4/5/2025

SV Project Synchronous FIFO

Abdelrahman Ahmed Sayed

UNDER SUPERVISION OF: KAREEM WASEEM

- **Contents: -**
    - Verification plan
    - Snippets for the FIFO top
    - Snippets for the FIFO fixed design
    - Snippets for the FIFO monitor
    - Snippets for the FIFO transaction
    - Snippets for the FIFO coverage
    - Snippets for the FIFO scoreboard
    - bugs detected
    - Do file
    - QuestaSim snippets and all Coverage report

VERIFICATION PLAN

[Link]

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	set rst_n to be 0 then 1 data_out should be 0 acting like nothing was there	direct in the testbench	----------	in check_data function in golden model
FIFO_2	receiving write_acknowledge after wr_en when not full	Randomization under constraints for write enable	coverpoint for the wr_en signal & for full signal	write_acknowledge_assert will check for the functionality
FIFO_3	receiving overflow hight when wr_en is high and full is high	Randomization under constraints for wr_en	covered by Cross_rd_wr_overflow	overflow_assert will check for the functionality
FIFO_4	receiving underflow when rd_en is high and empty is high	Randomization under constraints for rd_en	covered by Cross_rd_wr_empty	empty_assert will check for the functionality
FIFO_5	receiving full when internal siganl count is equal FIFO_DEPTH	Randomization under constraints for wr_en	covered by Cross_rd_wr_full	full_assert will check for the functionality
FIFO_6	receiving almsotfull when the internal signal count is equal to FIFO_DEPTH - 1	Randomization under constraints for wr_en	covered by Cross_rd_wr_almostf	almsotfull_assert will check for the functionality
FIFO_7	receiving almsotempty when the internal signal count is equal 1	Randomization under constraints for rd_en	covered by Cross_rd_wr_almostem	almsotempty_assert will check for the functionality
FIFO_8	receiving underflow when rd_en is high and empty is high	Randomization under constraints for rd_en	covered by Cross_rd_wr_empty	empty_assert will check for the functionality

---

TOP MODULE

FIFO_top.sv > ...

```
1   module FIFO_top;
2   bit clk;
3   initial begin
4       forever
5           #1 clk=~clk;
6   end
7
8   FIFO_interface if_inst (clk);
9   FIFO dut (if_inst);
10  FIFO_tb fifo_tb_inst (if_inst);
11  FIFO_monitor fifo_monitor (if_inst);
12
13  endmodule
```

FIXED DESIGN:

```
module FIFO(FIFO_interface.DUT fifo_if);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

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 fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        wr_ptr <= 0;
        fifo_if.wr_ack <= 0; /// should make it 0 again if rst_n active
        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;
    end
    else begin
        fifo_if.wr_ack <= 0;
        if (fifo_if.full & fifo_if.wr_en) /// fix & to &&
            fifo_if.overflow <= 1;
        else
            fifo_if.overflow <= 0;
    end
end

always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        rd_ptr <= 0;
        fifo_if.underflow <= 0;
        fifo_if.data_out <= 0;
```

```
        fifo_if.data_out <= 0;
    end
    else if (fifo_if.rd_en && count != 0) begin
        fifo_if.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin
        if(count == 0 && fifo_if.rd_en)begin
            fifo_if.underflow <= 1;
        end
        else begin
            fifo_if.underflow <= 0;
        end
    end
end


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.empty) /////// fixing here
            count <= count + 1;
        else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)
            count <= count - 1;
    end
end

assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifo_if.empty = (count == 0)? 1 : 0;
// assign fifo_if.underflow = (fifo_if.empty && fifo_if.rd_en)? 1 : 0;  ////// SHOULD BE SEQ not COMBINATIONAL
assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; /////// FIXING
assign fifo_if.almostempty = (count == 1)? 1 : 0;
```

```
////////////////////////////////////////////////////////////////////////////////
//////////////////////   assertions    /////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

`ifdef SIM
    always_comb begin
        if (!fifo_if.rst_n)
            Reset_Behavior: assert final ((wr_ptr == 0 && rd_ptr == 0 && count == 0));
    end
    ////////////////////
    property pr1;
        @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
            (!fifo_if.full && fifo_if.wr_en) |=> fifo_if.wr_ack;
    endproperty
    write_acknowledge_assert: assert property (pr1);
    Write_Acknowledge_cover:  cover property (pr1);
    ////////////////////
    property pr2;
        @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
            (fifo_if.full & fifo_if.wr_en) |=> fifo_if.overflow;
    endproperty
    overflow_assert : assert property(pr2);
    overflow_cover : cover property(pr2);
    ////////////////////
    property pr3;
        @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
            (fifo_if.empty && fifo_if.rd_en) |=> fifo_if.underflow;
    endproperty
    underflow_assert : assert property(pr3);
    underflow_cover : cover property(pr3);
    ////////////////////
    property pr4;
        @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
            (count == 0) |-> fifo_if.empty;
    endproperty
    empty_assert: assert property(pr4);
    empty_cover: cover property(pr4);
    ////////////////////
    property pr5;
        @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
            (count == FIFO_DEPTH) |-> fifo_if.full;
    endproperty
    full_assert: assert property(pr5);
    full_cover: cover property(pr5);
    ////////////////////
    property pr6;
```

```
////////////////////
property pr6;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
        (count == FIFO_DEPTH - 1) |-> fifo_if.almostfull;
endproperty
almsotfull_assert: assert property(pr6);
almsotfull_cover: cover property(pr6);
////////////////////
property pr7;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n)
        (count == 1) |-> fifo_if.almostempty;
endproperty
almsotempty_assert: assert property(pr7);
almsotempty_cover: cover property(pr7);
////////////////////
```

