SV project - Synchronous FIFO

Abdelrahman Mohamed Ali

30 April, 2024

Gmail:	**abdelrahmansalby23@gmail.com**
LinkedIn:	**Abdelrahman Mohamed**
GitHub:	**Abdelrahman1810**

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	asserted the reset at the first 5 clk cycle	Randomization with post_randomize dunction to make sure that at the first 5 clk cycle the reset will be low	included in covergroup CVG in coverpoint of reset named "rst_cp"	_____
FIFO_2	Case: reset activated Check the sequential signal (wr_ack, overflow, underflow, data_out) should be zero	Randomization with post_randomize dunction to make sure that at the first 5 clk cycle the reset will be low	included as a coverpoint of reset and wr_ack, overflow, underflow, data_out included as a cross coverage to check that signal will be in active when reset is active	Output Checked against golden model and checked with combinational assertion (label: assert_wr_akc_rst assert_overflow_rst assert_underflow_rst assert_data_out_rst)
FIFO_3	Case: reset is activated the internal signal (wr_otr, rd_ptr and count) should be inactive	Randomization with post_randomize dunction to make sure that at the first 5 clk cycle the reset will be low	included as a coverpoint of rst_n No coverage for internal signal	Output checked with assertion (label: assert_count_rst, assert_wr_ptr_rst, assert_rd_ptr_rst)
FIFO_4	make reset Inactive most of the time	Randomization under constraint (label: CON_RESET) to make rst_n is active => 95%, and rst_n is Inactive => 5%	included in covergroup CVG in coverpoint of reset named "rst_cp"	_____
FIFO_5	Case: wr_en = 1, rd_en = 0, full = 0, empty = 0. 1.Write operation done 2.wr_ack gets high 3.if count ==FIFO_DEPTH-1 4.then almostfull gets high	Randomization under constraint (label: MY_CON_ONLEY_W) that constraint wr_en to be active and rd_en to be inactive	Included a coverpoint wr_ack_cp. Included a coverpoint almostfull_cp. Included a cross almostfull & wr_en. Included a cross coverage wr_ack & wr_en.	Output Checked against golden model Checked by assertion Label: **ack_active,** **almostfull_count**
FIFO_6	Case: wr_en = 1, rd_en = 0, full = 1. 1.Write operation ignored 2.wr_ack gets low 3.overflow gets high	Randomization under constraint (label: MY_CON_ONLEY_W) that constraint wr_en to be active and rd_en to be inactive	Included a coverpoint overflow_cp Included a cross coverage overflow & wr_en.	Output Checked against golden model Checked by assertion label: **ack_inactive** **overflow_active**
FIFO_7	Case: wr_en = 1, rd_en = 0, empty = 1. 1.Write operation done 2.wr_ack gets high 3.empty gets low 4.almostempty gets high	Randomization under constraint (label: MY_CON_ONLEY_W) that constraint wr_en to be active and rd_en to be inactive	Included a coverpoint wr_ack_cp. Included a coverpoint for empty. Included a coverpoint almostempty_cp. Included a cross coverage wr_ack & empty (ack_empty_cross) Included a cross coverage almostempty & empty (almostempty_empty_cross)	Output Checked against golden model Checked by assertion Label: **ack_active,** **empty_inactive,** **almostempty_from_empty**
FIFO_8	Case: wr_en = 1, rd_en = 0, almostfull = 1. 1.Write operation done 2.wr_ack gets high 3.almostfull gets low 4.count == FIFO_DEPTH so full gets high	Randomization under constraint (label: MY_CON_ONLEY_W) that constraint wr_en to be active and rd_en to be inactive	Included a coverpoint wr_ack_cp. Included a coverpoint full_cp. Included a coverpoint almostfull_cp. Included a cross coverage wr_ack & full (ack_full_wr_cross) Included a cross coverage almostfull & full (almostfull_full_cross)	Output Checked against golden model Checked by assertion Label: **ack_inactive,** **almostempty_inactive,** **full_count**
FIFO_9	Case: wr_en = 1, rd_en = 0, almostempty = 1. 1.Write operation done 2.wr_ack gets high almostempty gets low	Randomization under constraint (label: MY_CON_ONLEY_W) that constraint wr_en to be active and rd_en to be inactive	Included a coverpoint wr_ack_cp. Included a coverpoint almostempty_cp. Included a cross coverage wr_ack & empty (ack_empty_cross) Included a cross coverage almostempty & wr_ack (ack_almostempty_cross)	Output Checked against golden model Checked by assertion Label: **almostempty_inactive**
FIFO_10	Case: wr_en = 0, rd_en = 1, empty = 0, full = 0. 1.Read operation done 2.check data_out	Randomization under constraint (label: MY_CON_ONLEY_R) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a cross coverage data_out &rd_en	Output Checked against golden model Checked by assertion Label: **almostempty_count**
FIFO_11	Case: wr_en = 0, rd_en = 1, empty = 1. 1.Read operation ignored 2.Underflow gets high	Randomization under constraint (label: MY_CON_ONLEY_R) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint underflow_cp Included a cross coverage: underflow &rd_en, underflow &empty	Output Checked against golden model Checked by assertion Label: **underflow_active**
FIFO_12	Case: wr_en = 0, rd_en = 1, full = 1. 1.Read operation done 2.Check data_out 3.full gets low 4.almostfull gets high	Randomization under constraint (label: MY_CON_ONLEY_R) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a coverpoint full_cp. Included a coverpoint almostfull_cp. Included a cross coverage almostfull & full (almostfull_full_cross)	Output Checked against golden model Checked by assertion Label: **almostfull_from_full**
FIFO_13	Case: wr_en = 0, rd_en = 1, almostempty = 1. 1.Read operation done 2.Checked data_out 3.almostempty gets low 4.empty gets high	Randomization under constraint (label: MY_CON_ONLEY_R) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a coverpoint emoty_cp. Included a coverpoint almostempty_cp. Included a cross coverage almostempty & empty (ack_empty_cross)	Output Checked against golden model Checked by assertion Label: **empty_from_almost**
FIFO_14	Case: wr_en=0, rd_en = 1, almostfull = 1. 1.Read operation done 2.Checked data_out 3.almostfull gets low	Randomization under constraint (label: MY_CON_ONLEY_R) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint almostfull_cp. Included a cross coverage almostfull &rd_en (almostfull_cross)	Output Checked against golden model Checked by assertion Label: **almostfull_inactive**
FIFO_15	Case: wr_en=1, rd_en=1, full = 0, empty = 0. 1.write operation done 2.wr_ack gets high 3.read operation done 4.checked data_out 5.Count will not change	Randomization under constraint (label: MY_CON_BOTH_ACTIVE) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint wr_ack_cp Included a coverpoint data_out_cp Included a cross coverage: wr_ack & wr_en (ack_wr_rd_cross). data_out&wr_en&rd_en (data_out_cross).	Output Checked against golden model Checked by assertion Label: **ack_active,** **count_noChange**
FIFO_16	Case: wr_en=1, rd_en=1,	Randomization under constraint (label:	Included a coverpoint data_out_cp.	Output

	full = 1. 1.write operation ignored 2.wr_ack gets low 3.read operation done 4.checked data_out 5.full gets low 6.almostfull gets high 7.overflow gets high 8.count will decrement.	MY_CON_BOTH_ACTIVE) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint full_cp. Included a coverpoint almostfull_cp. Included a coverpoint overflow_cp. Included a cross coverage: data_out&wr_en&rd_en (data_out_cross). almostfull & full (almostfull_full_cross) overflow& full (full_overflow_cross)	Checked against golden model Checked by assertion Label: **ack_inactive,** **full_inactive,** **almostfull_from_full,** **overflow_active,** **count_dec**
FIFO_17	Case: wr_en=1, rd_en=1, empty = 1. 1.write operation done 2.wr_ack gets high 3.read operation ignored 4.empty gets low 5.almostempty gets high 6.underflow gets high 7.count will increment.	Randomization under constraint (label: MY_CON_BOTH_ACTIVE) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a coverpoint wr_ack_cp. Included a coverpoint emoty_cp. Included a coverpoint almostempty_cp. Included a coverpoint underflow_cp. Included a cross coverage: data_out&wr_en&rd_en (data_out_cross). underflow & emoty (empty_underflow_cross) emoty & almostempty (almostempty_empty_cross) wr_ack&wr_en&rd_en (ack_wr_rd_cross)	Output Checked against golden model Checked by assertion Label: **ack_active,** **empty_inactive,** **almostempty_from_empty,** **underflow_active,** **count_inc**
FIFO_18	Case: wr_en=1, rd_en=1, almostfull = 1. 1.write operation done 2.wr_ack gets high 3.read operation done 4.checked data_out 5. almostfull remain high 6.count remain the same.	Randomization under constraint (label: MY_CON_BOTH_ACTIVE) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a coverpoint wr_ack_cp. Included a cross coverage: data_out&wr_en&rd_en (data_out_cross). wr_ack&wr_en&rd_en (ack_wr_rd_cross)	Output Checked against golden model Checked by assertion Label: **ack_active,** **almostfull_noChange,** **count_noChange**
FIFO_19	Case: wr_en=1, rd_en=1, almostempty = 1. 1.write operation done 2.wr_ack gets high 3.read operation done 4.checked data_out 5. almostempty remain high 6.count remain the same.	Randomization under constraint (label: MY_CON_BOTH_ACTIVE) that constraint wr_en to be inactive and rd_en to be active	Included a coverpoint data_out_cp. Included a coverpoint wr_ack_cp. Included a cross coverage: data_out&wr_en&rd_en (data_out_cross). wr_ack&wr_en&rd_en (ack_wr_rd_cross)	Output Checked against golden model Checked by assertion Label: **ack_active,** **almostempty_noChange,** **count_noChange**

