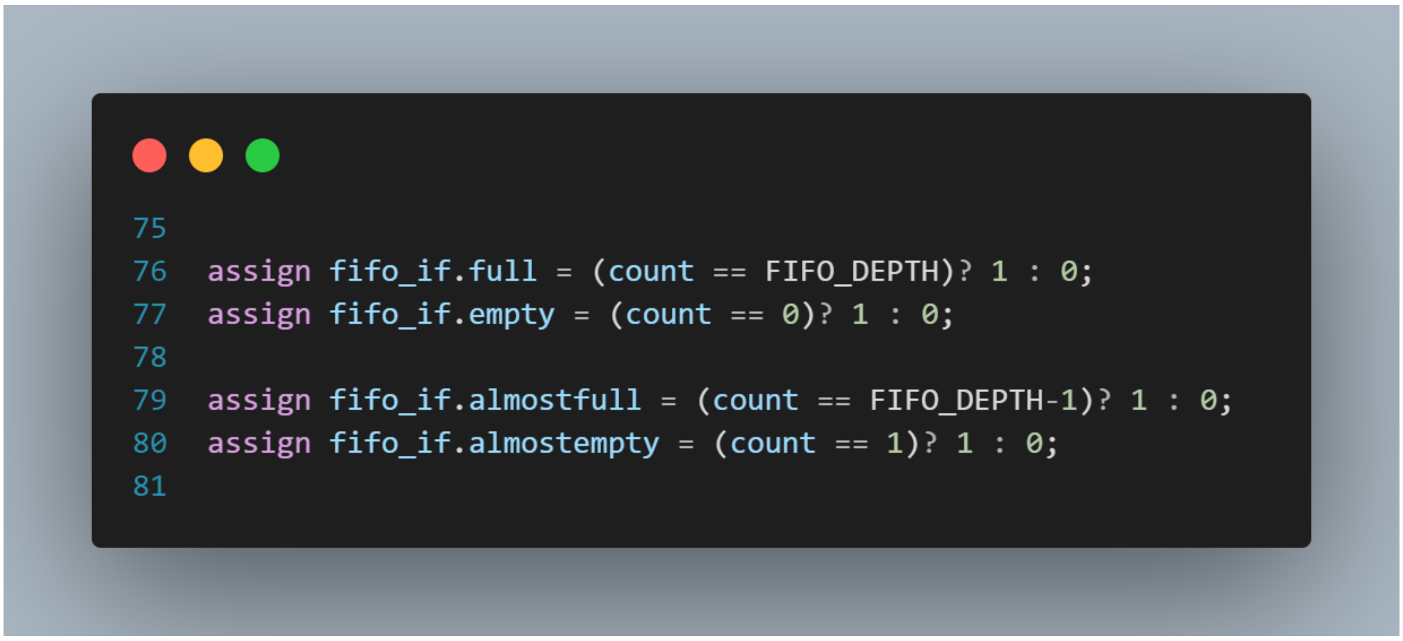
```

always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        wr_ptr <= 0;
        fifo_if.wr_ack <= 0;
        fifo_if.overflow <= 0;
    end
    else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= fifo_if.data_in;
        fifo_if.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        fifo_if.overflow <= 0;
    end
    else if(fifo_if.full & fifo_if.wr_en) begin
        fifo_if.wr_ack <= 0;
        fifo_if.overflow <= 1;
    end
    else begin
        fifo_if.overflow <= 0;
    end
end