FIFO monitor

FIFO_monitor.sv > FIFO_monitor

```
module FIFO_monitor(FIFO_interface.MON fifo_if);

import FIFO_transaction_pkg::*;
import FIFO_coverage_pkg::*;
import FIFO_scoreboard_pkg::*;
import shared_pkg::*;

FIFO_coverage cover_obj;
FIFO_transaction trans_obj;
FIFO_scoreboard score_obj;

initial begin
    cover_obj = new();
    trans_obj = new();
    score_obj = new();

    forever begin
        @(test_trigger);
        @(negedge fifo_if.clk);

        `ifdef DEBUG
        $display("***************  After trigger ******************");
        $display("tb: rst_n=%0d, data_in=%0d, wr_en=%0d, rd_en=%0d", fifo_if.rst_n, fifo_if.data_in, fifo_if.wr_en, fifo_if.rd_en);
        $display("dut_out = %0d", fifo_if.data_out);
        $display("************************************************");
        `endif

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

        fork
            begin

                cover_obj.sample_data(trans_obj);
            end
            begin
                score_obj.check_data(trans_obj);
            end
        join
        if (test_finish) begin
            $display("Simulation finished");
            $display("Correct count = %0d", correct_count);
            $display("Error count   = %0d", error_count);
            $stop;
        end
    end
end

endmodule
```

FIFO transaction

```
import shared_pkg::*;

class FIFO_transaction;
    rand logic [FIFO_WIDTH-1:0] data_in;
    rand logic rst_n, wr_en, rd_en;
    logic [FIFO_WIDTH-1:0] data_out;
    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;
    int RD_EN_ON_DIST, WR_EN_ON_DIST;

    function new (int RD_EN_ON_DIST_params=30, WR_EN_ON_DIST_params=70);
        RD_EN_ON_DIST = RD_EN_ON_DIST_params;
        WR_EN_ON_DIST = WR_EN_ON_DIST_params;
    endfunction

    constraint Reset_con {
        rst_n dist {1:/90, 0:/10};
    }

    constraint Wr_en_con{
        wr_en dist {1:/WR_EN_ON_DIST, 0:/(100-WR_EN_ON_DIST)};
    }

    constraint Rd_en_con{
        wr_en dist {1:/RD_EN_ON_DIST, 0:/(100-RD_EN_ON_DIST)};
    }
endclass

endpackage
```

FIFO coverage

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


class FIFO_coverage;
    FIFO_transaction F_cvg_txn = new();



    covergroup w_r_outs_cover_group;

        wr_en_cp    : coverpoint  F_cvg_txn.wr_en;
        rd_en_cp    : coverpoint  F_cvg_txn.rd_en;
        ack_cp      : coverpoint  F_cvg_txn.wr_ack;
        overflow_cp : coverpoint  F_cvg_txn.overflow;
        full_cp     : coverpoint  F_cvg_txn.full;
        underf_cp   : coverpoint  F_cvg_txn.underflow;

        Cross_rd_wr_Ack      : cross wr_en_cp, rd_en_cp, ack_cp{
                                ignore_bins wr_en_with_wr_ack = ! binsof(wr_en_cp) intersect {1} && binsof(ack_cp) intersect {1};
                                ignore_bins rd_en_active_with_wr_ack = ! binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(ack_cp) intersect {1};
                               }

        Cross_rd_wr_overflow : cross wr_en_cp, rd_en_cp, overflow_cp{
                                ignore_bins wr_en_with_overflow = ! binsof(wr_en_cp) intersect {1} && binsof(overflow_cp) intersect {1};
                               }
        Cross_rd_wr_full     : cross wr_en_cp, rd_en_cp, full_cp{
                                ignore_bins write_full = ! binsof(wr_en_cp) intersect {1} && binsof(full_cp) intersect {1};
                                ignore_bins all_ones = binsof(wr_en_cp) intersect {1}  && binsof(full_cp) intersect {1} && binsof(rd_en_cp) intersect {1};
                               }
        Cross_rd_wr_empty    : cross wr_en_cp, rd_en_cp, F_cvg_txn.empty;
        Cross_rd_wr_almostf  : cross wr_en_cp, rd_en_cp, F_cvg_txn.almostfull;
        Cross_rd_wr_almostem : cross wr_en_cp, rd_en_cp, F_cvg_txn.almostempty;
        Cross_rd_wr_unoverf  : cross wr_en_cp, rd_en_cp, underf_cp{
            ignore_bins one_0_one = binsof (wr_en_cp) intersect {1}  && binsof(rd_en_cp) intersect {0}  && binsof(underf_cp)intersect {1} ;
            ignore_bins zerp_zero_one = binsof (wr_en_cp) intersect {0}  && binsof(rd_en_cp) intersect {0}  && binsof(underf_cp) intersect {1} ;