/// Do file

```
vlib work

vlog -coveropt 3 +cover +acc {Codes\package\shared_pkg.sv}
vlog -coveropt 3 +cover +acc {Codes\package\transaction_pkg.sv}
vlog -coveropt 3 +cover +acc {Codes\package\coverage_pkg.sv}
vlog -coveropt 3 +cover +acc {Codes\package\scoreboard_pkg.sv}

vlog -coveropt 3 +cover +acc {Codes\interface\FIFO_interface.SV}

# adding "+define+SIM" to enable assertion if not then don't forget to comment <add wave <assertion label>>
vlog +define+SIM -coveropt 3 +cover +acc {Codes\design\FIFO.sv}
#vlog -coveropt 3 +cover +acc {Codes\design\FIFO.sv}

vlog -coveropt 3 +cover +acc {Codes\reference\FIFO_ref.sv}

vlog -coveropt 3 +cover +acc {Codes\monitor\monitor.sv}
vlog -coveropt 3 +cover +acc {Codes\testbench\testbench.sv}
# this module to test the refrence model
#vlog -coveropt 3 +cover +acc {Codes\testbench\ref_tb.sv}

vlog -coveropt 3 +cover +acc {Codes\top\top.sv}

vsim -voptargs=+acc work.top -cover
add wave *
add wave -position 1 -color white sim:/top/F_if/clk
add wave -position 2 -radix unsigned sim:/top/F_if/rst_n
add wave -position 3 -radix unsigned sim:/top/F_if/wr_en
add wave -position 4 -radix unsigned sim:/top/F_if/rd_en
add wave -position 5 -radix hexadecimal sim:/top/F_if/data_in
add wave -position 6  -color yellow -radix hexadecimal sim:/top/F_if/data_out
add wave -position 7  -color Orchid -radix hexadecimal sim:/top/F_if/data_out_ref
add wave -position 8  -color yellow -radix unsigned sim:/top/F_if/wr_ack
add wave -position 9  -color Orchid -radix unsigned sim:/top/F_if/wr_ack_ref
add wave -position 12 -color yellow -radix unsigned sim:/top/F_if/full
add wave -position 13 -color Orchid -radix unsigned sim:/top/F_if/full_ref
add wave -position 14 -color yellow -radix unsigned sim:/top/F_if/empty
add wave -position 15 -color Orchid -radix unsigned sim:/top/F_if/empty_ref
add wave -position 16 -color yellow -radix unsigned sim:/top/F_if/almostfull
add wave -position 17 -color Orchid -radix unsigned sim:/top/F_if/almostfull_ref
add wave -position 18 -color yellow -radix unsigned sim:/top/F_if/almostempty
add wave -position 19 -color Orchid -radix unsigned sim:/top/F_if/almostempty_ref
add wave -position 20 -color yellow -radix unsigned sim:/top/F_if/overflow
add wave -position 21 -color Orchid -radix unsigned sim:/top/F_if/overflow_ref
add wave -position 22 -color yellow -radix unsigned sim:/top/F_if/underflow
add wave -position 23 -color Orchid -radix unsigned sim:/top/F_if/underflow_ref
add wave -position 24 -radix unsigned sim:/top/dut/count

.vcop Action toggleleafnames
## Assertion
add wave /top/dut/rst_n_assert/assert_full_rst
add wave /top/dut/rst_n_assert/assert_almostfull_rst
add wave /top/dut/rst_n_assert/assert_empty_rst
add wave /top/dut/rst_n_assert/assert_almostempty_rst

add wave /top/dut/full_count
add wave /top/dut/full_from_almost
add wave /top/dut/full_noChange
add wave /top/dut/full_inactive

add wave /top/dut/ack_active
add wave /top/dut/ack_inactive

add wave /top/dut/almostfull_count
add wave /top/dut/almostfull_from_full
add wave /top/dut/almostfull_noChange
add wave /top/dut/almostfull_inactive

add wave /top/dut/overflow_active
add wave /top/dut/overflow_wr_in
add wave /top/dut/overflow_Nfull

add wave /top/dut/almostempty_count
add wave /top/dut/almostempty_noChange
add wave /top/dut/almostempty_from_empty
add wave /top/dut/almostempty_inactive

add wave /top/dut/empty_count
add wave /top/dut/empty_noChaneg
add wave /top/dut/empty_from_almost
add wave /top/dut/empty_inactive

add wave /top/dut/underflow_active
add wave /top/dut/underflow_Nempty
add wave /top/dut/underflow_Nrd

add wave /top/dut/count_inc
add wave /top/dut/count_dec
add wave /top/dut/count_noChange

add wave /top/dut/inc_wr_ptr_assert
add wave /top/dut/inc_rd_ptr_assert
## reset assertion
```

```
add wave /top/dut/rst_n_assert/assert_count_rst
add wave /top/dut/rst_n_assert/assert_wr_ptr_rst
add wave /top/dut/rst_n_assert/assert_rd_ptr_rst
add wave /top/dut/rst_n_assert/assert_wr_akc_rst
add wave /top/dut/rst_n_assert/assert_overflow_rst
add wave /top/dut/rst_n_assert/assert_underflow_rst
add wave /top/dut/rst_n_assert/assert_data_out_rst

run -all
vsim -coverage -vopt work.top -c -do "coverage save -onexit -du FIFO -directive -codeAll cover.ucdb; run -all"
coverage report -detail -cvg -directive -comments -output {Reports/Coverage group report/COV_GRP_FIFO.txt} {}
# if you want to see waveform you have top comment "quit -sim" instruction
quit -sim
vcover report cover.ucdb -details -all -annotate -output {Reports/code coverage report/CODE_COVER_FIFO.txt}
vcover report -html cover.ucdb -output {Reports/code_cover_report_html/.}
```

/// Interface

```
interface FIFO_interface (clk);
import shared_pkg::*;
    input bit clk;

    reg [FIFO_WIDTH-1:0] data_in;
    reg rst_n, wr_en, rd_en;

    reg [FIFO_WIDTH-1:0] data_out;
    reg wr_ack, overflow;
    reg full, empty, almostfull, almostempty, underflow;

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

    modport DUT (
        input clk,
        input data_in, rst_n, wr_en, rd_en,

        output data_out,
        output wr_ack, overflow,
        output full, empty, almostfull, almostempty, underflow
    );

    modport REF (
        input clk,
        input data_in, rst_n, wr_en, rd_en,

        output data_out_ref,
        output wr_ack_ref, overflow_ref,
        output full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref
    );

    modport TEST (
        input clk,

        input data_out,
        input wr_ack, overflow,
        input full, empty, almostfull, almostempty, underflow,

        input data_out_ref,
        input wr_ack_ref, overflow_ref,
        input full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref,

        output data_in, rst_n, wr_en, rd_en
    );

    modport MONITOR (
        input clk,
        input data_in, rst_n, wr_en, rd_en,

        input data_out,
        input wr_ack, overflow,
        input full, empty, almostfull, almostempty, underflow,

        input data_out_ref,
        input wr_ack_ref, overflow_ref,
        input full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref
    );
endinterface //FIRO_interface
```

/// FIFO design after fixing

```
import shared_pkg::*;
module FIFO(FIFO_interface.DUT F_if);

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

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

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

assign F_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign F_if.empty = (count == 0 && F_if.rst_n)? 1 : 0; // FIX

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

/// ** Sequence ** //

```
`ifdef SIM
// Sequences
sequence same_seq;
    (F_if.rd_en && F_if.wr_en && !F_if.empty && !F_if.full);
endsequence
sequence almE_wr_Nrd; // almostempty and write but NOT read
    (F_if.almostempty && F_if.wr_en && !F_if.rd_en);
endsequence
sequence F_wr; // full and write
    (F_if.full && F_if.wr_en);
endsequence
sequence E_wr; // empty and write
    (F_if.empty && F_if.wr_en);
endsequence
sequence E_rd; // empty and write
    (F_if.empty && F_if.rd_en);
endsequence
sequence NF_wr; // NOT full and write
    (!F_if.full && F_if.wr_en);
endsequence
sequence F_rd; // full and read
    (F_if.full && F_if.rd_en);
endsequence
sequence almE_Nwr_rd; // alostempty and read but NOT write
    (F_if.almostempty && !F_if.wr_en && F_if.rd_en);
endsequence
sequence almF_Nwr_rd; // alostfull and read but NOT write
    (F_if.almostempty && !F_if.wr_en && F_if.rd_en);
endsequence
sequence almF_wr_Nrd; // almostfull and write but NOT read
    (F_if.wr_en && F_if.almostfull && !F_if.rd_en);
endsequence
sequence w_ptr_seq;
    (F_if.wr_en && count < FIFO_DEPTH);
endsequence
sequence inc_rd_ptr_rslt;
    ($past(rd_ptr)+1 == rd_ptr)||($past(rd_ptr)==FIFO_DEPTH-1);
    // ($past(rd_ptr)==FIFO_DEPTH-1) => becouse for questa when ptr.past = 7 then ptr should be zero but questa will treats it as 8 NOT zero
    // when $past(rd_ptr) = 7 then $past(rd_ptr) + 1 = 0 (3-bit)
    // for questa $past(rd_ptr) + 1 = 8
endsequence
```

/// ** Assertion Internal signal and reset assertion **//

```
// ASSERTION
// Internal  signal assertion
count_inc: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (F_if.wr_en && !F_if.rd_en && !F_if.full) |=> ($past(count)+1 == count));
count_dec: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (!F_if.wr_en && F_if.rd_en && !F_if.empty)|=> ($past(count)-1 == count));
count_noChange: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(count) == count));



    inc_wr_ptr_assert:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n)  w_ptr_seq |=> ($past(wr_ptr)+1 == wr_ptr) || ($past(wr_ptr) ==
FIFO_DEPTH-1) );
    inc_rd_ptr_assert:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (F_if.rd_en && count != 0) |=> inc_rd_ptr_rslt );


    always_comb begin : rst_n_assert
        if (!F_if.rst_n) begin
            assert_count_rst:  assert final (count == 0);
            assert_wr_ptr_rst: assert final (wr_ptr == 0);
            assert_rd_ptr_rst: assert final (rd_ptr == 0);

            assert_wr_akc_rst:    assert final (F_if.wr_ack == 0);
            assert_overflow_rst:  assert final (F_if.overflow == 0);
            assert_underflow_rst: assert final (F_if.underflow == 0);
            assert_data_out_rst:  assert final (F_if.data_out == 0);



            assert_full_rst:        assert final (F_if.full == 0);
            assert_almostfull_rst:  assert final (F_if.almostfull == 0);
            assert_empty_rst:       assert final (F_if.empty == 0);
            assert_almostempty_rst: assert final (F_if.almostempty == 0);
        end
    end
```

/// ** Global signal Assertion **//

```
// Global signal assertion
// full
full_count:      assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count >= FIFO_DEPTH) |->  F_if.full);
full_from_almost:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almF_wr_Nrd |=> F_if.full);
full_noChange:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq    |=> !F_if.full);
full_inactive:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_rd        |=> !F_if.full);

// wr_ack
ack_active:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) NF_wr |=> F_if.wr_ack);
ack_inactive: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_wr  |=> !F_if.wr_ack);

// almostfull
almostfull_count:    assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==FIFO_DEPTH-1) |-> F_if.almostfull);
almostfull_from_full:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_rd  |=> ($fell(F_if.full) && $rose(F_if.almostfull)));
almostfull_noChange: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> $past(F_if.almostfull) == F_if.almostfull);
almostfull_inactive:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almF_Nwr_rd |=> !F_if.almostfull );

// overflow
overflow_active:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_wr |=> F_if.overflow);
overflow_wr_in: assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.overflow) && !F_if.wr_en) |=> !F_if.overflow);
overflow_Nfull:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.overflow) && !F_if.full)  |=> !F_if.overflow);

// almostempty
almostempty_count:     assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==1) |-> F_if.almostempty );
almostempty_noChange:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(F_if.almostempty) == F_if.almostempty));
almostempty_from_empty:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_wr |=> ($fell(F_if.empty) && $rose(F_if.almostempty)));
almostempty_inactive:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almE_wr_Nrd |=> !F_if.almostempty );

// empty
empty_count:      assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==ZERO) |-> F_if.empty );
empty_noChaneg:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(F_if.empty) == F_if.empty) );
empty_from_almost:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almE_Nwr_rd |=> F_if.empty);
empty_inactive:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_wr |=> !F_if.empty);

// underflow
underflow_active:  assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_rd |=> F_if.underflow);
underflow_Nempty:assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.underflow) && !F_if.empty)  |=> !F_if.underflow);
underflow_Nrd:   assert property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.underflow) && !F_if.rd_en)  |=> !F_if.underflow);
`endif
endmodule
```

Coverage Directives

```
// Coverage
// Internal  signal assertion
count_inc_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (F_if.wr_en && !F_if.rd_en && !F_if.full) |=> ($past(count)+1 == count));
count_dec_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (!F_if.wr_en && F_if.rd_en && !F_if.empty)|=> ($past(count)-1 == count));
count_noChange_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(count) == count));



inc_wr_ptr_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n)  w_ptr_seq |=> ($past(wr_ptr)+1 == wr_ptr) || ($past(wr_ptr) ==
FIFO_DEPTH-1) );
inc_rd_ptr_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (F_if.rd_en && count != 0) |=> inc_rd_ptr_rslt );


    always_comb begin : rst_n_cover
        if (!F_if.rst_n) begin
            cover_count_rst:  cover final (count == 0);
            cover_wr_ptr_rst: cover final (wr_ptr == 0);
            cover_rd_ptr_rst: cover final (rd_ptr == 0);

            cover_wr_akc_rst:    cover final (F_if.wr_ack == 0);
            cover_overflow_rst:  cover final (F_if.overflow == 0);
            cover_underflow_rst: cover final (F_if.underflow == 0);
            cover_data_out_rst:  cover final (F_if.data_out == 0);



            cover_full_rst:        cover final (F_if.full == 0);
            cover_almostfull_rst:  cover final (F_if.almostfull == 0);
            cover_empty_rst:       cover final (F_if.empty == 0);
            cover_almostempty_rst: cover final (F_if.almostempty == 0);
        end
    end

// Global signal cover
// full
full_count_cover:    cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count >= FIFO_DEPTH) |->  F_if.full);
full_from_almost_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almF_wr_Nrd |=> F_if.full);
```

```
full_noChange_cover:     cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq    |=> !F_if.full);
full_inactive_cover:     cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_rd     |=> !F_if.full);

// wr_ack
ack_active_cover:   cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) NF_wr |=> F_if.wr_ack);
ack_inactive_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_wr  |=> !F_if.wr_ack);