```

- Bug 5



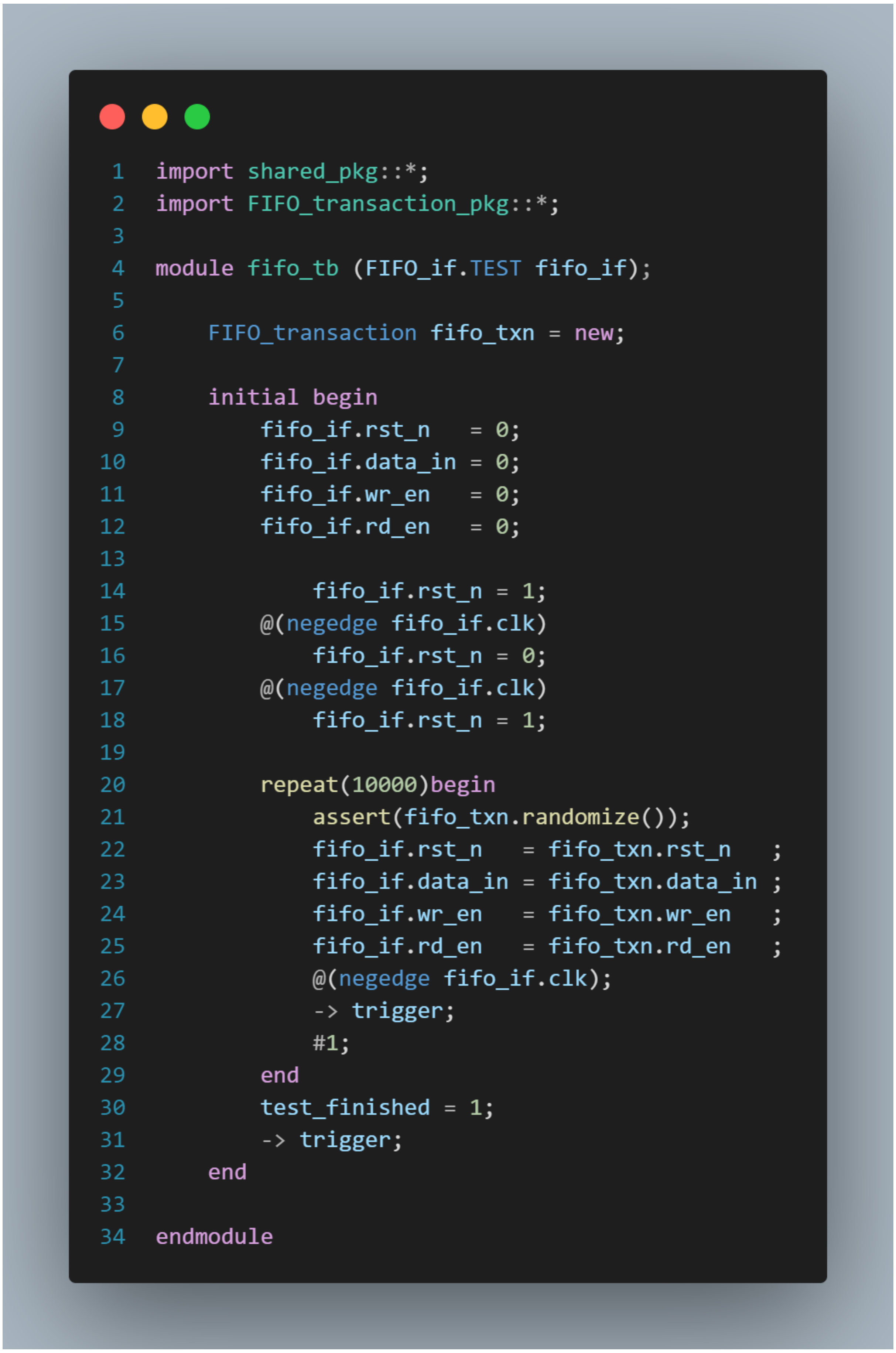
```
always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b10) && !fifo_if.full)
            count <= count + 1;
        else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b01)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

- Bug 6



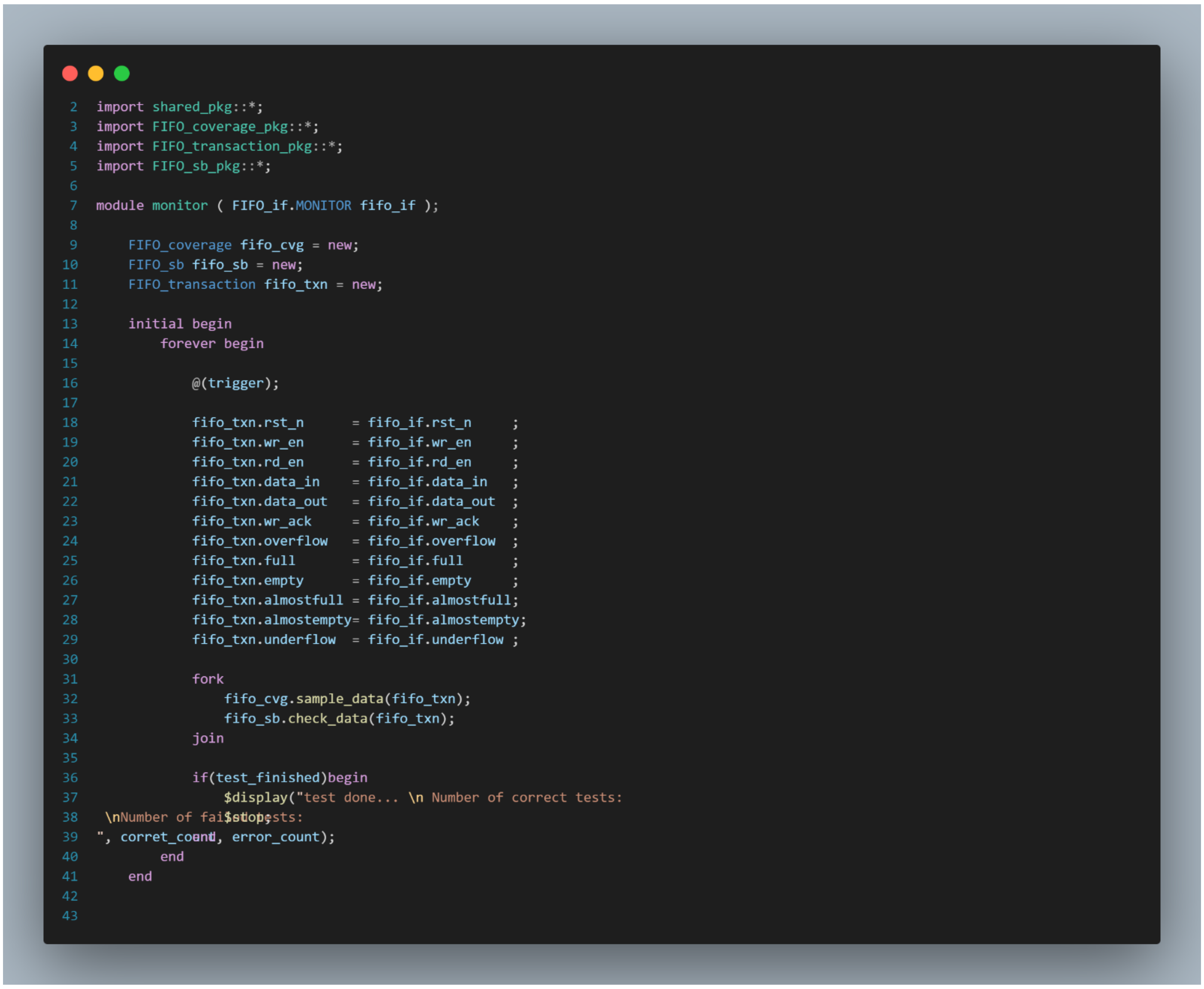
```

assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifo_if.empty = (count == 0)? 1 : 0;

assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
assign fifo_if.almostempty = (count == 1)? 1 : 0;

```

⇨ Tb



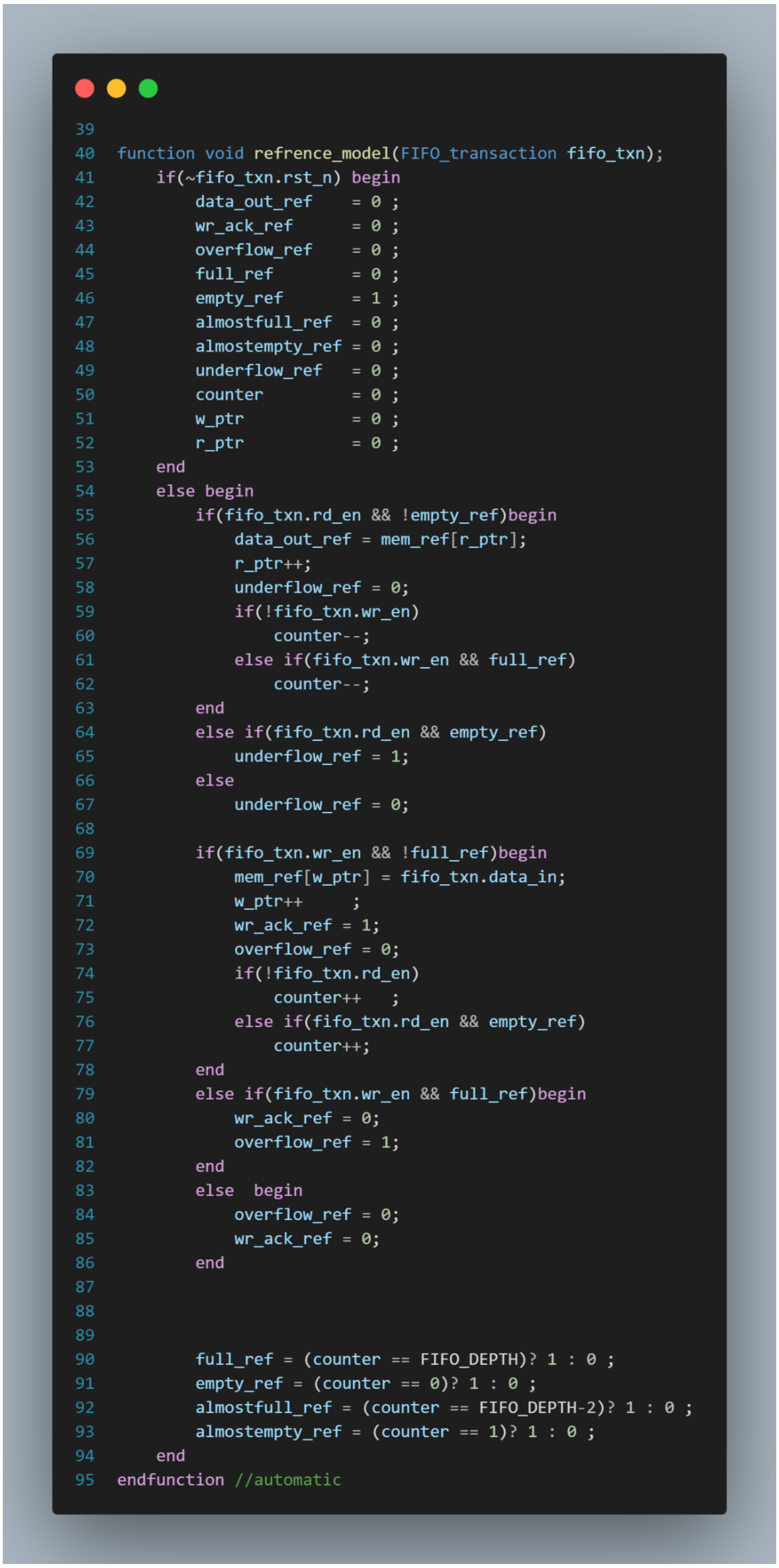
```
import shared_pkg::*;
import FIFO_transaction_pkg::*;

module fifo_tb (FIFO_if.TEST fifo_if);

    FIFO_transaction fifo_txn = new;

    initial begin
        fifo_if.rst_n   = 0;
        fifo_if.data_in = 0;
        fifo_if.wr_en   = 0;
        fifo_if.rd_en   = 0;

            fifo_if.rst_n = 1;
        @(negedge fifo_if.clk)
            fifo_if.rst_n = 0;
        @(negedge fifo_if.clk)
            fifo_if.rst_n = 1;

        repeat(10000)begin
            assert(fifo_txn.randomize());
            fifo_if.rst_n   = fifo_txn.rst_n   ;
            fifo_if.data_in = fifo_txn.data_in ;
            fifo_if.wr_en   = fifo_txn.wr_en   ;
            fifo_if.rd_en   = fifo_txn.rd_en   ;
            @(negedge fifo_if.clk);
            -> trigger;
            #1;
        end
        test_finished = 1;
        -> trigger;
    end