        }
    endgroup

    function new();
        w_r_outs_cover_group = new();
    endfunction

    function void sample_data(FIFO_transaction F_txn);
        F_cvg_txn = F_txn;
        w_r_outs_cover_group.sample();
    endfunction

endclass

endpackage
```

FIFO scoreboard:

```
package FIFO_scoreboard_pkg;
import shared_pkg::*;
import FIFO_transaction_pkg::*;

class FIFO_scoreboard;
    logic [FIFO_WIDTH-1:0] data_out_ref;


    bit [FIFO_WIDTH-1:0] fifo_ref[$];

    function void check_data(FIFO_transaction F_txn);
        reference_model(F_txn);
        if (data_out_ref !== F_txn.data_out)begin
            error_count++;
            $display("==============  ERROR  ========================");
            $display("scoreboard_ERROR: output => %0d not equal the ref Out => %0d", F_txn.data_out, data_out_ref);
            $display("When rst_n: %0d, wr_en: %0d, rd_en: %0d, data_in: %0d", F_txn.rst_n, F_txn.wr_en, F_txn.rd_en, F_txn.data_in);
            $display("=====================================================");
        end
        else begin
            correct_count++;
            $display("===============  Correct   ====================");
            $display("Correct:: output => %0d equal the ref Out => %0d", F_txn.data_out, data_out_ref);
            $display("When rst_n: %0d, wr_en: %0d, rd_en: %0d, data_in: %0d", F_txn.rst_n, F_txn.wr_en, F_txn.rd_en, F_txn.data_in);
            $display("=====================================================");
        end
    endfunction

    function void reference_model(FIFO_transaction F_txn_param);
        if (!F_txn_param.rst_n)begin
            data_out_ref = 0;
            fifo_ref.delete();
        end
        else begin
            if (F_txn_param.wr_en && !F_txn_param.rd_en) begin
                if (!isFull())begin
                    fifo_ref.push_back(F_txn_param.data_in);
                    $display("HERE in pushing 1");
                end
            end
            else if (F_txn_param.rd_en && !F_txn_param.wr_en) begin
                    if (!isEmpty())begin
                        data_out_ref = fifo_ref.pop_front();
                        $display("HERE in popping 1");

                    end
                    else
                        $display("EMpty");
                end
            else if (F_txn_param.wr_en && F_txn_param.rd_en) begin
                if (!isEmpty() && !isFull()) begin
                    data_out_ref = fifo_ref.pop_front();
                    fifo_ref.push_back(F_txn_param.data_in);
                    $display("HERE in pushing popping");
                end
                else if (isEmpty()) begin
                    fifo_ref.push_back(F_txn_param.data_in);
                    $display("HERE in pushing 2");
                end
                else if (isFull()) begin
                    data_out_ref = fifo_ref.pop_front();
                    $display("HERE in popping 2");

                end
```

```
                    $display("HERE in pushing popping");
                end
                else if (isEmpty()) begin
                    fifo_ref.push_back(F_txn_param.data_in);
                    $display("HERE in pushing 2");
                end
                else if (isFull()) begin
                    data_out_ref = fifo_ref.pop_front();
                    $display("HERE in popping 2");

                end
            end
        end

        `ifdef DEBUG
        $display("------------------INSIDE GOLDEN MODEL-------------");
        $display("When rst_n: %0d, wr_en: %0d, rd_en: %0d, data_in: %0d", F_txn_param.rst_n, F_txn_param.wr_en, F_txn_param.rd_en, F_txn_param.data_in);
        $display("size of queue: %0d", fifo_ref.size());
        $display("(1)%0d \n(2)%0d \n(3)%0d \n(4)%0d \n(5)%0d", fifo_ref[0], fifo_ref[1], fifo_ref[2], fifo_ref[3], fifo_ref[4]);
        $display("(6)%0d \n(7)%0d \n(8)%0d \n(9)%0d \n(10)%0d", fifo_ref[5], fifo_ref[6], fifo_ref[7], fifo_ref[8], fifo_ref[9]);
        $display("(11)%0d \n(12)%0d", fifo_ref[10], fifo_ref[11]);
        $display("-----------------------------------------------------");
        `endif
    endfunction

    function bit isFull();
        if (fifo_ref.size() == FIFO_DEPTH)
            return 1;
        return 0;
    endfunction