// almostfull
almostfull_count_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==FIFO_DEPTH-1) |-> F_if.almostfull);
almostfull_from_full_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_rd  |=> ($fell(F_if.full) && $rose(F_if.almostfull)));
almostfull_noChange_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> $past(F_if.almostfull) == F_if.almostfull);
almostfull_inactive_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almF_Nwr_rd |=> !F_if.almostfull );

// overflow
overflow_active_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) F_wr |=> F_if.overflow);
overflow_wr_in_cover: cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.overflow) && !F_if.wr_en) |=> !F_if.overflow);
overflow_Nfull_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.overflow) && !F_if.full)  |=> !F_if.overflow);

// almostempty
almostempty_count_cover:       cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==1) |-> F_if.almostempty );
almostempty_noChange_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(F_if.almostempty) == F_if.almostempty));
almostempty_from_empty_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_wr |=> ($fell(F_if.empty) && $rose(F_if.almostempty)));
almostempty_inactive_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almE_wr_Nrd |=> !F_if.almostempty );

// empty
empty_count_cover:    cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) (count==ZERO) |-> F_if.empty );
empty_noChaneg_cover:   cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) same_seq |=> ($past(F_if.empty) == F_if.empty) );
empty_from_almost_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) almE_Nwr_rd |=> F_if.empty);
empty_inactive_cover:   cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_wr |=> !F_if.empty);

// underflow
underflow_active_cover:  cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) E_rd |=> F_if.underflow);
underflow_Nempty_cover:cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.underflow) && !F_if.empty)  |=> !F_if.underflow);
underflow_Nrd_cover:   cover property (@(posedge F_if.clk) disable iff(!F_if.rst_n) ($past(F_if.underflow) && !F_if.rd_en)  |=> !F_if.underflow);
`endif
endmodule
```

/// shared_pkg

```
package shared_pkg;
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;
        parameter ACTIVE = 1;
        parameter INACTIVE = 0;

        int WR_EN_ON_DIST = 70;
        int RD_EN_ON_DIST = 30;
        int RESET_ACTIVE = 5;

        parameter ZERO = 0;
        parameter MAX = (2**FIFO_DEPTH) - 1;
        reg [FIFO_WIDTH-1:0] one_bit_high [16] = '{16'h1, 16'h2, 16'h4, 16'h8, 16'h10, 16'h20, 16'h40, 16'h80,
                                                   16'h100, 16'h200, 16'h400, 16'h800, 16'h1000, 16'h2000, 16'h4000, 16'h8000};
        //bit [FIFO_WIDTH-1:0] one_bit_high [16] = '{16'b0000_0000_0000_0001,
        //                                           16'b0000_0000_0000_0010,
        //                                           16'b0000_0000_0000_0100,
        //                                           16'b0000_0000_0000_1000,
        //                                           16'b0000_0000_0001_0000,
        //                                           16'b0000_0000_0010_0000,
        //                                           16'b0000_0000_0100_0000,
        //                                           16'b0000_0000_1000_0000,
        //                                           16'b0000_0001_0000_0000,
        //                                           16'b0000_0010_0000_0000,
        //                                           16'b0000_0100_0000_0000,
        //                                           16'b0000_1000_0000_0000,
        //                                           16'b0001_0000_0000_0000,
        //                                           16'b0010_0000_0000_0000,
        //                                           16'b0100_0000_0000_0000,
        //                                           16'b1000_0000_0000_0000}
        bit test_finished;
        int error_counter = 0;
        int correct_counter = 0;
        int inc_correct_counter = 0;

        int LOOP0 = 1_00;
        int LOOP1 = 1_000;
        int LOOP2 = 10_000;
        int LOOP3 = 100_000;
endpackage
```

/// transaction_pkg

```
package transaction_pkg;
import shared_pkg::*;

class FIFO_transaction;
// input
    rand bit [FIFO_WIDTH-1:0] data_in;
    rand bit rst_n, wr_en, rd_en;
    // bit clk;

// output
    bit [FIFO_WIDTH-1:0] data_out;
    bit wr_ack, overflow;
    bit full, empty, almostfull, almostempty, underflow;

bit toggle;
// Costraint Block
    // requirement constraint
    constraint CON_RESET {
        rst_n dist {1:=100-RESET_ACTIVE, 0:=RESET_ACTIVE};
    }
    constraint CON_W_R_DIST {
        wr_en dist {ACTIVE:=WR_EN_ON_DIST, INACTIVE:=100-WR_EN_ON_DIST};
        rd_en dist {ACTIVE:=RD_EN_ON_DIST, INACTIVE:=100-RD_EN_ON_DIST};
    }

    // constraint to assert onley write with constraint one bit high to data_in
    constraint MY_CON_ONLEY_W {
        wr_en == ACTIVE;
        rd_en == INACTIVE;
    }

    // constraint to assert onley read with constraint zero or MAX ro data_in
    constraint MY_CON_ONLEY_R {
        wr_en == INACTIVE;
        rd_en == ACTIVE;
    }

    // constraint to Toggle write and read enable
    constraint MY_CON_OPPSITE {
        wr_en == ~rd_en;
    }

    // constraint to make write and read active at the same time
    constraint MY_CON_BOTH_ACTIVE {
        wr_en == ACTIVE;
        rd_en == ACTIVE;
    }

    constraint CON_DATA_OUT_ONE_BIT{
        $countones(data_in) == 1;
    }
    constraint CON_DATA_OUT_M_Z{
        ((data_in == ZERO) || (data_in == MAX));
    }

// Mythods
    int wr_old = 0, rd_old = 0;
    function void pre_randomize();
        wr_old = wr_en;
        rd_old = rd_en;
    endfunction

    // I want to assert rst for the first 5 clk cycle
    int first_rst = 0;
    function void post_randomize();
        if (first_rst <= 5) begin
            first_rst++;
            rst_n = 0;
        end
        if (toggle) begin
            wr_en = ~wr_old;
            rd_en = ~rd_old;
        end
    endfunction
```

```
    function void assigned_values(
        //  output
        output bit [FIFO_WIDTH-1:0] data_in,
        output bit rst_n, wr_en, rd_en,

        // input
        input bit [FIFO_WIDTH-1:0] data_out,
        input bit wr_ack, overflow,
        input bit full, empty, almostfull, almostempty, underflow
    );
        // output
        data_in = this.data_in;
        rst_n   = this.rst_n;
        wr_en   = this.wr_en;
        rd_en   = this.rd_en;

        // input
        this.data_out    = data_out;
        this.wr_ack      = wr_ack;
        this.overflow    = overflow;
        this.full        = full;
        this.empty       = empty;
        this.almostfull  = almostfull;
        this.almostempty = almostempty;
        this.underflow   = underflow;
    endfunction

    function new();
    endfunction //new()
endclass //FIFO_transaction
endpackage
```

/// scoreboard_pkg

```
package scoreboard_pkg;
import transaction_pkg::*;
import shared_pkg::*;

class FIFO_scoreboard;
// input
    bit [FIFO_WIDTH-1:0] data_in;
    bit rst_n, wr_en, rd_en;
// output refrence
    bit [FIFO_WIDTH-1:0] data_out_ref;
    bit wr_ack_ref, overflow_ref;
    bit full_ref, empty_ref, almostfull_ref;
    bit almostempty_ref, underflow_ref;

    function void reference_model(
        input bit [FIFO_WIDTH-1:0] data_out_,
        input bit wr_ack_, overflow_, full_, empty_,
        input bit almostfull_, almostempty_, underflow_
    );
        data_out_ref     = data_out_;
        wr_ack_ref       = wr_ack_;
        overflow_ref     = overflow_;
        full_ref         = full_;
        empty_ref        = empty_;
        almostfull_ref   = almostfull_;
        almostempty_ref  = almostempty_;
        underflow_ref    = underflow_;
    endfunction // reference_model()

    function void check_data (input FIFO_transaction F_txn);
        if (data_out_ref != F_txn.data_out) begin
            $error("%t: Error - data_out_ref = %0d, data_out = %0d",$time, data_out_ref, F_txn.data_out);
            inc_correct_counter++;
        end
        display_errors(wr_ack_ref, F_txn.wr_ack, "wr_ack");
        display_errors(full_ref, F_txn.full, "full");
        display_errors(empty_ref, F_txn.empty, "empty");
        display_errors(almostfull_ref, F_txn.almostfull, "almostfull");
        display_errors(almostempty_ref, F_txn.almostempty, "almostempty");
        display_errors(overflow_ref, F_txn.overflow, "overflow");
        display_errors(underflow_ref, F_txn.underflow, "underflow");
        if (inc_correct_counter == 7) begin
            correct_counter++;
            inc_correct_counter = 0;
        end
    endfunction //check_data ()

    function void display_errors (input bit gld, dut, input string x);
        if (gld!=dut) begin
            $error("%s = %0d, %s_ref = %0d", x, dut, x, gld);
            error_counter++;
        end else
            inc_correct_counter++;
    endfunction

    function new();
    endfunction //new()
endclass //FIFO_scoreboard
endpackage
```

/// coverage_pkg

```
package coverage_pkg;
import transaction_pkg::*;
import shared_pkg::*;

class FIFO_coverage;
FIFO_transaction F_cvg_txn = new();

covergroup CVG;
// rst_n coverage
    rst_cp: coverpoint F_cvg_txn.rst_n{
            bins active = {0};
            bins inactive = {1};
            bins inactive_to_active = (1 => 0);
            bins active_to_inactive = (0 => 1);
    }

// write and read enable signal coverpoint
    wr_en_cp:        coverpoint F_cvg_txn.wr_en{
        bins active = {1};
        bins inactive = {0};
    }
    rd_en_cp:        coverpoint F_cvg_txn.rd_en{
        bins active = {1};
        bins inactive = {0};
    }

// data_out bus coverpoint
    data_out_cp:     coverpoint F_cvg_txn.data_out{
        bins one_bit_H[] = one_bit_high;
        bins zero = {ZERO};
        bins max = {MAX};
        bins others = default;
    }

// outputs signals coverpoint
    wr_ack_cp:       coverpoint F_cvg_txn.wr_ack{
        bins active = {1};
        bins inactive = {0};
        bins inactive_to_active = (0 => 1);
        bins active_to_inactive = (1 => 0);
    }
    full_cp:         coverpoint F_cvg_txn.full{
        bins active = {1};
        bins inactive = {0};
        bins active_to_inactive = (1 => 0);
        bins inactive_to_active = (0 => 1);
    }
    empty_cp:        coverpoint F_cvg_txn.empty{
        bins active = {1};
        bins inactive = {0};
        bins active_to_inactive = (1 => 0);
        bins inactive_to_active = (0 => 1);
    }
    almostfull_cp:   coverpoint F_cvg_txn.almostfull{
        bins active = {1};
        bins inactive = {0};
        bins active_to_inactive = (1 => 0);
        bins inactive_to_active = (0 => 1);
    }
    almostempty_cp:  coverpoint F_cvg_txn.almostempty{
        bins active = {1};
        bins inactive = {0};
        bins active_to_inactive = (1 => 0);
        bins inactive_to_active = (0 => 1);
    }
    underflow_cp:    coverpoint F_cvg_txn.underflow{
        bins active = {1};
        bins inactive = {0};
    }
    overflow_cp:     coverpoint F_cvg_txn.overflow{
        bins active = {1};
        bins inactive = {0};
    }

// Cross coverage
// A -> refear to Active
// I -> refear to Inactive

// wr_ack signal
    // reset
    ack_rst_cross: cross rst_cp, wr_ack_cp {
        bins rst_ack = binsof(rst_cp.active) && binsof(wr_ack_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

    // wr_en and rd_en ** requirement **
    ack_wr_rd_cross: cross wr_ack_cp, wr_en_cp, rd_en_cp;

    // full and wr_en
    // crossing wr_ack with full when wr_ack is active and full is active
    // crossing wr_ack with full when full rose and wr_ack fell
    ack_full_wr_cross: cross wr_ack_cp, wr_en_cp, full_cp{
```

```
        bins A_ack_A_wr_I_full = binsof(wr_ack_cp.active)
                                 && binsof(wr_en_cp.active)
                                 && binsof(full_cp.inactive);
        bins ack_full = binsof(wr_ack_cp.active) && binsof(full_cp.inactive);
        bins A_ack_A_full_trans = binsof(wr_ack_cp.inactive_to_active) && binsof(full_cp.inactive_to_active);
        option.cross_auto_bin_max = 0;
    }