endmodule
```

⇨ Monitor



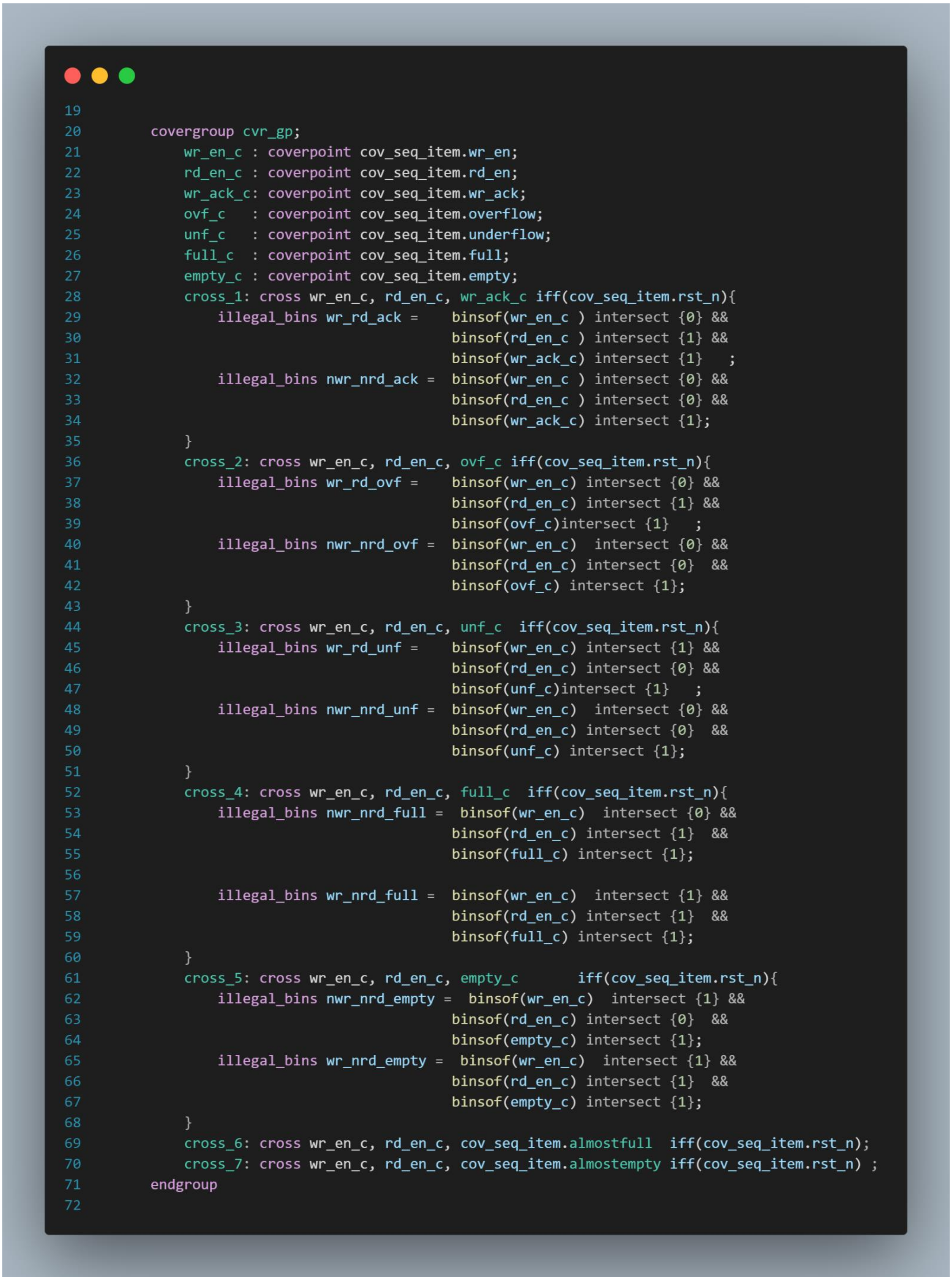
```
import shared_pkg::*;
import FIFO_coverage_pkg::*;
import FIFO_transaction_pkg::*;
import FIFO_sb_pkg::*;

module monitor ( FIFO_if.MONITOR fifo_if );

    FIFO_coverage fifo_cvg = new;
    FIFO_sb fifo_sb = new;
    FIFO_transaction fifo_txn = new;

    initial begin
        forever begin

            @(trigger);

            fifo_txn.rst_n       = fifo_if.rst_n      ;
            fifo_txn.wr_en       = fifo_if.wr_en      ;
            fifo_txn.rd_en       = fifo_if.rd_en      ;
            fifo_txn.data_in     = fifo_if.data_in    ;
            fifo_txn.data_out    = fifo_if.data_out   ;
            fifo_txn.wr_ack      = fifo_if.wr_ack     ;
            fifo_txn.overflow    = fifo_if.overflow   ;
            fifo_txn.full        = fifo_if.full       ;
            fifo_txn.empty       = fifo_if.empty      ;
            fifo_txn.almostfull = fifo_if.almostfull;
            fifo_txn.almostempty= fifo_if.almostempty;
            fifo_txn.underflow   = fifo_if.underflow ;

            fork
                fifo_cvg.sample_data(fifo_txn);
                fifo_sb.check_data(fifo_txn);
            join

