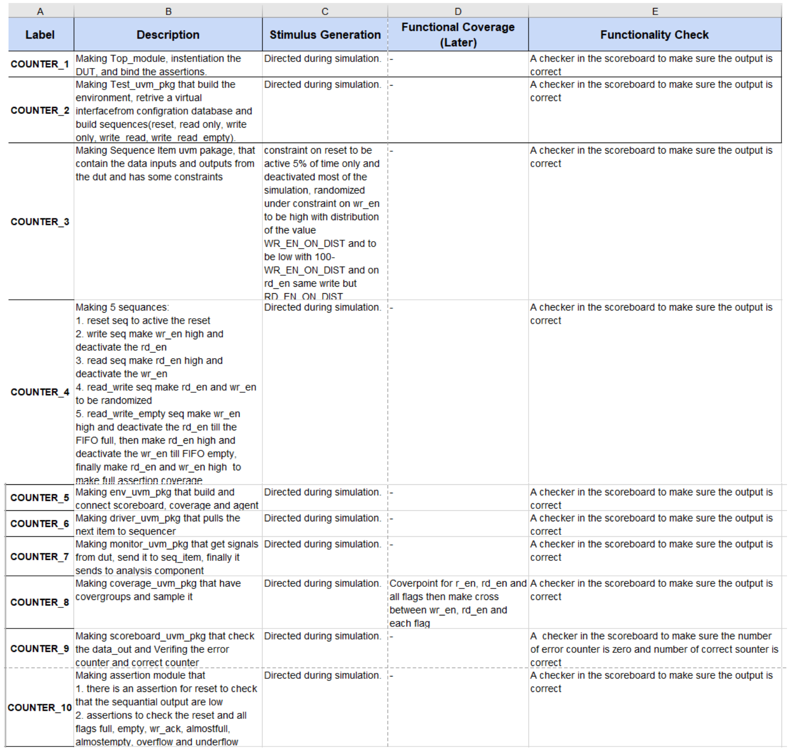
FIFO_UVM PROJECT

Name	Sarah Abdelatty Ibrahem
Group	**2**

Verification Plan snippets:



A	B	C	D	E
Label	**Description**	**Stimulus Generation**	**Functional Coverage (Later)**	**Functionality Check**
COUNTER_1	Making Top_module, instentiation the DUT, and bind the assertions.	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_2	Making Test_uvm_pkg that build the environment, retrive a virtual interfacefrom configration database and build sequences(reset, read only, write only, write_read, write_read_empty).	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_3	Making Sequence Item uvm pakage, that contain the data inputs and outputs from the dut and has some constraints	constraint on reset to be active 5% of time only and deactivated most of the simulation, randomized under constraint on wr_en to be high with distribution of the value WR_EN_ON_DIST and to be low with 100-WR_EN_ON_DIST and on rd_en same write but RD_EN_ON_DIST	-	A checker in the scoreboard to make sure the output is correct
COUNTER_4	Making 5 sequances: 1. reset seq to active the reset 2. write seq make wr_en high and deactivate the rd_en 3. read seq make rd_en high and deactivate the wr_en 4. read_write seq make rd_en and wr_en to be randomized 5. read_write_empty seq make wr_en high and deactivate the rd_en till the FIFO full, then make rd_en high and deactivate the wr_en till FIFO empty, finally make rd_en and wr_en high to make full assertion coverage	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_5	Making env_uvm_pkg that build and connect scoreboard, coverage and agent	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_6	Making driver_uvm_pkg that pulls the next item to sequencer	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_7	Making monitor_uvm_pkg that get signals from dut, send it to seq_item, finally it sends to analysis component	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct
COUNTER_8	Making coverage_uvm_pkg that have covergroups and sample it	Directed during simulation.	Coverpoint for r_en, rd_en and all flags then make cross between wr_en, rd_en and each flag	A checker in the scoreboard to make sure the output is correct
COUNTER_9	Making scoreboard_uvm_pkg that check the data_out and Verifing the error counter and correct counter	Directed during simulation.	-	A checker in the scoreboard to make sure the number of error counter is zero and number of correct sounter is correct
COUNTER_10	Making assertion module that 1. there is an assertion for reset to check that the sequantial output are low 2. assertions to check the reset and all flags full, empty, wr_ack, almostfull, almostempty, overflow and underflow	Directed during simulation.	-	A checker in the scoreboard to make sure the output is correct