    // empty
    ack_empty_cross: cross empty_cp, wr_ack_cp {
        bins emptyTRANS_ackAC = binsof(wr_ack_cp.active) && binsof(empty_cp.active_to_inactive);
        option.cross_auto_bin_max = 0;
    }

    // almostempty
    ack_almostempty_cross: cross almostempty_cp, wr_ack_cp {
        bins Aack_Ialmostempty = binsof(wr_ack_cp.active) && binsof(almostempty_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

// full signal
    // rst transaction
    rst_full_cross: cross full_cp, rst_cp{
        bins trans_rst_full = binsof(full_cp.active_to_inactive) && binsof(rst_cp.inactive_to_active);
        option.cross_auto_bin_max = 0;
    }
    // wr_en and rd_en ** requirement **
    full_cross: cross full_cp, wr_en_cp, rd_en_cp;

    // almostfull transaction
    // crossing to detect when almostfull trans from active to inactive and full from inactive to active
    // and oppesite operation
    almostfull_full_cross: cross almostfull_cp, full_cp{
        bins trans_almostfull_to_full = binsof(almostfull_cp.active_to_inactive) && binsof(full_cp.inactive_to_active);
        bins trans_full_to_almostfull = binsof(almostfull_cp.inactive_to_active) && binsof(full_cp.active_to_inactive);
        option.cross_auto_bin_max = 0;
    }

    // overflow
    // crossing to detect when overflow and full both active
    full_overflow_cross:   cross overflow_cp, full_cp{
        bins overflow_full = binsof(overflow_cp.active) && binsof(full_cp.active);
        option.cross_auto_bin_max = 0;
    }

// empty signal
    // rst
    rst_empty_cross: cross empty_cp, rst_cp {
        bins rst_empty = binsof(empty_cp.active) && binsof(rst_cp.active);
        option.cross_auto_bin_max = 0;
    }

    // almostempty trans
    // crossing to detect when almostempty trans from active to inactive and empty from inactive to active
    // and oppesite operation
    almostempty_empty_cross: cross almostempty_cp, empty_cp{
        bins trans_almostempty_to_empty = binsof(almostempty_cp.active_to_inactive) && binsof(empty_cp.inactive_to_active);
        bins trans_empty_to_almostempty = binsof(almostempty_cp.inactive_to_active) && binsof(empty_cp.active_to_inactive);
        option.cross_auto_bin_max = 0;
    }

    // rd_en and wr_en
    empty_cross: cross empty_cp, wr_en_cp, rd_en_cp;

    // underflow
    // crossing to detect when underflow and empty both active
    empty_underflow_cross: cross underflow_cp, empty_cp{
        bins underflow_empty = binsof(underflow_cp.active) && binsof(empty_cp.active);
        option.cross_auto_bin_max = 0;
    }

// overflow signal
    // rst
    rst_overflow_cross: cross rst_cp, overflow_cp {
        bins rst_ack = binsof(rst_cp.active) && binsof(overflow_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

    // wr_en
    // crossing to detect when overflow and write enabe both active
    wr_overflow_cross: cross wr_en_cp, rd_en_cp, overflow_cp{
        bins both_high = binsof(wr_en_cp.active) && binsof(overflow_cp.active);
    }

// underflow signal
    // rst
    rst_underflow_cross: cross rst_cp, underflow_cp {
        bins rst_ack = binsof(rst_cp.active) && binsof(underflow_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

    // rd_en
    // crossing to detect when underflow and read enabe both active
    rd_underflow_cross: cross wr_en_cp, rd_en_cp, underflow_cp{
        bins both_high = binsof(rd_en_cp.active) && binsof(underflow_cp.active);
    }
```

```
// almostempty signal
    // rst
    rst_almostempty_cross: cross rst_cp, almostempty_cp{
        bins rst_almostempty = binsof(rst_cp.active) && binsof(almostempty_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

    // rd_en and wr_en
    almostempty_cross:  cross wr_en_cp, rd_en_cp, almostempty_cp{
        bins both_Wr_high = binsof(wr_en_cp.active) && binsof(almostempty_cp.active);
        bins INalmostEmp_Awr = binsof(wr_en_cp.active) && binsof(almostempty_cp.inactive);
        bins both_Re_high = binsof(rd_en_cp.active) && binsof(almostempty_cp.active);
        bins INalmostEmp_Ard = binsof(rd_en_cp.active) && binsof(almostempty_cp.inactive);
    }

// almostfull signal
    // rst
    rst_almostfull_cross: cross rst_cp, almostfull_cp{
        bins rst_almostfull = binsof(rst_cp.active) && binsof(almostfull_cp.inactive);
        option.cross_auto_bin_max = 0;
    }

    // wr_en and rd_en
    almostfull_cross:   cross wr_en_cp, rd_en_cp, almostfull_cp{
        bins both_Wr_high = binsof(wr_en_cp.active) && binsof(almostfull_cp.active);
        bins INalmostfull_Awr = binsof(wr_en_cp.active) && binsof(almostfull_cp.inactive);
        bins both_Re_high = binsof(rd_en_cp.active) && binsof(almostfull_cp.active);
        bins INalmostfull_Ard = binsof(rd_en_cp.active) && binsof(almostfull_cp.inactive);
    }

// data_out bus
    // reset
    rst_data_out_cross: cross rst_cp, data_out_cp {
        bins rst_data = binsof(rst_cp.active) && binsof(data_out_cp.zero);
        option.cross_auto_bin_max = 0;
    }

    // rd_en and wr_en ** requirement **
    data_out_cross:     cross data_out_cp, wr_en_cp, rd_en_cp;

// Crossing onley read and write
    rd_wr_cross:           cross rd_en_cp, wr_en_cp;
endgroup
// methods
    function new();
        CVG = new();
    endfunction //new()

    function void sample_data(FIFO_transaction F_txn);
        F_cvg_txn = F_txn; // shallow copy
        CVG.sample();
    endfunction //sample_data()
endclass //FIFO_coverage
endpackage
```

/// golden model design

```
import shared_pkg::*;
module FIFO_ref(FIFO_interface.REF F_if);
    reg [FIFO_WIDTH-1:0] fifoo_q [$];

    // Write
    always @(posedge F_if.clk or negedge F_if.rst_n) begin
        if (!F_if.rst_n) begin
            fifoo_q.delete();
            F_if.wr_ack_ref <= 0;
            F_if.overflow_ref <= 0;
        end
        else if (F_if.wr_en && !F_if.full_ref) begin
            fifoo_q.push_back(F_if.data_in);
            F_if.wr_ack_ref <= 1;
            F_if.overflow_ref <= 0;
        end
        else begin
            F_if.wr_ack_ref <= 0;
            if (F_if.full_ref && F_if.wr_en)
                F_if.overflow_ref <= 1;
            else
                F_if.overflow_ref <= 0;
        end
    end

    // Read
    always @(posedge F_if.clk or negedge F_if.rst_n) begin
        if (!F_if.rst_n) begin
            fifoo_q.delete();
            F_if.underflow_ref <= 0;
            F_if.data_out_ref <= 0;
        end
        else if (F_if.rd_en && !F_if.empty_ref) begin
            F_if.data_out_ref <= fifoo_q.pop_front();
            F_if.underflow_ref <= 0;
        end
        else begin
            if (F_if.empty_ref && F_if.rd_en)
                F_if.underflow_ref <= 1;
            else
                F_if.underflow_ref <= 0;
        end
    end

    assign F_if.almostfull_ref = (fifoo_q.size() == FIFO_DEPTH-1)? 1:0;
    assign F_if.full_ref = (fifoo_q.size() >= FIFO_DEPTH)? 1:0;

    assign F_if.empty_ref = (fifoo_q.size() == 0 && F_if.rst_n)? 1:0;
    assign F_if.almostempty_ref = (fifoo_q.size() == 1)? 1:0;
endmodule
```

/// monitor

```
import shared_pkg::*;
import scoreboard_pkg::*;
import transaction_pkg::*;
import coverage_pkg::*;

module monitor (FIFO_interface.MONITOR F_if);
    FIFO_transaction tr;
    FIFO_scoreboard sb;
    FIFO_coverage cov;

initial begin
    tr = new();
    sb = new();
    cov = new();
    forever begin

        @(negedge F_if.clk) begin
            // input
            tr.data_in = F_if.data_in;
            tr.rst_n   = F_if.rst_n;
            tr.wr_en   = F_if.wr_en;
            tr.rd_en   = F_if.rd_en;

            // output
            tr.data_out    = F_if.data_out;
            tr.wr_ack      = F_if.wr_ack;
            tr.overflow    = F_if.overflow;
            tr.full        = F_if.full;
            tr.empty       = F_if.empty;
            tr.almostfull  = F_if.almostfull;
            tr.almostempty = F_if.almostempty;
            tr.underflow   = F_if.underflow;
        end

        fork
            // Process 1
            begin
                sb.reference_model(
                    F_if.data_out_ref, F_if.wr_ack_ref, F_if.overflow_ref,
                    F_if.full_ref, F_if.empty_ref, F_if.almostfull_ref,
                    F_if.almostempty_ref, F_if.underflow_ref
                );
                sb.check_data(tr);
            end

            // Process 2
            begin
                cov.sample_data(tr);
            end
        join
        if (test_finished) begin
            $display("");
            $display("************ ************************************** ************");
            $display("************ ************** summary: ************** ************");
            $display("************ ************************************** ************");
            $display("************ ************************************** ************");
            $display("********** error_counter = %0d, correct_counter = %0d **********",error_counter, correct_counter);
            $stop;
        end
    end
end

endmodule
```

/// testbench

```
import shared_pkg::*;
import transaction_pkg::*;

module testbench (FIFO_interface.TEST F_if);
FIFO_transaction tr = new();

initial begin
// All constraint
// tr.CONSTRAINT_NAME.constraint_mode(0); => Turn OFF specific constraint
// tr.CONSTRAINT_NAME.constraint_mode(1); => Turn ON specific constraint

// disable all constraint
tr.constraint_mode(0);

RESET_ACTIVE = 10;
// turn on reset constraint in all tb cases
tr.CON_RESET.constraint_mode(1);



// turn ON requirement constraint
tr.CON_W_R_DIST.constraint_mode(1);
$display("%0t: Turn ON CON_W_R_DIST",$time);

tr.CON_DATA_OUT_ONE_BIT.constraint_mode(1); // ON constraint data_in to be only one bit high
$display("%0t: Turn ON CON_DATA_OUT_ONE_BIT",$time);
    repeat(LOOP2) randomization;
$display("%0t: Turn OFF CON_DATA_OUT_ONE_BIT",$time);
tr.CON_DATA_OUT_ONE_BIT.constraint_mode(0); // OFF constraint data_in to be only one bit high

tr.CON_DATA_OUT_M_Z.constraint_mode(1); // ON constraint data_in to be MAX or ZERO
$display("%0t: Turn ON CON_DATA_OUT_M_Z ",$time);
    repeat(LOOP1) randomization;
$display("%0t: Turn OFF CON_DATA_OUT_M_Z ",$time);
tr.CON_DATA_OUT_M_Z.constraint_mode(0); // OFF constraint data_in to be MAX or ZERO

$display("%0t: Turn OFF CON_W_R_DIST",$time);
tr.CON_W_R_DIST.constraint_mode(0); // turn OFF requirement constraint

// turn on MY_CON_ONLEY_W constraint => wr_en always active, rd_en always inactive
tr.MY_CON_ONLEY_W.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_W",$time);
    repeat(FIFO_DEPTH) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_W",$time);
tr.MY_CON_ONLEY_W.constraint_mode(0);

// reset
reset;

// turn on MY_CON_ONLEY_W constraint => wr_en always active, rd_en always inactive
tr.MY_CON_ONLEY_W.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_W",$time);
    repeat(FIFO_DEPTH * 4) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_W",$time);
tr.MY_CON_ONLEY_W.constraint_mode(0);

// turn on MY_CON_ONLEY_R constraint => rd_en always active, wr_en always inactive
tr.MY_CON_ONLEY_R.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_R",$time);
    repeat(FIFO_DEPTH-1) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_R",$time);
tr.MY_CON_ONLEY_R.constraint_mode(0);