            if(test_finished)begin
                $display("test done... \n Number of correct tests:
 \nNumber of failed tests:
                $stop;
", corret_count, error_count);
        end
    end


```

⇨ Reference model



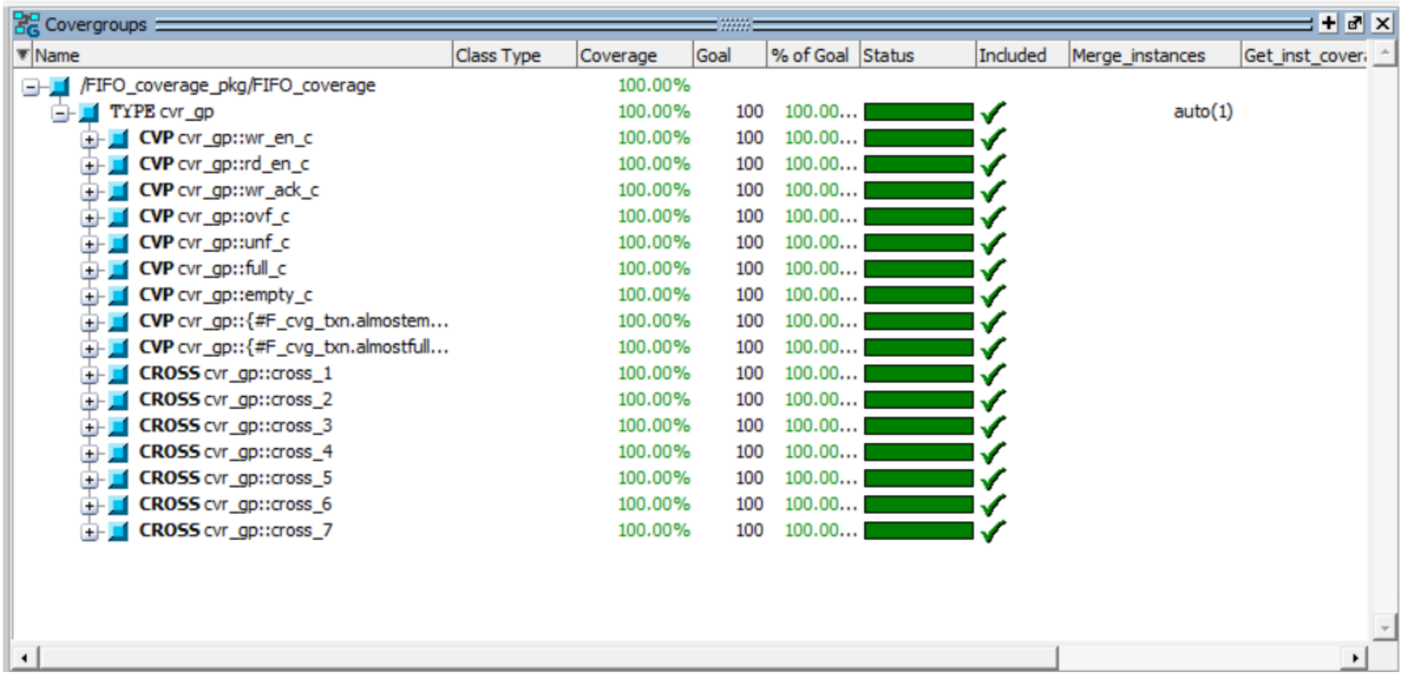
```
function void refrence_model(FIFO_transaction fifo_txn);
    if(~fifo_txn.rst_n) begin
        data_out_ref     = 0 ;
        wr_ack_ref       = 0 ;
        overflow_ref     = 0 ;
        full_ref         = 0 ;
        empty_ref        = 1 ;
        almostfull_ref   = 0 ;
        almostempty_ref  = 0 ;
        underflow_ref    = 0 ;
        counter          = 0 ;
        w_ptr            = 0 ;
        r_ptr            = 0 ;
    end
    else begin
        if(fifo_txn.rd_en && !empty_ref)begin
            data_out_ref = mem_ref[r_ptr];
            r_ptr++;
            underflow_ref = 0;
            if(!fifo_txn.wr_en)
                counter--;
            else if(fifo_txn.wr_en && full_ref)
                counter--;
        end
        else if(fifo_txn.rd_en && empty_ref)
            underflow_ref = 1;
        else
            underflow_ref = 0;

        if(fifo_txn.wr_en && !full_ref)begin
            mem_ref[w_ptr] = fifo_txn.data_in;
            w_ptr++      ;
            wr_ack_ref = 1;
            overflow_ref = 0;
            if(!fifo_txn.rd_en)
                counter++    ;
            else if(fifo_txn.rd_en && empty_ref)
                counter++;
        end
        else if(fifo_txn.wr_en && full_ref)begin
            wr_ack_ref = 0;
            overflow_ref = 1;
        end
        else  begin
            overflow_ref = 0;
            wr_ack_ref = 0;
        end



        full_ref = (counter == FIFO_DEPTH)? 1 : 0 ;
        empty_ref = (counter == 0)? 1 : 0 ;
        almostfull_ref = (counter == FIFO_DEPTH-2)? 1 : 0 ;
        almostempty_ref = (counter == 1)? 1 : 0 ;
    end
endfunction //automatic
```

⇨ Coverage group



```

    covergroup cvr_gp;
        wr_en_c : coverpoint cov_seq_item.wr_en;
        rd_en_c : coverpoint cov_seq_item.rd_en;
        wr_ack_c: coverpoint cov_seq_item.wr_ack;
        ovf_c    : coverpoint cov_seq_item.overflow;
        unf_c    : coverpoint cov_seq_item.underflow;
        full_c   : coverpoint cov_seq_item.full;
        empty_c : coverpoint cov_seq_item.empty;
        cross_1: cross wr_en_c, rd_en_c, wr_ack_c iff(cov_seq_item.rst_n){
            illegal_bins wr_rd_ack =     binsof(wr_en_c ) intersect {0} &&
                                         binsof(rd_en_c ) intersect {1} &&
                                         binsof(wr_ack_c) intersect {1}   ;
            illegal_bins nwr_nrd_ack =   binsof(wr_en_c ) intersect {0} &&
                                         binsof(rd_en_c ) intersect {0} &&
                                         binsof(wr_ack_c) intersect {1};
        }
        cross_2: cross wr_en_c, rd_en_c, ovf_c iff(cov_seq_item.rst_n){
            illegal_bins wr_rd_ovf =     binsof(wr_en_c) intersect {0} &&
                                         binsof(rd_en_c) intersect {1} &&
                                         binsof(ovf_c)intersect {1}   ;
            illegal_bins nwr_nrd_ovf =   binsof(wr_en_c)  intersect {0} &&
                                         binsof(rd_en_c) intersect {0}  &&
                                         binsof(ovf_c) intersect {1};
        }
        cross_3: cross wr_en_c, rd_en_c, unf_c  iff(cov_seq_item.rst_n){
            illegal_bins wr_rd_unf =     binsof(wr_en_c) intersect {1} &&
                                         binsof(rd_en_c) intersect {0} &&
                                         binsof(unf_c)intersect {1}   ;
            illegal_bins nwr_nrd_unf =   binsof(wr_en_c)  intersect {0} &&
                                         binsof(rd_en_c) intersect {0}  &&
                                         binsof(unf_c) intersect {1};
        }
        cross_4: cross wr_en_c, rd_en_c, full_c  iff(cov_seq_item.rst_n){
            illegal_bins nwr_nrd_full =  binsof(wr_en_c)  intersect {0} &&
                                         binsof(rd_en_c) intersect {1}  &&
                                         binsof(full_c) intersect {1};