Bug report:

1. Reset the sequential outputs signals (overflow, underflow and wr_ack).
2. Underflow is sequential output signal so handle it in always block (always block for read).
3. Almostfull flag is high when the FIFO has one place not two.
4. Unhandled cases for read and write:
   - If a read and write enables were high and the FIFO was empty, only writing will take place and vice verse if the FIFO was full.
5. When successful write, overflow should be low.
6. When successful read, underflow should be low.

Testbench Flow:

A) **FIFO_sequence_item**:
   1. Contains data fields to interact with the DUT (input and output signals).
   2. Randomizes these signals.
   3. Contains constraint blocks to ensure the verification plan is met.

B) **Sequences**:
   1. There are four types of sequences: Reset, Write-only, Read-only, and Write-Read sequences.
   2. These sequences form the core stimulus of the verification plan.
   3. Each sequence is written within a task body.

C) **FIFO_sequencer**: Generates transaction class objects and sends them to the driver (FIFO_driver) for execution.

D) **FIFO_scoreboard**:
   1. Receives sequence items from the monitor.
   2. Runs input signals through a reference model (implemented as a task or module; here, a task is used) to compare the DUT's output with the expected output and verify FIFO functionality.

E) **FIFO_coverage**:
   1. Receives sequence items from the monitor.
   2. Contains covergroups to ensure the verification plan is covered.
   3. Samples data fields for functional coverage.

F) **FIFO_driver:**
   1. Retrieves the next item from the sequencer.
   2. Drives the sequence item in the `run_phase` task using the virtual interface.

G) **FIFO_monitor**: Captures signal information from the DUT, converts it into sequence items, and sends it to analysis components (Ports and Exports).

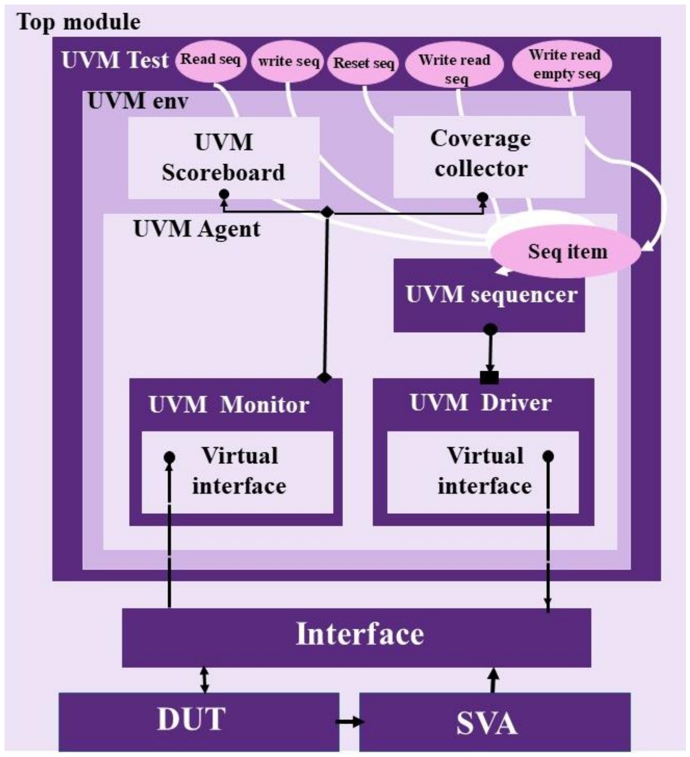
H) **FIFO_env**: Builds and connects the scoreboard (FIFO_scoreboard), coverage collector (FIFO_coverage), agent (FIFO_agent), and analysis components (Ports and Exports).