// turn on MY_CON_BOTH_ACTIVE constraint => rd_en always active, wr_en always active
tr.MY_CON_BOTH_ACTIVE.constraint_mode(1);
$display("%0t: Turn ON MY_CON_BOTH_ACTIVE",$time);
    repeat(LOOP1) randomization;
$display("%0t: Turn OFF MY_CON_BOTH_ACTIVE",$time);
tr.MY_CON_BOTH_ACTIVE.constraint_mode(0);

// turn on MY_CON_OPPSITE constraint => rd_en always invert of wr_en
tr.MY_CON_OPPSITE.constraint_mode(1);
$display("%0t: Turn ON MY_CON_OPPSITE",$time);
    repeat(LOOP1) randomization;
$display("%0t: Turn OFF MY_CON_OPPSITE",$time);
tr.MY_CON_OPPSITE.constraint_mode(0);

// TURN on toggle => wr_en and rd_en toggle every clk cycle
tr.toggle = 1;
$display("%0t: Turn ON Toggle",$time);
repeat(LOOP1) randomization;
$display("%0t: Turn OFF Toggle",$time);
tr.toggle = 0;

// turn on MY_CON_ONLEY_R constraint => rd_en always active, wr_en always inactive
tr.MY_CON_ONLEY_R.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_R",$time);
    repeat(FIFO_DEPTH * 5) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_R",$time);
tr.MY_CON_ONLEY_R.constraint_mode(0);

// turn on MY_CON_ONLEY_W constraint => wr_en always active, rd_en always inactive
```

```
tr.MY_CON_ONLEY_W.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_W",$time);
    repeat(FIFO_DEPTH - 1) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_W",$time);
tr.MY_CON_ONLEY_W.constraint_mode(0);

// turn on MY_CON_BOTH_ACTIVE constraint => rd_en always active, wr_en always active
tr.MY_CON_BOTH_ACTIVE.constraint_mode(1);
$display("%0t: Turn ON MY_CON_BOTH_ACTIVE",$time);
    repeat(LOOP1) randomization;
$display("%0t: Turn OFF MY_CON_BOTH_ACTIVE",$time);
tr.MY_CON_BOTH_ACTIVE.constraint_mode(0);

// turn on MY_CON_ONLEY_W constraint => wr_en always active, rd_en always inactive
tr.MY_CON_ONLEY_W.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_W",$time);
    repeat(FIFO_DEPTH) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_W",$time);
tr.MY_CON_ONLEY_W.constraint_mode(0);

// turn on MY_CON_ONLEY_R constraint => rd_en always active, wr_en always inactive
tr.MY_CON_ONLEY_R.constraint_mode(1);
$display("%0t: Turn ON MY_CON_ONLEY_R",$time);
    repeat(FIFO_DEPTH - 1) randomization;
$display("%0t: Turn OFF MY_CON_ONLEY_R",$time);
tr.MY_CON_ONLEY_R.constraint_mode(0);

// turn on MY_CON_BOTH_ACTIVE constraint => rd_en always active, wr_en always active
tr.MY_CON_BOTH_ACTIVE.constraint_mode(1);
$display("%0t: Turn ON MY_CON_BOTH_ACTIVE",$time);
    repeat(LOOP0) randomization;
$display("%0t: Turn OFF MY_CON_BOTH_ACTIVE",$time);
tr.MY_CON_BOTH_ACTIVE.constraint_mode(0);

// reset
reset;

tr.constraint_mode(0);
WR_EN_ON_DIST = 30; // change probablty of wr_en get high
RD_EN_ON_DIST = 70; // change probablty of rd_en get high
tr.CON_W_R_DIST.constraint_mode(1);
$display("%0t: Turn ON CON_W_R_DIST",$time);
    repeat(LOOP2) randomization;
$display("%0t: Turn OFF CON_W_R_DIST",$time);
tr.CON_W_R_DIST.constraint_mode(0);

tr.constraint_mode(0);
// reset
reset;
RESET_ACTIVE = 10; // change probablty of rst_n get active
// turn of rst constraint
tr.CON_RESET.constraint_mode(1);
repeat(LOOP3) randomization;

// Test Finished
test_finished = 1;
@(negedge F_if.clk);
//$stop;
end
task randomization;
        assert(tr.randomize());
        assigned_tr_itf;
        @(negedge F_if.clk);
endtask //
task assigned_tr_itf;
    F_if.data_in = tr.data_in;
    F_if.rst_n   = tr.rst_n;
    F_if.wr_en   = tr.wr_en;
    F_if.rd_en   = tr.rd_en;
endtask //

task reset;
    F_if.rst_n = 0;
    tr.rst_n = F_if.rst_n;
    repeat(3) @(negedge F_if.clk);
    F_if.rst_n = 1;
    tr.rst_n = F_if.rst_n;
endtask //
endmodule
```

/// top

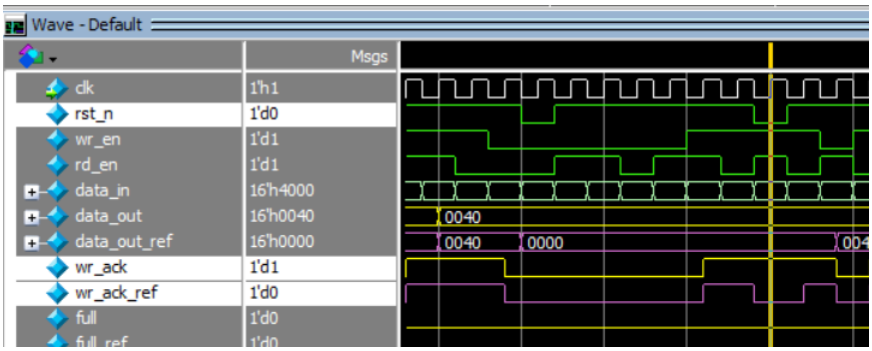
```
module top ();
    bit clk;
    initial begin
        forever #1 clk = ~clk;
    end

    FIFO_interface F_if(clk);

    FIFO dut(F_if);
    FIFO_ref gld(F_if);
    testbench tb(F_if);
    monitor mon(F_if);
endmodule
```

/// small testbench to test golden model

```
import shared_pkg::*;
// Testbench for refrence
module refrence_tb ();
    logic [FIFO_WIDTH-1:0] data_in;
    logic clk, rst_n, wr_en, rd_en;
    logic [FIFO_WIDTH-1:0] data_out_ref;
    logic wr_ack_ref, overflow_ref;
    logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;

    initial begin
        clk = 0;
        forever begin
            #1 clk = ~clk;
        end
    end

    FIFO_ref dut(.*);
    int i;
    initial begin
        $readmemh ("fifoo.dat", dut.mem);
        rst_n = 0; #2; rst_n = 1; #1;
        rst_n = 0; #2; rst_n = 1; #1;

        wr_en = 1; rd_en = 0;
        for (i = 0; i<FIFO_DEPTH; i++) begin
            data_in = i+1;
            @(posedge clk);
        end
        data_in = 0;
        repeat(3) @(posedge clk);

        wr_en = 0; rd_en = 1;
        repeat(FIFO_DEPTH) begin
            @(posedge clk);
        end
        repeat(3) @(posedge clk);

        wr_en = 1; rd_en = 0;
        for (i = 0; i<5; i++) begin
            data_in = i+1;
            @(posedge clk);
        end

        wr_en = 1; rd_en = 1;
        for (i = 5; i<25; i++) begin
            data_in = 25-i;
            @(posedge clk);
        end
    $stop;
    end
endmodule
```

/// ** Bug_1: The signal wr_ack is sequential so it must be reset to zero when reset is activate ** ///

```
#    Time: 66 ns  Scope: scoreboard_pkg.FIFO_scoreboard.check_data File: package/scoreboard_pkg.sv Line: 38
# ** Error:                  66: Error - wr_ack_ref = 0, wr_ack = 1
```

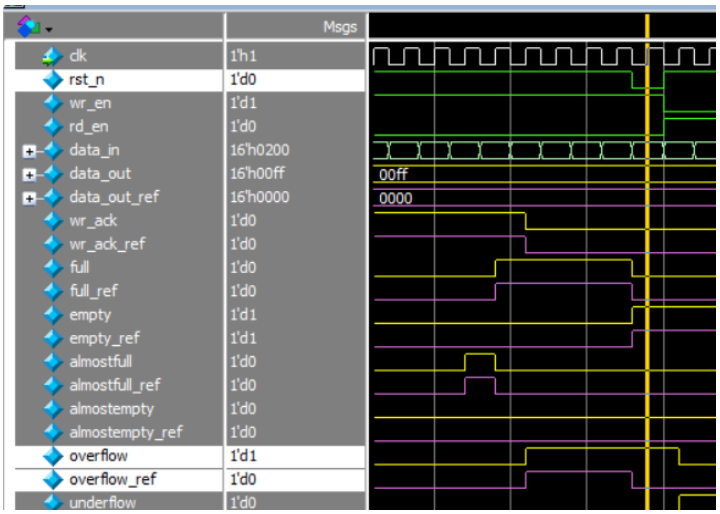


FIX:

```
if (!F_if.rst_n) begin
    wr_ptr <= 0;
    F_if.wr_ack <= 0; //
end
```

/// ** Bug_2: The signal overflow is sequential so it must be reset to zero when reset is activate ** ///

```
#    Time: 670 ns  Scope: scoreboard_pkg.FIFO_scoreboard.check_data File: package/scoreboard_pkg.sv Line: 38
# ** Error:                  670: Error - overflow_ref = 0, overflow = 1
```

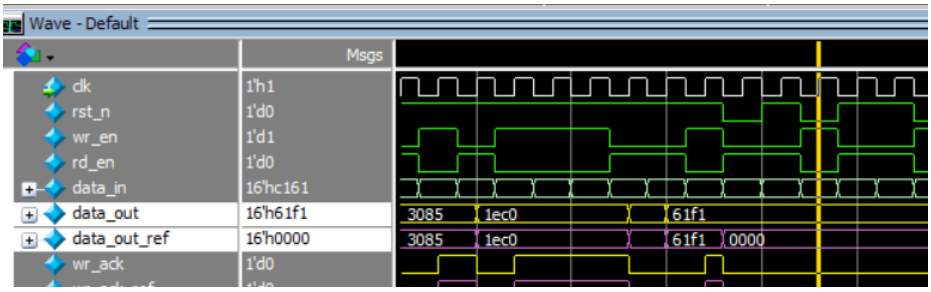


Fix:

```
if (!F_if.rst_n) begin
    wr_ptr <= 0;
    F_if.wr_ack <= 0; //
    F_if.overflow <= 0; //
end
```

/// ** Bug_3: The data_out bus is sequential so it must be reset to zero when reset is activate ** ///

```
# ** Error:                  388: Error - data_out_ref = 0, data_out = 4096
#    Time: 388 ns  Scope: scoreboard_pkg.FIFO_scoreboard.check_data File: package/scoreboard_pkg.sv Line: 38
```



FIX:

```
    if (!F_if.rst_n) begin
        rd_ptr <= 0;
        F_if.underflow <= 0;// Fix
        F_if.data_out <= 0;// Fix
    end
```

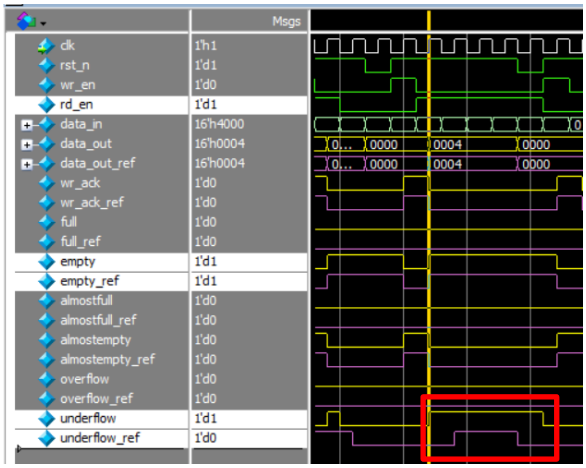
/// ** Bug_4: underflow must be sequential not combinational ** ///

Bug in code:

```
assign underflow = (empty && rd_en)? 1 : 0;
```

Bug in waveform:

```
#    Time: 530 ns  Scope: scoreboard_pkg.FIFO_scoreboard.check_data File: package/scoreboard_pkg.sv Line: 66
# ** Error:                  540: Error - underflow_ref = 0, underflow = 1
```



FIX:

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