            illegal_bins wr_nrd_full =   binsof(wr_en_c)  intersect {1} &&
                                         binsof(rd_en_c) intersect {1}  &&
                                         binsof(full_c) intersect {1};
        }
        cross_5: cross wr_en_c, rd_en_c, empty_c       iff(cov_seq_item.rst_n){
            illegal_bins nwr_nrd_empty =  binsof(wr_en_c)  intersect {1} &&
                                         binsof(rd_en_c) intersect {0}  &&
                                         binsof(empty_c) intersect {1};
            illegal_bins wr_nrd_empty =  binsof(wr_en_c)  intersect {1} &&
                                         binsof(rd_en_c) intersect {1}  &&
                                         binsof(empty_c) intersect {1};
        }
        cross_6: cross wr_en_c, rd_en_c, cov_seq_item.almostfull  iff(cov_seq_item.rst_n);
        cross_7: cross wr_en_c, rd_en_c, cov_seq_item.almostempty iff(cov_seq_item.rst_n) ;
    endgroup

```

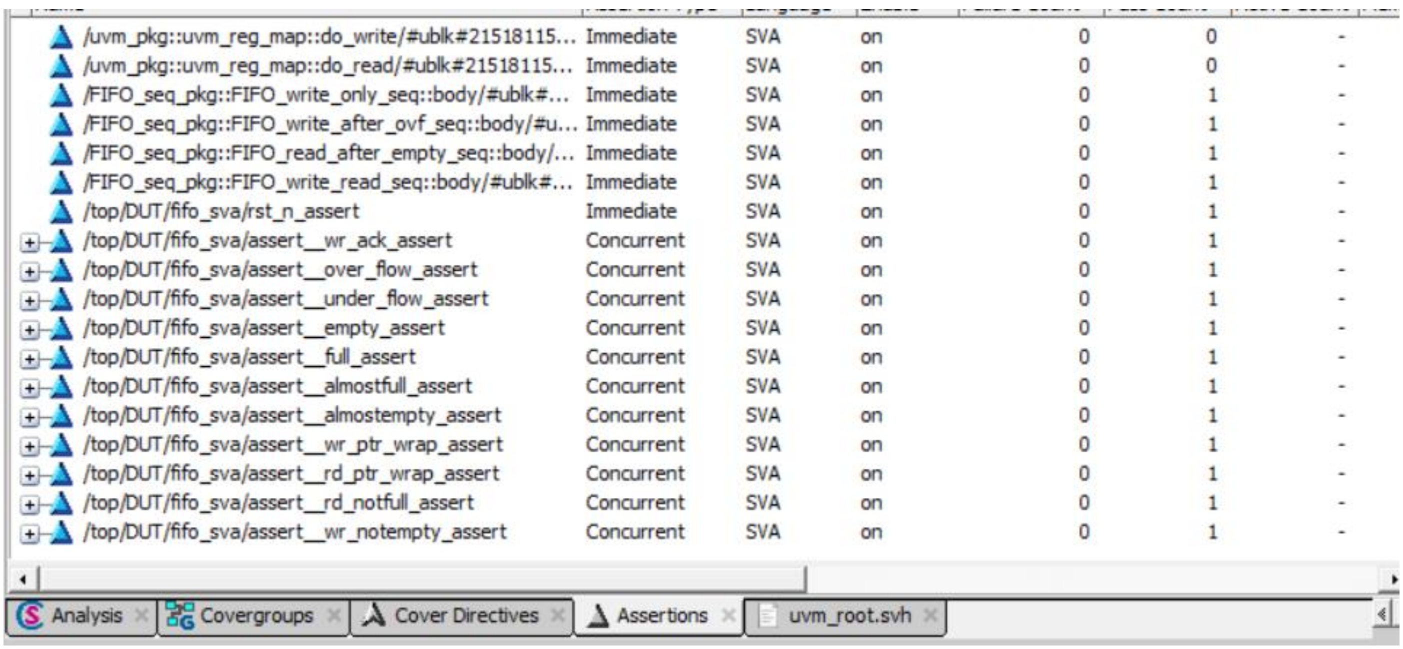
⇨ Functional coverage



Covergroups

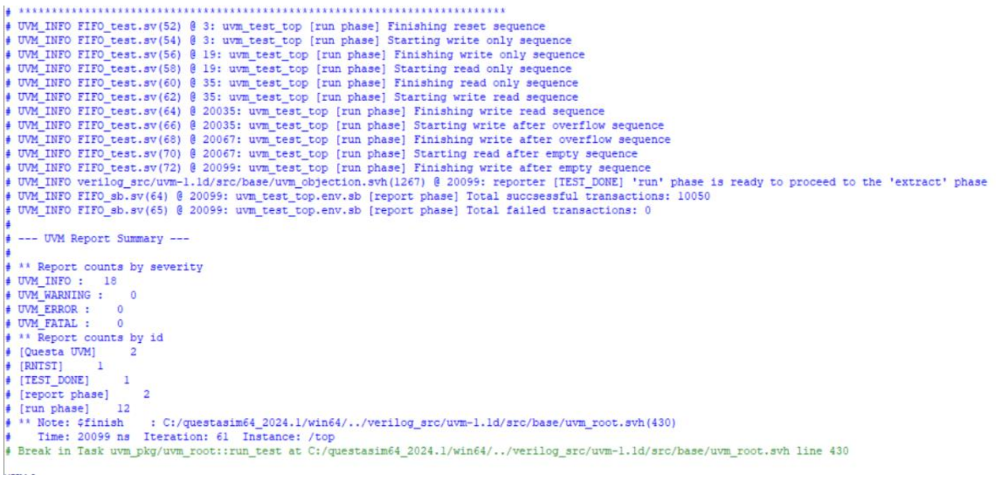
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_cover
/FIFO_coverage_pkg/FIFO_coverage		100.00%						
TYPE cvr_gp		100.00%	100	100.00...	🟩	✓	auto(1)	
CVP cvr_gp::wr_en_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::rd_en_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::wr_ack_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::ovf_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::unf_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::full_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::empty_c		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::{#F_cvg_txn.almostem...		100.00%	100	100.00...	🟩	✓		
CVP cvr_gp::{#F_cvg_txn.almostfull...		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_1		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_2		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_3		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_4		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_5		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_6		100.00%	100	100.00...	🟩	✓		
CROSS cvr_gp::cross_7		100.00%	100	100.00...	🟩	✓		

⇨ Assertions



/uvm_pkg::uvm_reg_map::do_write/#ublk#21518115...	Immediate	SVA	on	0	0	-
/uvm_pkg::uvm_reg_map::do_read/#ublk#21518115...	Immediate	SVA	on	0	0	-
/FIFO_seq_pkg::FIFO_write_only_seq::body/#ublk#...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_write_after_ovf_seq::body/#u...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_read_after_empty_seq::body/...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_write_read_seq::body/#ublk#...	Immediate	SVA	on	0	1	-