    function bit isEmpty();
        if (fifo_ref.size() == 0)
            return 1;
        return 0;
    endfunction
```

FIFO testbench:

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

FIFO_transaction trans_obj = new();

module FIFO_tb (FIFO_interface.TEST fifo_if);

int loop_count = 0;
initial begin
    #0;

    direct_test;

    repeat(10000)begin
        assert(trans_obj.randomize());
        fifo_if.data_in = trans_obj.data_in;
        fifo_if.wr_en = trans_obj.wr_en;
        fifo_if.rd_en = trans_obj.rd_en;
        fifo_if.rst_n = trans_obj.rst_n;

        `ifdef DEBUG
        $display("***************  in  tb ******************");
        $display("tb: rst_n=%0d, data_in=%0d, wr_en=%0d, rd_en=%0d", trans_obj.rst_n, trans_obj.data_in, trans_obj.wr_en, trans_obj.rd_en);
        $display("*****************************************");
        `endif

        -> test_trigger;
        @(negedge fifo_if.clk);


    end
    -> test_trigger;
    test_finish = 1;
end
```

```
task direct_test;
    #0;
    fifo_if.rst_n = 0;
    -> test_trigger;
    @(negedge fifo_if.clk);
    fifo_if.rst_n = 1;
    $display("Writting.....................................................");
    fifo_if.data_in = 121;
    fifo_if.wr_en = 1;
    fifo_if.rd_en = 0;
    -> test_trigger;
    @(negedge fifo_if.clk);

    fifo_if.data_in = 122;
    fifo_if.wr_en = 1;
    fifo_if.rd_en = 0;
    -> test_trigger;
    @(negedge fifo_if.clk);

    fifo_if.data_in = 123;
    fifo_if.wr_en = 1;
    fifo_if.rd_en = 0;
    -> test_trigger;
    @(negedge fifo_if.clk);
    ////////////////////
    $display("Reading .....................................................");
    fifo_if.data_in = 333;
    fifo_if.wr_en = 0;
    fifo_if.rd_en = 1;
    -> test_trigger;
    @(negedge fifo_if.clk);

    fifo_if.data_in = 444;
    fifo_if.wr_en = 0;
    fifo_if.rd_en = 1;
    -> test_trigger;
    @(negedge fifo_if.clk);

    fifo_if.data_in = 555;
    fifo_if.wr_en = 0;
```

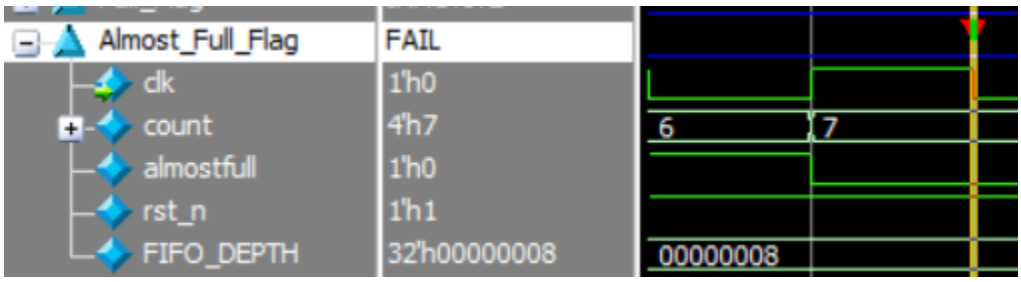
BUGS

- **Bug 1(almostfull)**

In the design:

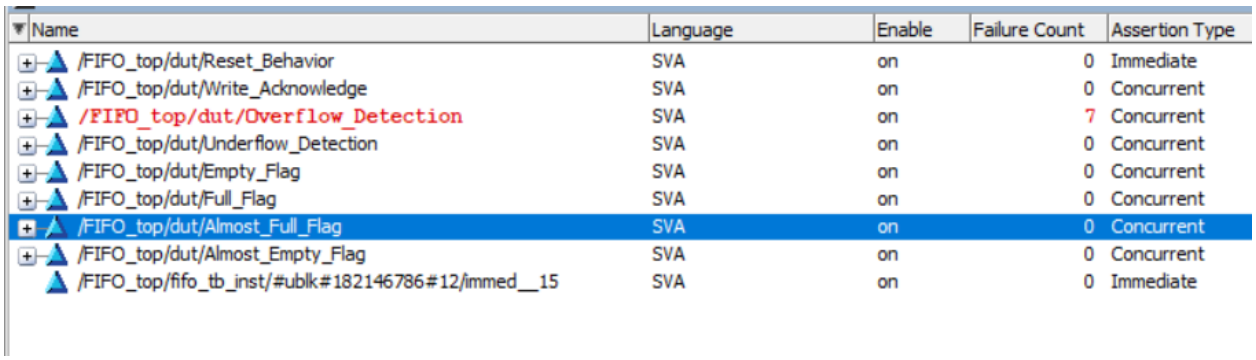
```
62
63   assign fifo_if.almostfull = (count == FIFO_DEPTH-2)? 1 : 0; /
64
```

In assertions:



It should be:

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



- **BUG 2** (wr_ack)

In design:

missing this part:

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

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; /// should make it 0 again if rst_n active
    end
    else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= fifo_if.data_in;
```

Before it:

Assertions

Name	Language	Enable	Failure Count	Assertion Type
/FIFO_top/dut/write_acknowledge_assert	SVA	on	386	Concurrent
/FIFO_top/dut/overflow_assert	SVA	on	0	Concurrent

After it:

Assertions

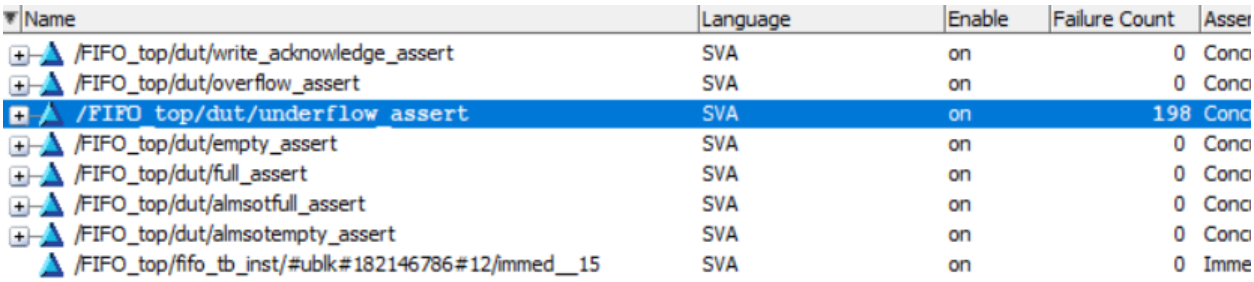
Name	Language	Enable	Failure Count	Assertion Type	Pass Count
/FIFO_top/dut/write_acknowledge_assert	SVA	on	0	Concurrent	1
/FIFO_top/dut/overflow_assert	SVA	on	0	Concurrent	1

- **BUG 3(underflow)**

In the design:

```
assign fifo_if.empty = (count == 0)? 1 : 0;
// assign fifo_if.underflow = (fifo_if.empty && fifo_if.rd_en)? 1 : 0;  ////// SHOULD BE SEQ not
assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; /////// FIXING
```

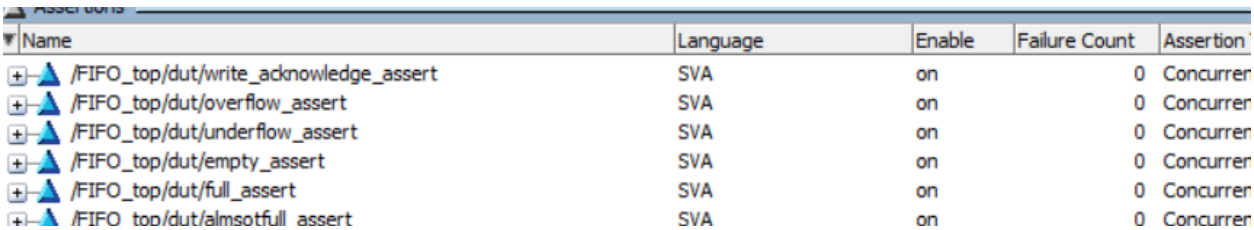
Underflow should be sequential



Name	Language	Enable	Failure Count	Asser
/FIFO_top/dut/write_acknowledge_assert	SVA	on	0	Conc
/FIFO_top/dut/overflow_assert	SVA	on	0	Conc
/FIFO_top/dut/underflow_assert	SVA	on	198	Conc
/FIFO_top/dut/empty_assert	SVA	on	0	Conc
/FIFO_top/dut/full_assert	SVA	on	0	Conc
/FIFO_top/dut/almsotfull_assert	SVA	on	0	Conc
/FIFO_top/dut/almsotempty_assert	SVA	on	0	Conc
/FIFO_top/fifo_tb_inst/#ublk#182146786#12/immed__15	SVA	on	0	Imme

Should be like this:

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



Name	Language	Enable	Failure Count	Assertion
/FIFO_top/dut/write_acknowledge_assert	SVA	on	0	Concurren
/FIFO_top/dut/overflow_assert	SVA	on	0	Concurren
/FIFO_top/dut/underflow_assert	SVA	on	0	Concurren
/FIFO_top/dut/empty_assert	SVA	on	0	Concurren
/FIFO_top/dut/full_assert	SVA	on	0	Concurren
/FIFO_top/dut/almsotfull_assert	SVA	on	0	Concurren

- **BUG 4** (missing when write and read high)

```
else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.empty)
    count <= count + 1;
else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)
    count <= count - 1;
```

Do file

run_fixed.do

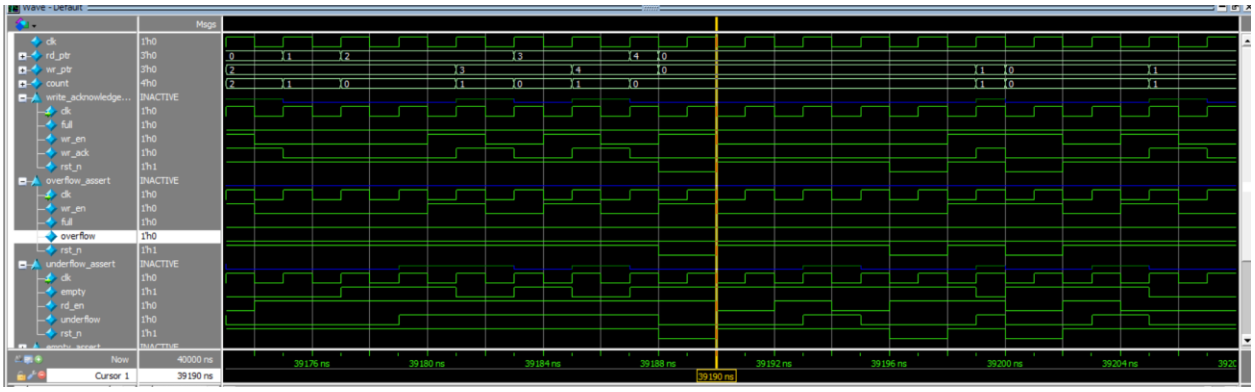
```
vlib work

vlog -f src_files_fixed.list  +define+SIM  +cover
# ADD THIS IF YOU WANT TO SEE EVERYTHING:=> <+define+DEBUG>