```
// assign F_if.underflow = (F_if.empty && F_if.rd_en)? 1 : 0; //
```

/// ** Bug_4 cont.: The underflow is sequential so it must be reset to zero when reset is activate ** ///

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

/// ** Bug_5: Counter is not handle the case of wr_en and rd_en both are high ** ///

Bug in code:

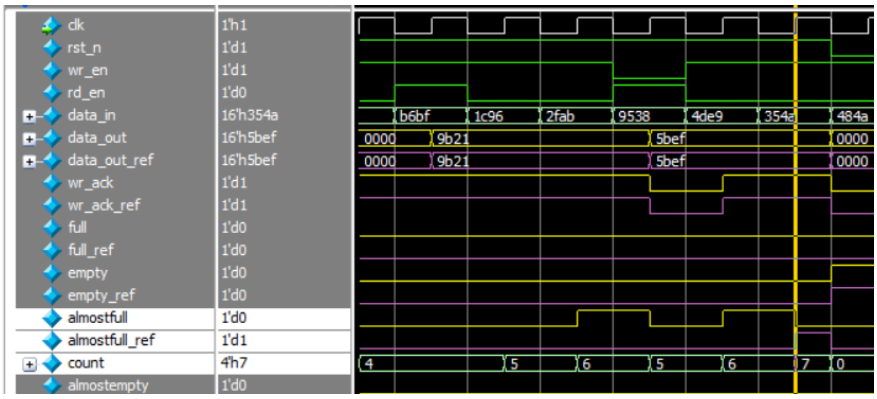
```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
    end
end
```

Fix:

```
        if (({F_if.wr_en, F_if.rd_en} == 2'b11) && F_if.full) // FIX
            count <= count - 1; // FIX
        else if (({F_if.wr_en, F_if.rd_en} == 2'b11) && F_if.empty) // FIX
            count <= count + 1; // FIX
        else if ( ({F_if.wr_en, F_if.rd_en} == 2'b10) && !F_if.full)
            count <= count + 1;
        else if ( ({F_if.wr_en, F_if.rd_en} == 2'b01) && !F_if.empty)
            count <= count - 1;
```

/// ** Bug_6: almostfull is high when internal signal “count” is less than FIFO_DEPTH by one** ///

```
# ** Error:                  158: Error - almostfull_ref = 0, almostfull = 1
#    Time: 158 ns  Scope: scoreboard_pkg.FIFO_scoreboard.check_data File: package/scoreboard_pkg.sv Line: 58
```




```
assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
```

Fix:

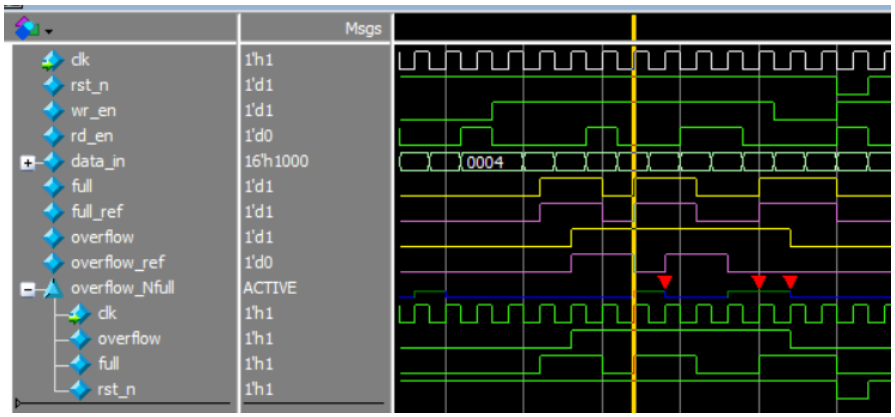
```
assign F_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //
```

/// ** Bug_7: overflow** ///

FIFO is full => inputs(wr_en = 1, rd_en = 1) then overflow gets high and FIFO no longer full

At next clk cycle inputs(wr_en = 1, rd_en = X) write operation will succeed but overflow still high

/top/dut/overflow_Nfull Concurrent SVA on 172



Fix:

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

/// ** Bug_8: in if condition should be “&&” not “&” only to achieve 100% Condition Coverage** ///

```
----------------Focused Condition View-------------------
Line       31 Item    1  (F_if.full & F_if.wr_en)
Condition totals: 1 of 2 input terms covered = 50.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
   F_if.full         N  '_0' not hit             Hit '_0'
  F_if.wr_en         Y
```

```
        if (F_if.full & F_if.wr_en)
            F_if.overflow <= 1;
```

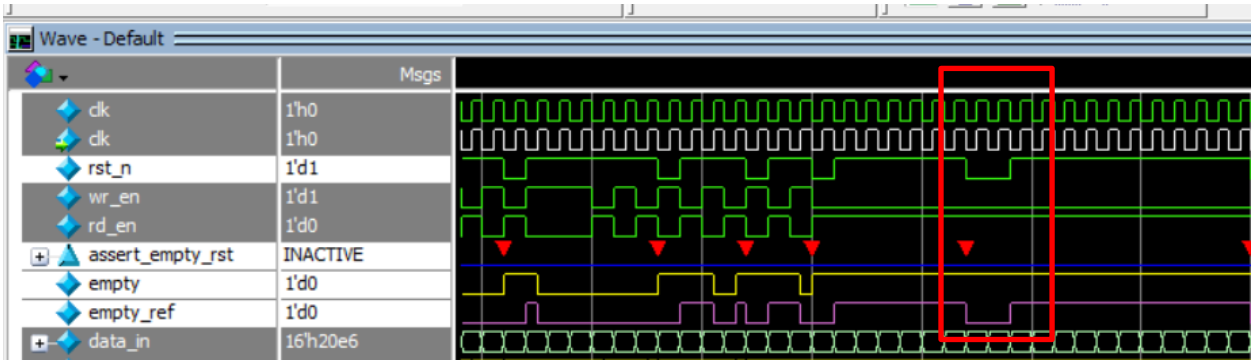
Fix:

```
        if (F_if.full && F_if.wr_en)// FIX
            F_if.overflow <= 1;
```

```
----------------Focused Condition View-------------------
Line       31 Item    1  (F_if.full && F_if.wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

/// ** Bug_9: when rst_n is activated then empty signal must be zero** ///

```
** Error: Assertion error.
   Time: 764 ns  Scope: top.dut.rst_n_assert.assert_empty_rst File: design/FIFO.sv Line: 139
** Error: empty = 1, empty_ref = 0
   Time: 766 ns  Scope: scoreboard_pkg.FIFO_scoreboard.display_errors File: package/scoreboard_pkg.sv Line: 51
```



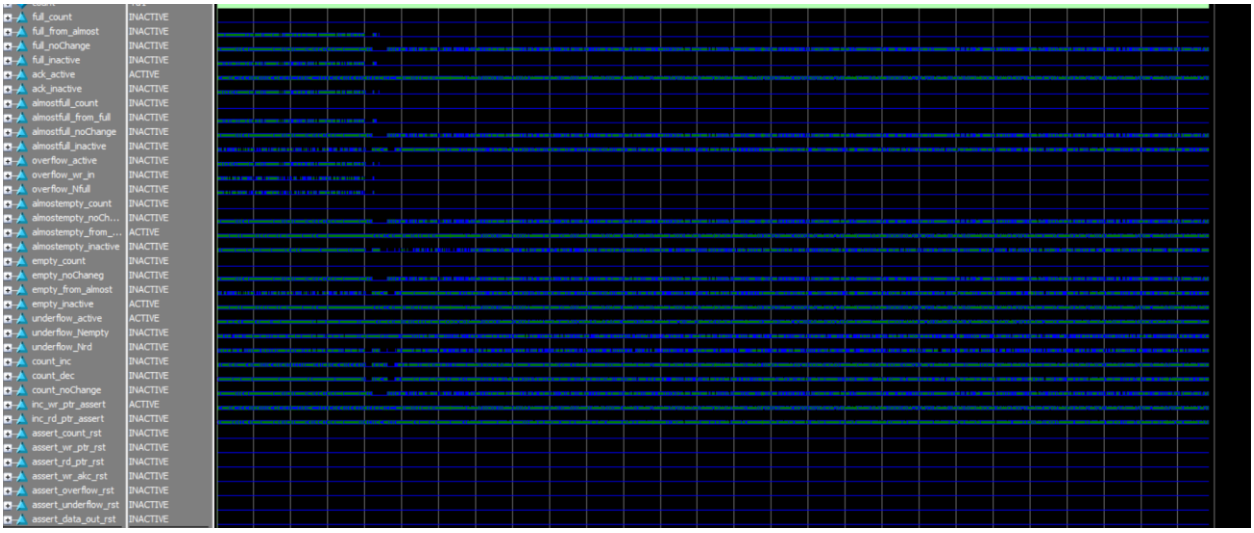
Fix:

```
assign F_if.empty = (count == 0 && F_if.rst_n)? 1 : 0; // FIX
```

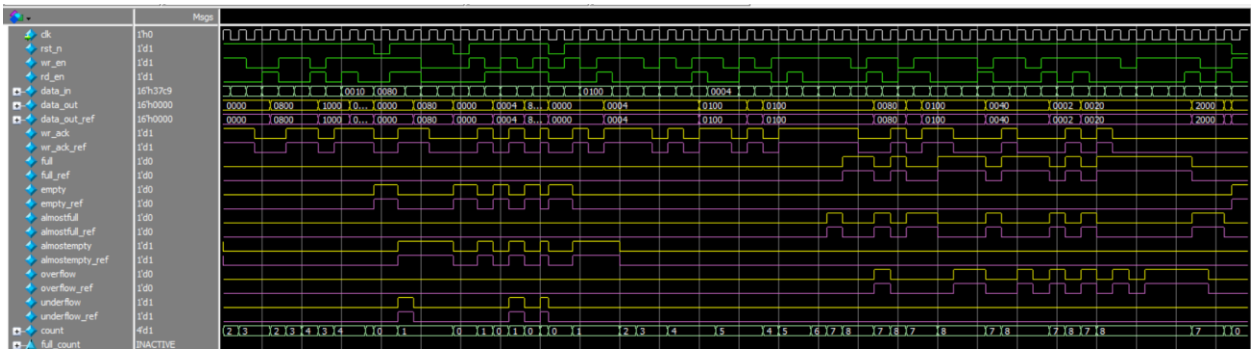
Bugs report summary.	
1. wr_ack	reset when reset is activated
2. overflow	reset when reset is activated
3. underflow	Add instruction to make it sequential.
4. underflow	reset when reset is activated
5. data_out	reset when reset is activated
6. count	Adding case when wr_en and rd_en are high
7. almostfull	High when count less than FIFO_DEPTH by 1 not 2
8. overflow	Must get low when full is zero
9. If statement	Should be “&&” not “&”
10. empty	empty must get low when reset is activated

Questa Snippet

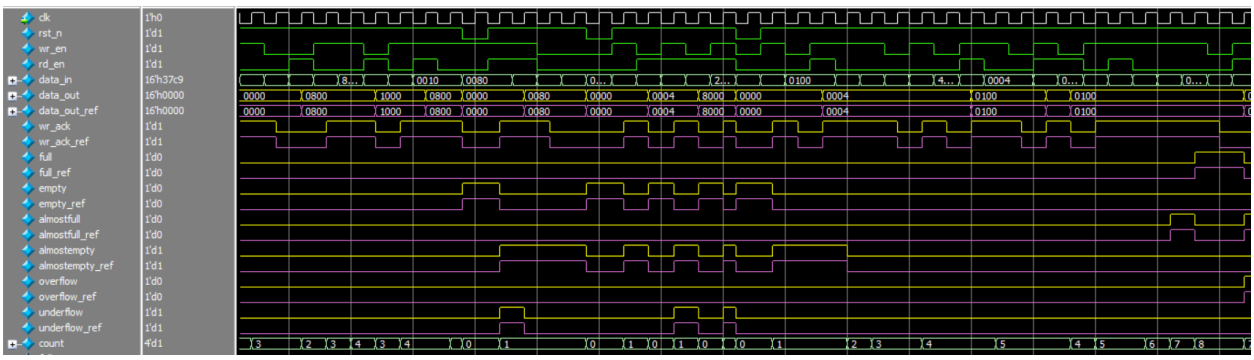
/// Big Picture for assertion shows that they are activated///