I) **FIFO_test**:
   1. Builds the FIFO environment and sequences.
   2. Retrieves the virtual interface from the configuration database using a configuration object, which holds the settings and parameters for UVM components.
   3. Sets the configuration object into the configuration database.
   4. Constructs the environment (FIFO_env).
   5. Initiates the sequences on the sequencer.

J) **FIFO_top module**:

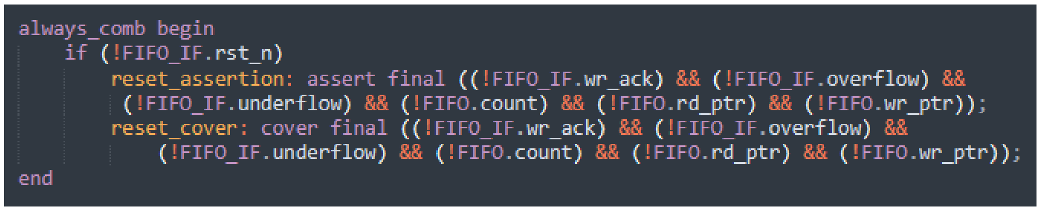
1. Instantiates the DUT, FIFO_interface, and binds assertions (FIFO_SVA).
2. Generates the clock.
3. Passes the interface (virtual FIFO_interface) via a shared configuration database.
4. Executes the test.

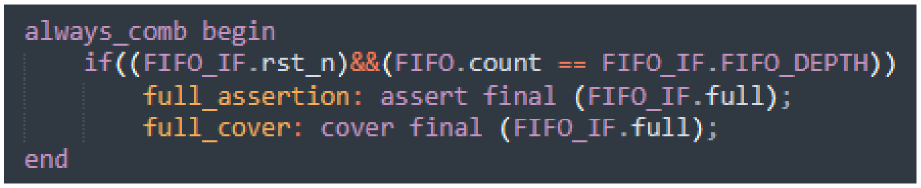
UVM testbench structure:

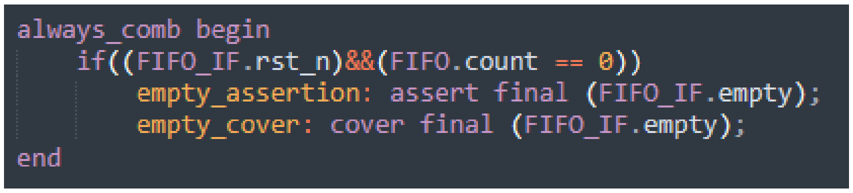


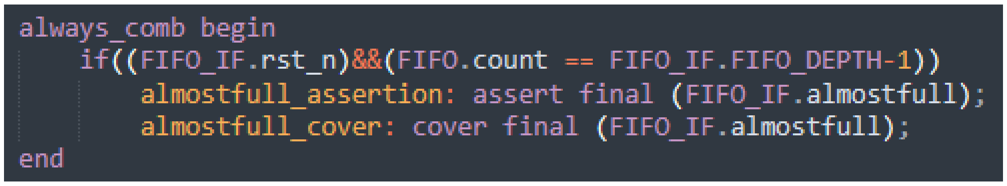
Assertion Table:

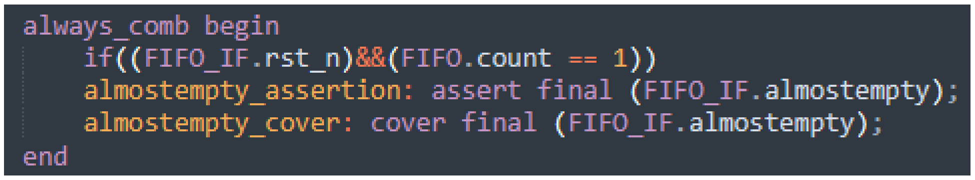
Feature	Assertion

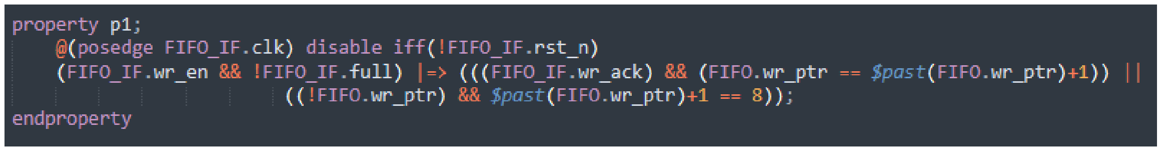
Whenever the rst_n is active, All sequential flags and internal signals should be low.	
	`always_comb begin`
	`    if (!FIFO_IF.rst_n)`
	`        reset_assertion: assert final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&`
	`         (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));`
	`        reset_cover: cover final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&`
	`            (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));`
	`end`

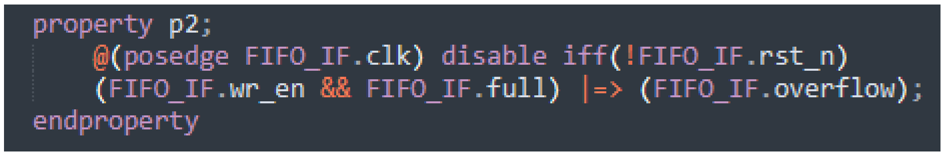
Whenever the rst_n is deactivated & number of FIFO elements equal FIFO maximum depth, full flag should be high.	
	`always_comb begin`
	`    if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH))`
	`        full_assertion: assert final (FIFO_IF.full);`
	`        full_cover: cover final (FIFO_IF.full);`
	`end`

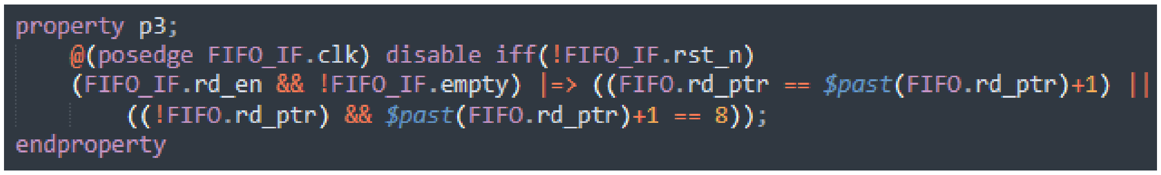
Whenever the rst_n is deactivated & number of FIFO elements equal zero, empty flag should be high.	
	`always_comb begin`
	`    if((FIFO_IF.rst_n)&&(FIFO.count == 0))`
	`        empty_assertion: assert final (FIFO_IF.empty);`
	`        empty_cover: cover final (FIFO_IF.empty);`
	`end`

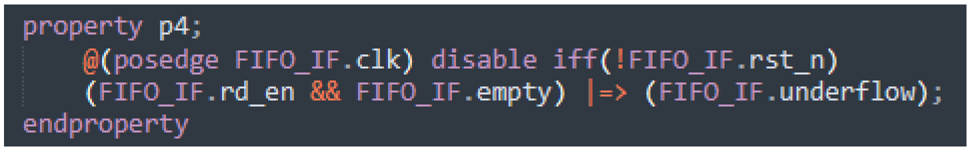
Whenever the rst_n is deactivated & number of FIFO elements equal FIFO maximum depth -1, almostfull flag should be high.	
	`always_comb begin`
	`    if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH-1))`
	`        almostfull_assertion: assert final (FIFO_IF.almostfull);`
	`        almostfull_cover: cover final (FIFO_IF.almostfull);`
	`end`

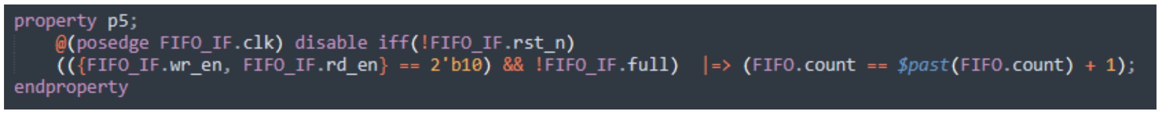
Whenever the rst_n is deactivated & number of FIFO elements equal 1, almostempty flag should be high.	
	`always_comb begin`
	`    if((FIFO_IF.rst_n)&&(FIFO.count == 1))`
	`    almostempty_assertion: assert final (FIFO_IF.almostempty);`
	`    almostempty_cover: cover final (FIFO_IF.almostempty);`
	`end`