vsim -voptargs=+acc work.FIFO_top -cover
add wave *
add wave  /FIFO_top/dut/rd_ptr /FIFO_top/dut/wr_ptr /FIFO_top/dut/count /FIFO_top/dut/write_acknowledge_assert /FIFO_top/dut/overflow_assert /FIFO_top/dut/underf
coverage save FIFO_cov.ucdb -onexit -du work.FIFO
run -all
```

QuestaSim snippets

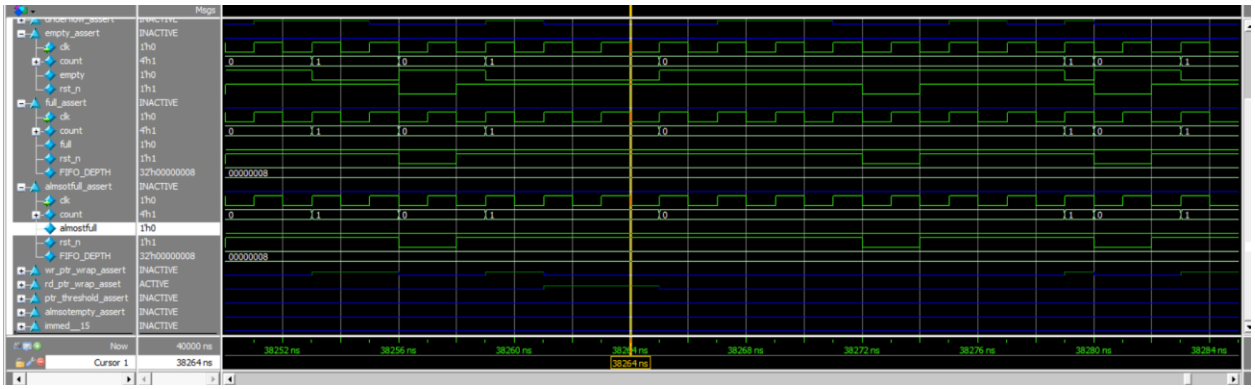
No errors in the fixed design:

```
# ================  Correct   =====================
# Correct:: output => 4249 equal the ref Out => 4249
# When rst_n: 1, wr_en: 0, rd_en: 0, data_in: 32091
# ====================================================
# HERE in pushing 1
# ================  Correct   =====================
# Correct:: output => 4249 equal the ref Out => 4249
# When rst_n: 1, wr_en: 1, rd_en: 0, data_in: 41337
# ====================================================
# HERE in pushing popping
# ================  Correct   =====================
# Correct:: output => 49050 equal the ref Out => 49050
# When rst_n: 1, wr_en: 1, rd_en: 1, data_in: 43235
# ====================================================
# ================  Correct   =====================
# Correct:: output => 49050 equal the ref Out => 49050
# When rst_n: 1, wr_en: 0, rd_en: 0, data_in: 45636
# ====================================================
# ================  Correct   =====================
# Correct:: output => 49050 equal the ref Out => 49050
# When rst_n: 1, wr_en: 0, rd_en: 0, data_in: 16200
# ====================================================
# ================  Correct   =====================
# Correct:: output => 49050 equal the ref Out => 49050
# When rst_n: 1, wr_en: 0, rd_en: 0, data_in: 16200
# ====================================================
# Simulation finished
# Correct count = 10009
# Error count   = 0
# ** Note: $stop    : FIFO_monitor.sv(54)
#    Time: 200180 ns  Iteration: 1  Instance: /FIFO_top/fifo_monitor
```

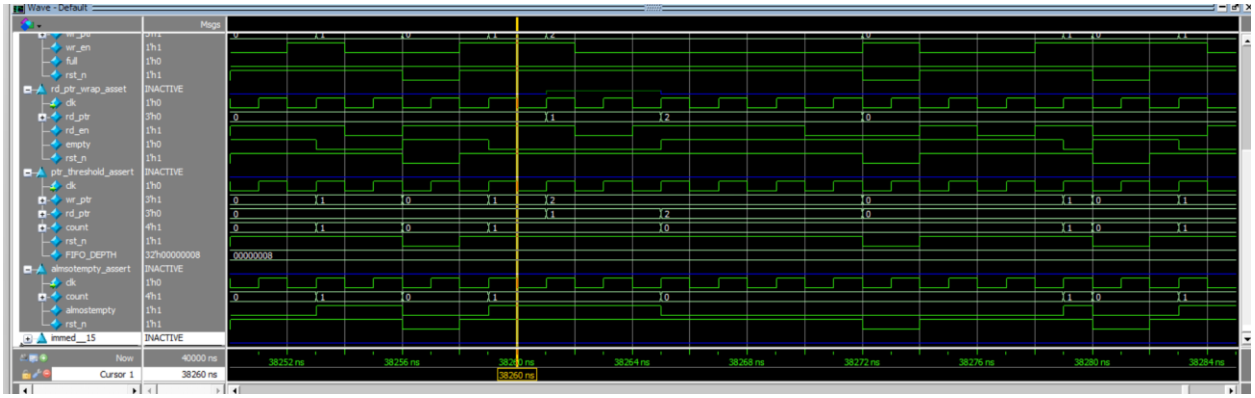
Write ack and Overflow and underflow:



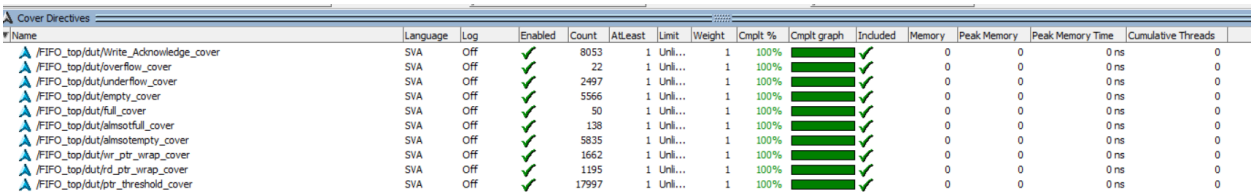
Empty and full and almostfull:



almostempty and Pointer Wraparound Pointer threshold:

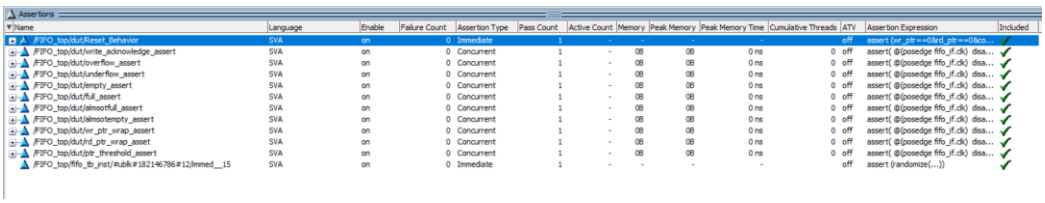


Cover directives :



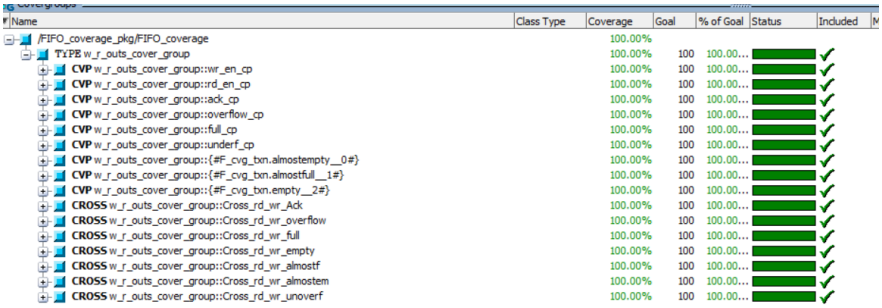
Name	Language	Log	Enabled	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/FIFO_top/dut/Write_Acknowledge_cover	SVA	Off	✓	8053	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/overflow_cover	SVA	Off	✓	22	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/underflow_cover	SVA	Off	✓	2497	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/empty_cover	SVA	Off	✓	5566	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/full_cover	SVA	Off	✓	50	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/almsotfull_cover	SVA	Off	✓	138	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/almsotempty_cover	SVA	Off	✓	5835	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/wr_ptr_wrap_cover	SVA	Off	✓	1662	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/rd_ptr_wrap_cover	SVA	Off	✓	1195	1	Unli...	1	100%		✓	0	0	0 ns	0
/FIFO_top/dut/ptr_threshold_cover	SVA	Off	✓	17997	1	Unli...	1	100%		✓	0	0	0 ns	0

Assertions:



Name	Language	Enable	Failure Count	Assertion Type	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/FIFO_top/dut/Reset_Behavior	SVA	on	0	Immediate	1	-	-	-	-		off	assert (wr_ptr==0)&rd_ptr==0)&co...	✓
/FIFO_top/dut/write_acknowledge_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/overflow_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/underflow_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/empty_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/full_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/almsotfull_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/almsotempty_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/wr_ptr_wrap_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/rd_ptr_wrap_asset	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/dut/ptr_threshold_assert	SVA	on	0	Concurrent	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
/FIFO_top/fifo_tb_inst/#ublk#182146786#12/immed__15	SVA	on	0	Immediate	1	-	-	-	-		off	assert (randomize(...))	✓

Group coverage:



Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/FIFO_coverage_pkg/FIFO_coverage		100.00%				
TYPE w_r_outs_cover_group		100.00%	100	100.00...		
CVP w_r_outs_cover_group::wr_en_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::rd_en_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::ack_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::overflow_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::full_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::underf_cp		100.00%	100	100.00...		
CVP w_r_outs_cover_group::{#F_cvg_txn.almostempty__0#}		100.00%	100	100.00...		
CVP w_r_outs_cover_group::{#F_cvg_txn.almostfull__1#}		100.00%	100	100.00...		
CVP w_r_outs_cover_group::{#F_cvg_txn.empty__2#}		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_Ack		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_overflow		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_full		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_empty		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_almostf		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_almostem		100.00%	100	100.00...		
CROSS w_r_outs_cover_group::Cross_rd_wr_unoverf		100.00%	100	100.00...		