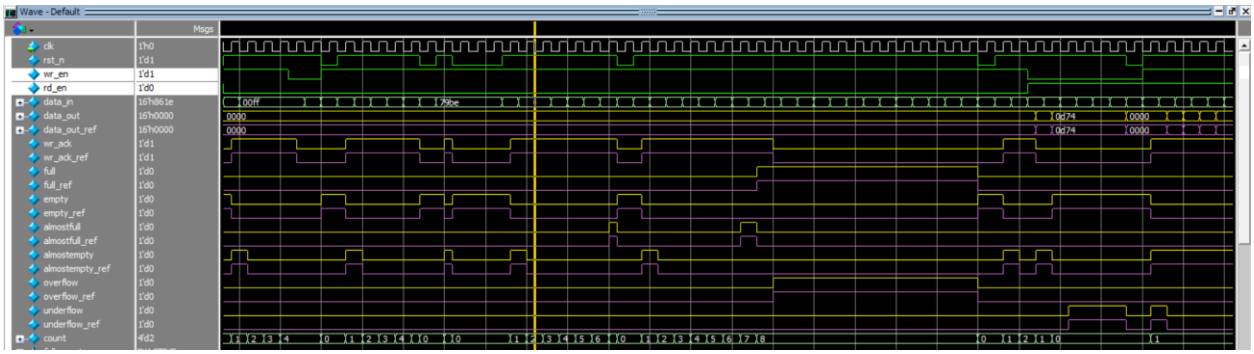
/// Big Picture for signals ///



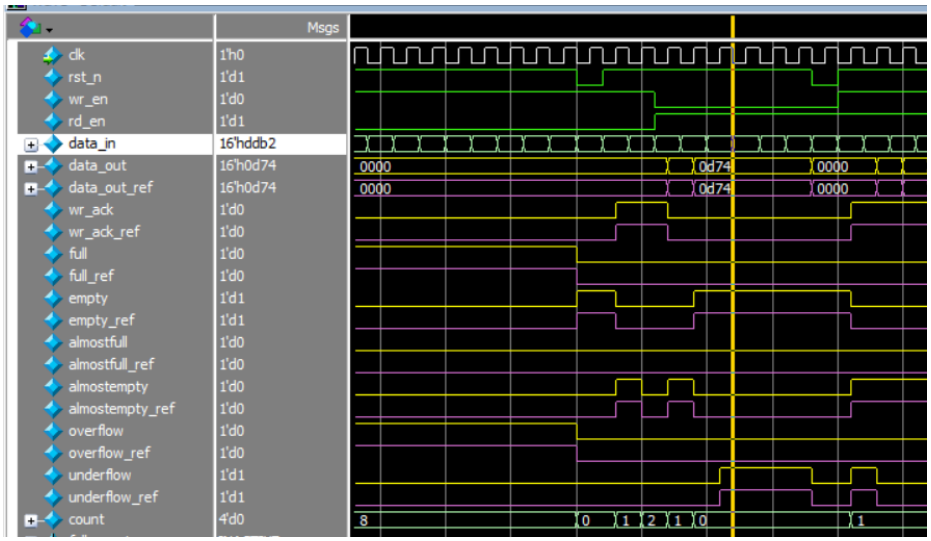
// Constraint CON_W_R_DIST ///



/// Constraint MY_CON_ONLEY_W ///

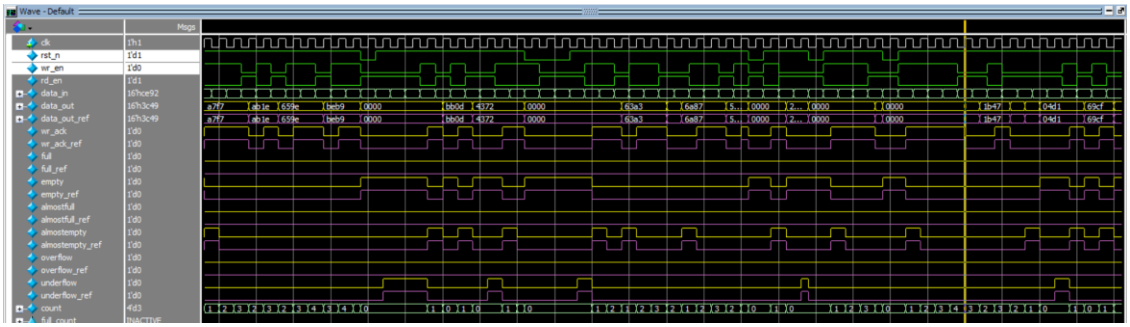


/// Constraint MY_CON_ONLEY_R ///

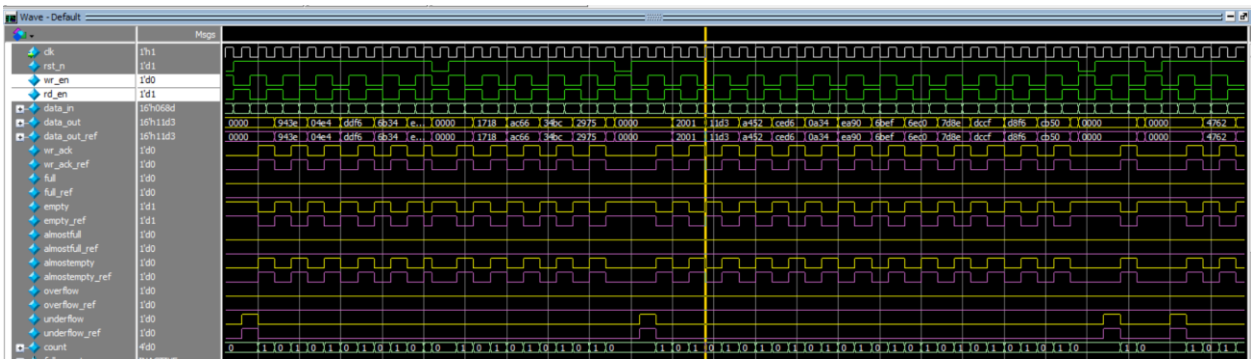


/// Constraint MY_CON_BOTH_ACTIVE ///

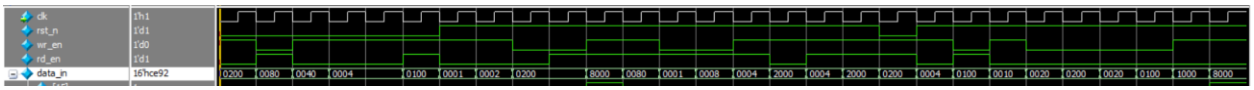
/// Constraint MY_CON_OPPSITE ///



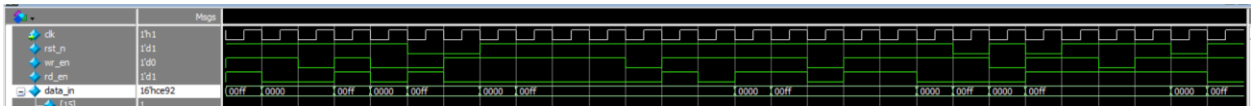
/// Constraint Toggle ///



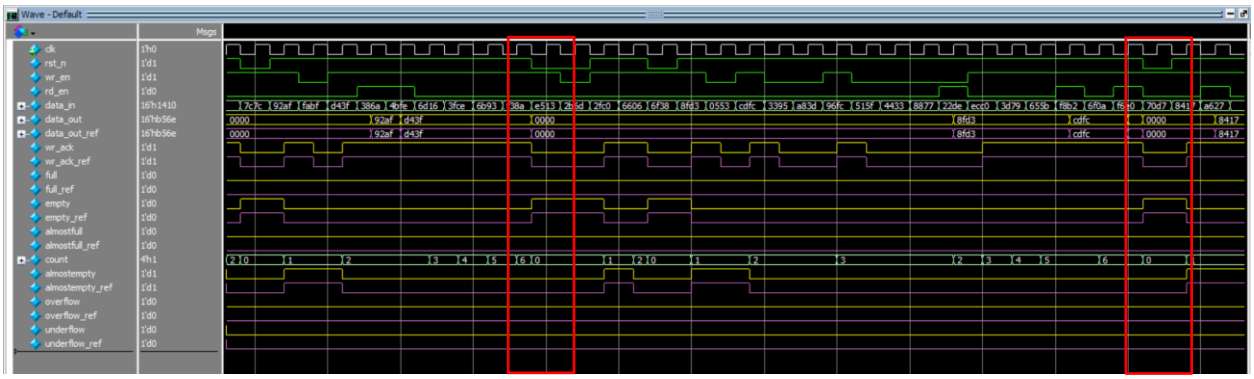
/// Constraint CON_DATA_OUT_ONE_BIT ///



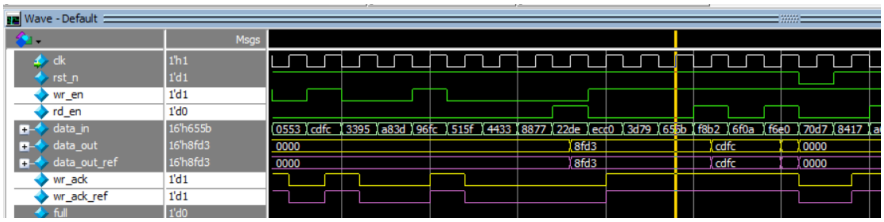
/// Constraint CON_DATA_OUT_M_Z ///



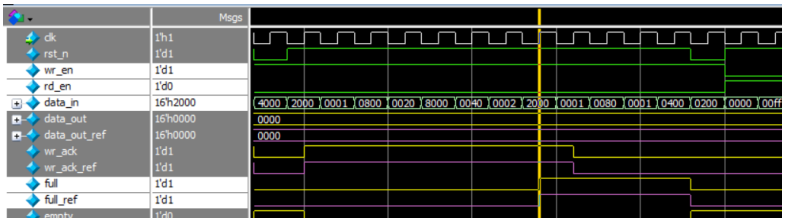
/// reset activate ///



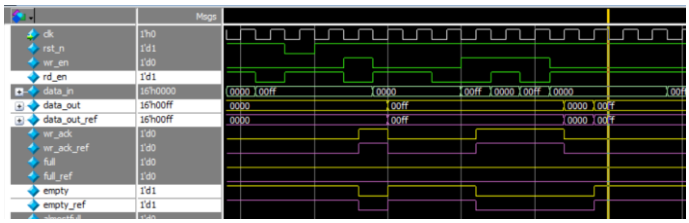
/// Write Acknowledge///



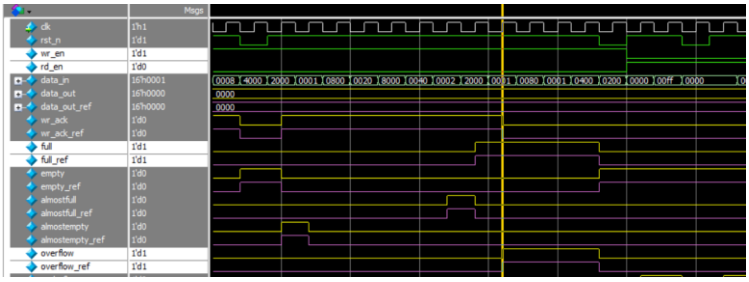
/// Write until gets full///



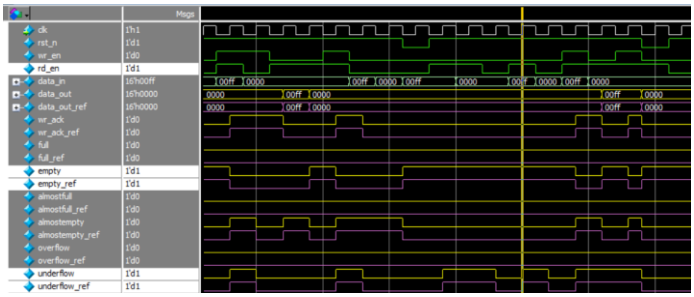
/// Read until gets empty///



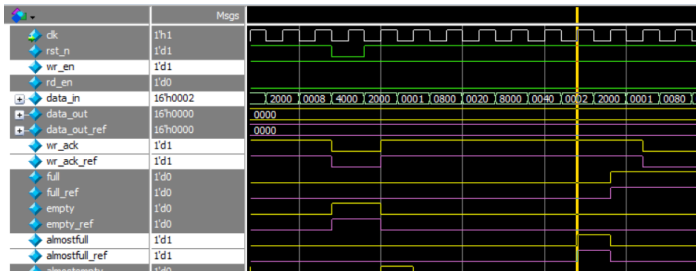
/// Overflow///



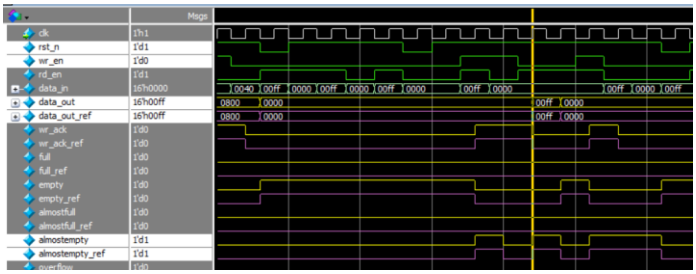
/// Underflow///



/// almost_full///



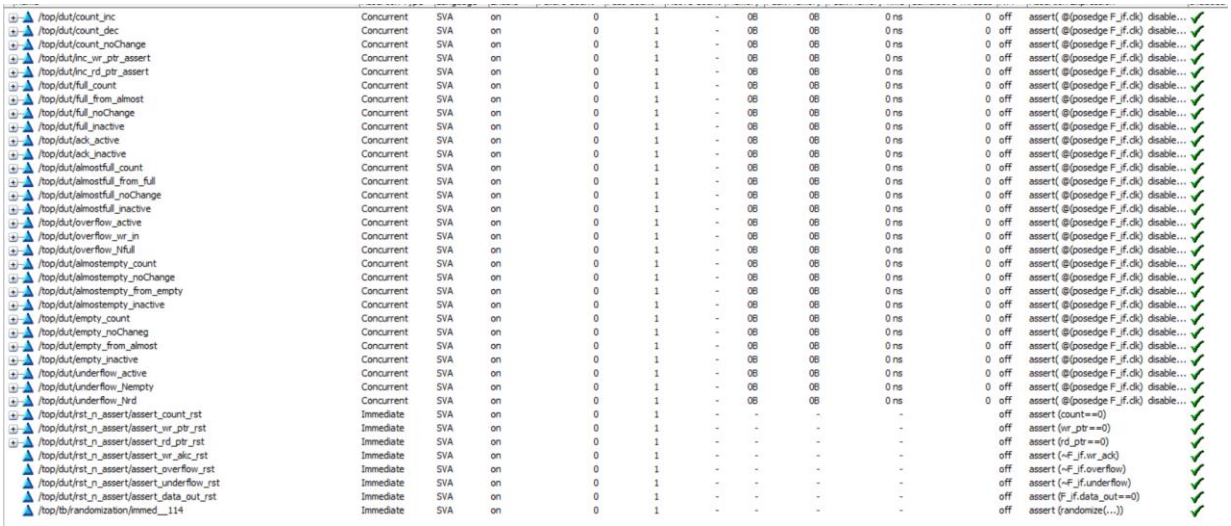
/// almost_empty///



/// Summary at the end of simulation ///

```
# ************ ****************************************** ************
# ************ ***************** summary: ***************** ************
# ************ ****************************************** ************
# ************ ****************************************** ************
# ************ error counter = 0, correct counter = 125218 ************
# ** Note: $stop    : common/monitor.sv(58)
#    Time: 250436 ns  Iteration: 1  Instance: /top/mon
# Break in Module monitor at common/monitor.sv line 58
```

/// Assertion ///



/// Coverage directive ///

fcover_report.txt

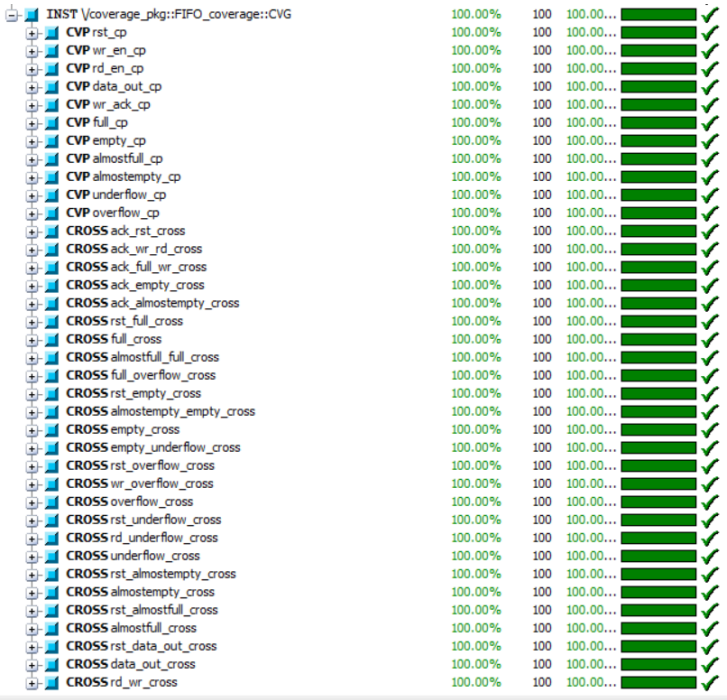
```
Coverage Report by instance with details

=================================================================================
=== Instance: /top/dut
=== Design Unit: work.FIFO
=================================================================================

Directive Coverage:
    Directives                      40        40         0    100.00%

DIRECTIVE COVERAGE:
---------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)
                                         Unit   UnitType
```

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/dut/count_inc_cover	SVA	✓	Off	71	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/count_dec_cover	SVA	✓	Off	34	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/count_noChange_cover	SVA	✓	Off	96	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/inc_wr_ptr_cover	SVA	✓	Off	188	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/inc_rd_ptr_cover	SVA	✓	Off	130	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_count_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_wr_ptr_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_rd_ptr_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_wr_akc_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_overflow_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_underflow_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_data_out_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_full_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_almostfull_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_empty_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/rst_n_cover/cover_almostempty_rst	SVA	✓	Off	40	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/full_count_cover	SVA	✓	Off	18	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/full_from_almost_cover	SVA	✓	Off	4	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/full_noChange_cover	SVA	✓	Off	96	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/full_inactive_cover	SVA	✓	Off	2	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/ack_active_cover	SVA	✓	Off	188	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/ack_inactive_cover	SVA	✓	Off	14	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostfull_count_cover	SVA	✓	Off	16	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostfull_from_full_cover	SVA	✓	Off	2	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostfull_noChange_cover	SVA	✓	Off	96	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostfull_inactive_cover	SVA	✓	Off	10	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/overflow_active_cover	SVA	