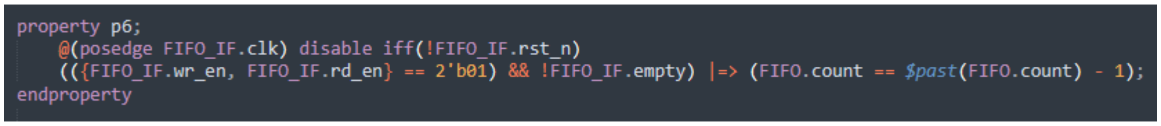
Whenever the rst_n is deactivated, Write enable is high & FIFO is not full, wr_ack should be high & wr_ptr should increment.	
	`property p1;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (FIFO_IF.wr_en && !FIFO_IF.full) |=> (((FIFO_IF.wr_ack) && (FIFO.wr_ptr == $past(FIFO.wr_ptr)+1)) ||`
	`                        ((!FIFO.wr_ptr) && $past(FIFO.wr_ptr)+1 == 8));`
	`endproperty`

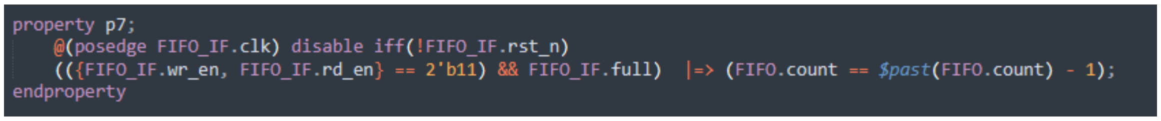
Whenever the rst_n is deactivated, Write enable is high & FIFO is full, overflow should be high.	
	`property p2;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (FIFO_IF.wr_en && FIFO_IF.full) |=> (FIFO_IF.overflow);`
	`endproperty`

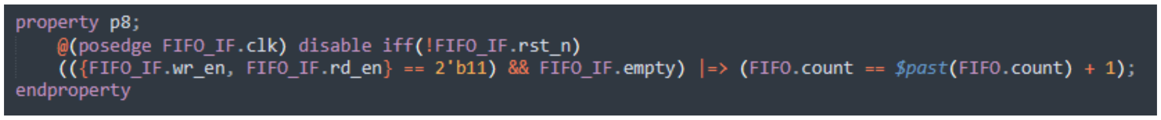
Whenever the rst_n is deactivated, Read enable is high & number of FIFO elements doesn’t equal zero, rd_ptr should increment.	 `property p3;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (FIFO_IF.rd_en && !FIFO_IF.empty) |=> ((FIFO.rd_ptr == $past(FIFO.rd_ptr)+1) ||`
	`        ((!FIFO.rd_ptr) && $past(FIFO.rd_ptr)+1 == 8));`
	`endproperty`

Whenever the rst_n is deactivated, Read enable is high & FIFO is empty, underflow should be high.	 `property p4;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (FIFO_IF.rd_en && FIFO_IF.empty) |=> (FIFO_IF.underflow);`
	`endproperty`