```
203  ----------------------------------------------------------------------------------------------------------------
204      TYPE /FIFO_coverage_pkg/FIFO_coverage/w_r_outs_cover_group
329          Cross Cross_rd_wr_almostem                      100.00%        100          -    Covered
333              Auto, Default and User Defined Bins:
340                  bin <auto[1],auto[0],auto[0]>              1797          1          -    Covered
341                  bin <auto[0],auto[0],auto[0]>              1743          1          -    Covered
342          Cross Cross_rd_wr_unoverf                       100.00%        100          -    Covered
343              covered/total bins:                               6          6          -
344              missing/total bins:                               0          6          -
345              % Hit:                                      100.00%        100          -
346              Auto, Default and User Defined Bins:
347                  bin <auto[1],auto[1],auto[1]>              907          1          -    Covered
348                  bin <auto[1],auto[1],auto[0]>             1550          1          -    Covered
349                  bin <auto[0],auto[1],auto[1]>              891          1          -    Covered
350                  bin <auto[0],auto[1],auto[0]>             1634          1          -    Covered
351                  bin <auto[1],auto[0],auto[0]>             2520          1          -    Covered
352                  bin <auto[0],auto[0],auto[0]>             2506          1          -    Covered
353              Illegal and Ignore Bins:
354                  ignore_bin zerp_zero_one                     0                     -    ZERO
355                  ignore_bin one_0_one                         0                     -    ZERO
356
357  TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
358
359  DIRECTIVE COVERAGE:
360  ----------------------------------------------------------------------------------------------
361  Name                                 Design Design   Lang File(Line)      Hits Status
362                                       Unit   UnitType
363  ----------------------------------------------------------------------------------------------
364  /FIFO_top/dut/Write_Acknowledge_cover    FIFO   Verilog  SVA  FIFO_fixed.sv(99)
365                                                                           4034 Covered
366  /FIFO_top/dut/overflow_cover             FIFO   Verilog  SVA  FIFO_fixed.sv(106)
367                                                                             14 Covered
368  /FIFO_top/dut/underflow_cover            FIFO   Verilog  SVA  FIFO_fixed.sv(113)
369                                                                           1251 Covered
370  /FIFO_top/dut/empty_cover                FIFO   Verilog  SVA  FIFO_fixed.sv(120)
371                                                                           2794 Covered
372  /FIFO_top/dut/full_cover                 FIFO   Verilog  SVA  FIFO_fixed.sv(127)
373                                                                             25 Covered
374  /FIFO_top/dut/almsotfull_cover           FIFO   Verilog  SVA  FIFO_fixed.sv(134)
375                                                                             68 Covered
376  /FIFO_top/dut/almsotempty_cover          FIFO   Verilog  SVA  FIFO_fixed.sv(141)
377                                                                           2958 Covered
378  /FIFO_top/dut/rd_ptr_wrap_cover          FIFO   Verilog  SVA  FIFO_fixed.sv(150)
379                                                                             23 Covered
380
381  TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 8
382
383  Total Coverage By Instance (filtered view): 100.00%
384
385
```

All coverage 100%:

```
                                    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)


    DIRECTIVE COVERAGE:
    ----------------------------------------------------------------------------------------------
    Name                                 Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
    ----------------------------------------------------------------------------------------------
    /\FIFO_top#dut /Write_Acknowledge_cover  FIFO   Verilog  SVA  FIFO_fixed.sv(99)
                                                                                4034 Covered
    /\FIFO_top#dut /overflow_cover          FIFO   Verilog  SVA  FIFO_fixed.sv(106)
                                                                                  14 Covered
    /\FIFO_top#dut /underflow_cover         FIFO   Verilog  SVA  FIFO_fixed.sv(113)
                                                                                1251 Covered
    /\FIFO_top#dut /empty_cover             FIFO   Verilog  SVA  FIFO_fixed.sv(120)
                                                                                2794 Covered
    /\FIFO_top#dut /full_cover              FIFO   Verilog  SVA  FIFO_fixed.sv(127)
                                                                                  25 Covered
    /\FIFO_top#dut /almsotfull_cover        FIFO   Verilog  SVA  FIFO_fixed.sv(134)
                                                                                  68 Covered
    /\FIFO_top#dut /almsotempty_cover       FIFO   Verilog  SVA  FIFO_fixed.sv(141)
                                                                                2958 Covered
    /\FIFO_top#dut /rd_ptr_wrap_cover       FIFO   Verilog  SVA  FIFO_fixed.sv(150)
                                                                                  23 Covered

    TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 8

    ASSERTION RESULTS:
    ---------------------------------------------------------------------
    Name                File(Line)                   Failure      Pass
                                                     Count        Count
    ---------------------------------------------------------------------
    /\FIFO_top#dut /Reset_Behavior
                        FIFO_fixed.sv(91)                  0         1
    /\FIFO_top#dut /write_acknowledge_assert
                        FIFO_fixed.sv(98)                  0         1
    /\FIFO_top#dut /overflow_assert
                        FIFO_fixed.sv(105)                 0         1
    /\FIFO_top#dut /underflow_assert
                        FIFO_fixed.sv(112)                 0         1
    /\FIFO_top#dut /empty_assert
                        FIFO_fixed.sv(119)                 0         1
    /\FIFO_top#dut /full_assert
                        FIFO_fixed.sv(126)                 0         1
    /\FIFO_top#dut /almsotfull_assert
                        FIFO_fixed.sv(133)                 0         1
    /\FIFO_top#dut /almsotempty_assert
                        FIFO_fixed.sv(140)                 0         1
    /\FIFO_top#dut /rd_ptr_wrap_asset
                        FIFO_fixed.sv(149)                 0         1

    Total Coverage By Instance (filtered view): 100.00%

    End time: 09:34:18 on May 06,2025, Elapsed time: 0:00:00
    Errors: 0, Warnings: 0
```

THANK YOU