/top/DUT/fifo_sva/rst_n_assert	Immediate	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__wr_ack_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__over_flow_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__under_flow_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__empty_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__full_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__almostfull_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__almostempty_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__wr_ptr_wrap_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__rd_ptr_wrap_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__rd_notfull_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert__wr_notempty_assert	Concurrent	SVA	on	0	1	-

⇨ UVM report



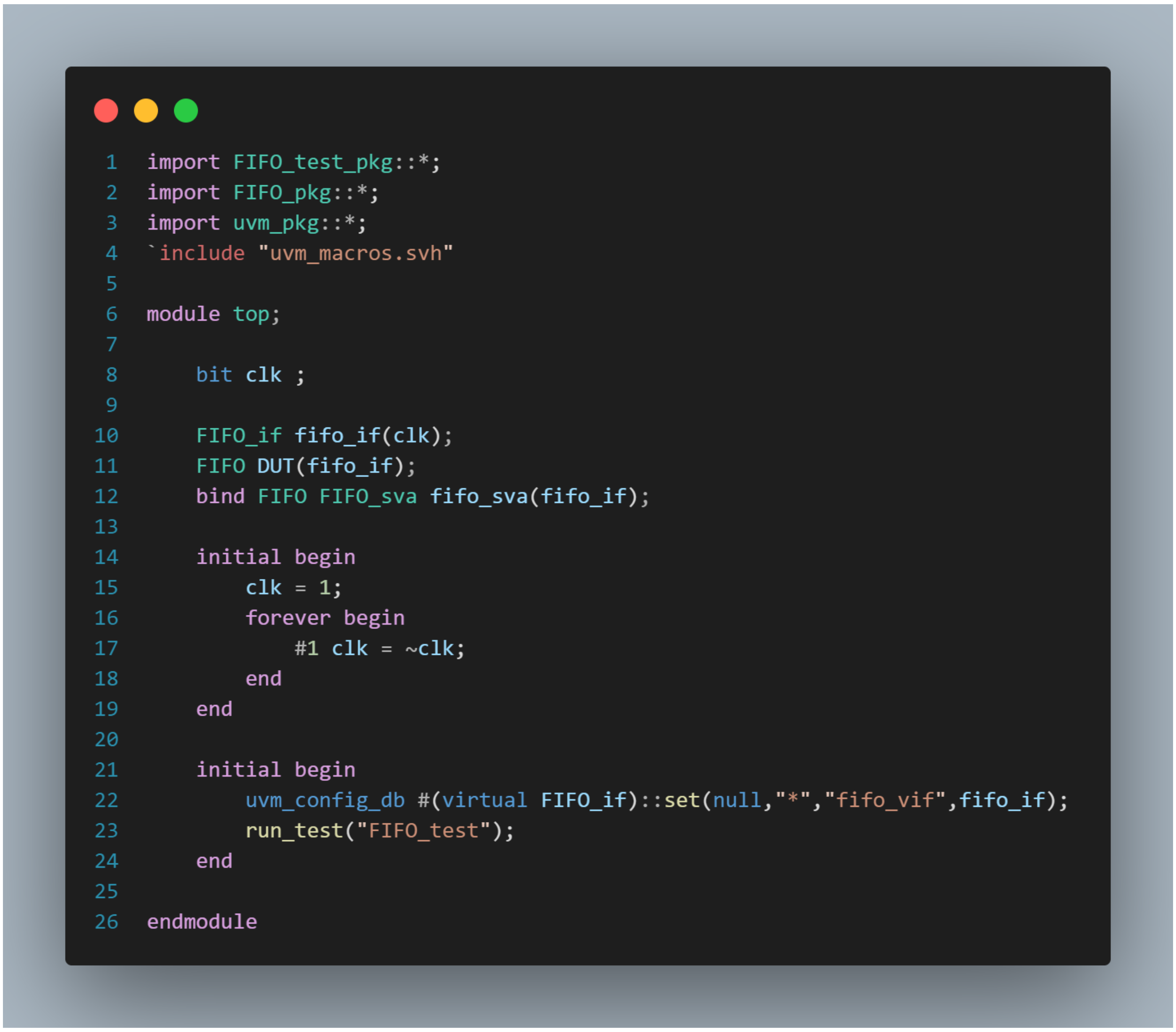
```
# ***************************************************************************
# UVM_INFO FIFO_test.sv(52) @ 3: uvm_test_top [run phase] Finishing reset sequence
# UVM_INFO FIFO_test.sv(54) @ 3: uvm_test_top [run phase] Starting write only sequence
# UVM_INFO FIFO_test.sv(56) @ 19: uvm_test_top [run phase] Finishing write only sequence
# UVM_INFO FIFO_test.sv(58) @ 19: uvm_test_top [run phase] Starting read only sequence
# UVM_INFO FIFO_test.sv(60) @ 35: uvm_test_top [run phase] Finishing read only sequence
# UVM_INFO FIFO_test.sv(62) @ 35: uvm_test_top [run phase] Starting write read sequence
# UVM_INFO FIFO_test.sv(64) @ 20035: uvm_test_top [run phase] Finishing write read sequence
# UVM_INFO FIFO_test.sv(66) @ 20035: uvm_test_top [run phase] Starting write after overflow sequence
# UVM_INFO FIFO_test.sv(68) @ 20067: uvm_test_top [run phase] Finishing write after overflow sequence
# UVM_INFO FIFO_test.sv(70) @ 20067: uvm_test_top [run phase] Starting read after empty sequence
# UVM_INFO FIFO_test.sv(72) @ 20099: uvm_test_top [run phase] Finishing write after empty sequence
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20099: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_sb.sv(64) @ 20099: uvm_test_top.env.sb [report phase] Total succsessful transactions: 10050
# UVM_INFO FIFO_sb.sv(65) @ 20099: uvm_test_top.env.sb [report phase] Total failed transactions: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   18
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report phase]     2
# [run phase]    12
# ** Note: $finish    : C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 20099 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