✓	Off	14	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/overflow_wr_in_cover	SVA	✓	Off	4	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/overflow_hfull_cover	SVA	✓	Off	2	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostempty_count_cover	SVA	✓	Off	126	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostempty_noChange_cover	SVA	✓	Off	96	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostempty_from_empty_cover	SVA	✓	Off	38	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/almostempty_inactive_cover	SVA	✓	Off	15	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/empty_count_cover	SVA	✓	Off	90	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/empty_noChaneg_cover	SVA	✓	Off	96	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/empty_from_almost_cover	SVA	✓	Off	10	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/empty_inactive_cover	SVA	✓	Off	38	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/underflow_active_cover	SVA	✓	Off	61	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/underflow_hempty_cover	SVA	✓	Off	19	1	Uni...	1	100%		✓	0	0	0 ns	0
/top/dut/underflow_hrd_cover	SVA	✓	Off	6	1	Uni...	1	100%		✓	0	0	0 ns	0

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

Directive Coverage:
    Directives                      40        40         0    100.00%
```

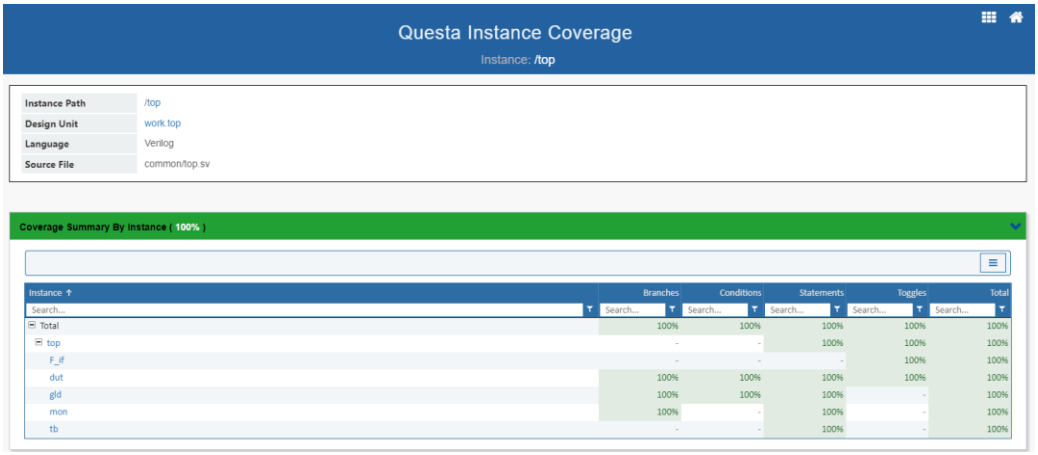
/// Covergroup ///



INST \/coverage_pkg::FIFO_coverage::CVG	100.00%	100	100.00...
CVP rst_cp	100.00%	100	100.00...
CVP wr_en_cp	100.00%	100	100.00...
CVP rd_en_cp	100.00%	100	100.00...
CVP data_out_cp	100.00%	100	100.00...
CVP wr_ack_cp	100.00%	100	100.00...
CVP full_cp	100.00%	100	100.00...
CVP empty_cp	100.00%	100	100.00...
CVP almostfull_cp	100.00%	100	100.00...
CVP almostempty_cp	100.00%	100	100.00...
CVP underflow_cp	100.00%	100	100.00...
CVP overflow_cp	100.00%	100	100.00...
CROSS ack_rst_cross	100.00%	100	100.00...
CROSS ack_wr_rd_cross	100.00%	100	100.00...
CROSS ack_full_wr_cross	100.00%	100	100.00...
CROSS ack_empty_cross	100.00%	100	100.00...
CROSS ack_almostempty_cross	100.00%	100	100.00...
CROSS rst_full_cross	100.00%	100	100.00...
CROSS full_cross	100.00%	100	100.00...
CROSS almostfull_full_cross	100.00%	100	100.00...
CROSS full_overflow_cross	100.00%	100	100.00...
CROSS rst_empty_cross	100.00%	100	100.00...
CROSS almostempty_empty_cross	100.00%	100	100.00...
CROSS empty_cross	100.00%	100	100.00...
CROSS empty_underflow_cross	100.00%	100	100.00...
CROSS rst_overflow_cross	100.00%	100	100.00...
CROSS wr_overflow_cross	100.00%	100	100.00...
CROSS overflow_cross	100.00%	100	100.00...
CROSS rst_underflow_cross	100.00%	100	100.00...
CROSS rd_underflow_cross	100.00%	100	100.00...
CROSS underflow_cross	100.00%	100	100.00...
CROSS rst_almostempty_cross	100.00%	100	100.00...
CROSS almostempty_cross	100.00%	100	100.00...
CROSS rst_almostfull_cross	100.00%	100	100.00...
CROSS almostfull_cross	100.00%	100	100.00...
CROSS rst_data_out_cross	100.00%	100	100.00...
CROSS data_out_cross	100.00%	100	100.00...
CROSS rd_wr_cross	100.00%	100	100.00...

```
================================================================================
=== Instance: /coverage_pkg
=== Design Unit: work.coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        35        na        na        na
            Covergroup Bins       235       235         0   100.00%
```

/// Code Coverage Summary ///




```
Coverage Report by instance with details

================================================================================
=== Instance: /\top#dut
=== Design Unit: work.FIFO
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        27        27         0   100.00%
```

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      24        24         0   100.00%
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        31         0   100.00%
```

FIFO toggle for internal signal (wr_ptr, rd_ptr and count)

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

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

Toggle Coverage for instance /top/dut --

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

FIFO_interfase toggle for all signals (inputs and outputs)

```
================================================================================
=== Instance: /top/F_if
=== Design Unit: work.FIFO_interface
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        132       132         0   100.00%
```