⇨ Features && Check

Feature	Assertion
When FIFO is full the wr_ack shall never rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.full) |=> !fifo_if.wr_ack ;`
When FIFO is almost full and only read operation occures full flag shall rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (!fifo_if.rd_en && fifo_if.wr_en && fifo_if.almostfull) |=> fifo_if.full ;`
When FIFO is almost empty and only write operation occures empty flag shall rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.rd_en && !fifo_if.wr_en && fifo_if.almostempty) |=> fifo_if.empty ;`
When FIFO is full and both wr_rn and rd_en are high, write operation is rejected(full -> 0)	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.rd_en && fifo_if.full) |=> !fifo_if.full ;`
When FIFO is empty and both wr_rn and rd_en are high, read operation is rejected(empty -> 0)	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.empty && fifo_if.wr_en) |=> !fifo_if.empty ;`
When FIFO is not full and wr_en rise the wr_ack shall rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en && !fifo_if.full) |=> fifo_if.wr_ack ;`
When FIFO is full and wr_en rise the over_flow shall rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en && fifo_if.full) |=> fifo_if.overflow ;`
When FIFO is empty and rd_en rise the underflow shall rise	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.rd_en && fifo_if.empty) |=> fifo_if.underflow ;`
If the internal counter == 0 , FIFO is empty	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count==0) |-> fifo_if.empty ;`
If the internal counter == FIFO_DEPTH , FIFO is full	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count==FIFO_DEPTH) |-> fifo_if.full ;`
If the internal counter == 1 , FIFO is almostempty	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count==FIFO_DEPTH-1) |-> fifo_if.almostfull ;`
If the internal counter == FIFO_DEPTH -1, FIFO is almostfull	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count==1) |-> fifo_if.almostempty ;`
When wr_ptr == FIFO_DEPTH-1 and a write operation is accepted counter should wrap to 0	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.wr_ptr == FIFO_DEPTH-1 && fifo_if.wr_en && !fifo_if.full) |=> DUT.wr_ptr==0 ;`
When rd_ptr == FIFO_DEPTH-1 and a read operation is accepted counter should wrap to 0	`@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.rd_ptr == FIFO_DEPTH-1 && fifo_if.rd_en && !fifo_if.empty) |=> DUT.rd_ptr==0 ;`

When reset is asserted all pointers and output should return to default	`if(~fifo_if.rst_n) begin`
	`rst_n_assert: assert final ( fifo_if.data_out == 0 && DUT.count == 0 && DUT.wr_ptr == 0 && DUT.rd_ptr == 0 )`
	`else $error("DUT.count = %0d, DUT.wr_ptr = %0d, DUT.rd_ptr = %0d, data_out = %0d", DUT.count, DUT.wr_ptr, DUT.rd_ptr, fifo_if.data_out);`
	`end`

⇨ Top



```systemverilog
import FIFO_test_pkg::*;
import FIFO_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

module top;

    bit clk ;

    FIFO_if fifo_if(clk);
    FIFO DUT(fifo_if);
    bind FIFO FIFO_sva fifo_sva(fifo_if);

    initial begin
        clk = 1;
        forever begin
            #1 clk = ~clk;
        end
    end

    initial begin
        uvm_config_db #(virtual FIFO_if)::set(null,"*","fifo_vif",fifo_if);
        run_test("FIFO_test");
    end

endmodule
```