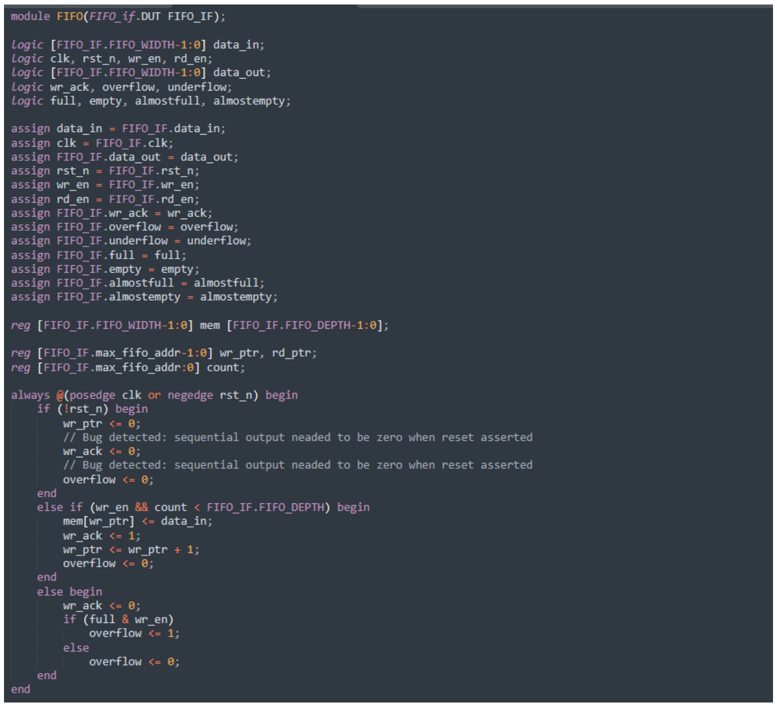
Whenever the rst_n is deactivated, Write enable is high & FIFO is not full, Write operation should take place.	 `property p5;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full)  |=> (FIFO.count == $past(FIFO.count) + 1);`
	`endproperty`

Whenever the rst_n is deactivated, read enable is high & FIFO is not empty, Read operation should take place.	 `property p6;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty) |=> (FIFO.count == $past(FIFO.count) - 1);`
	`endproperty`

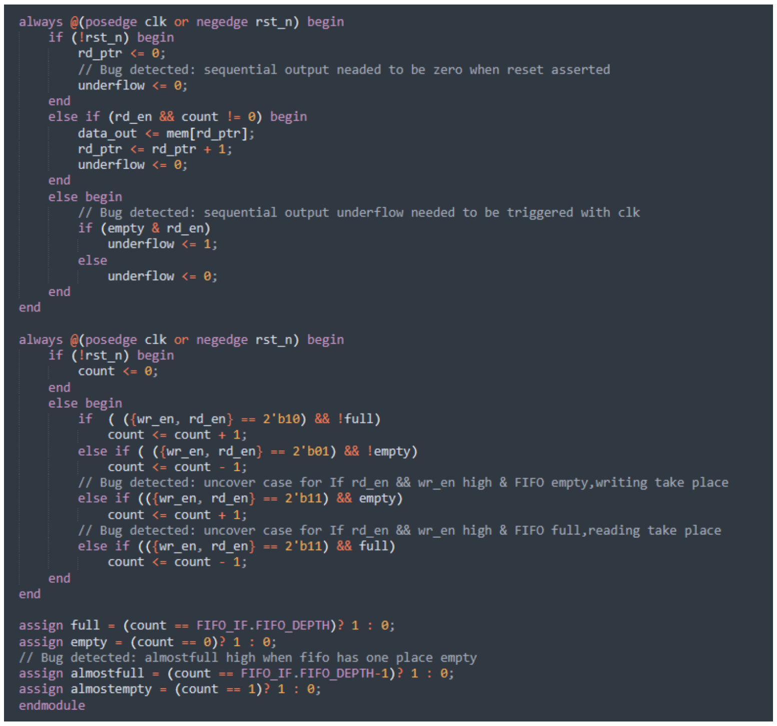
Whenever the rst_n is deactivated, Both of read & write enables are high & FIFO is empty, Write operation should take place.	 `property p7;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full)  |=> (FIFO.count == $past(FIFO.count) - 1);`
	`endproperty`

Whenever the rst_n is deactivated, Both of read & write enables are high & FIFO is not full, Read operation should take place.	 `property p8;`
	`    @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`    (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty) |=> (FIFO.count == $past(FIFO.count) + 1);`
	`endproperty`

Design code snippets:

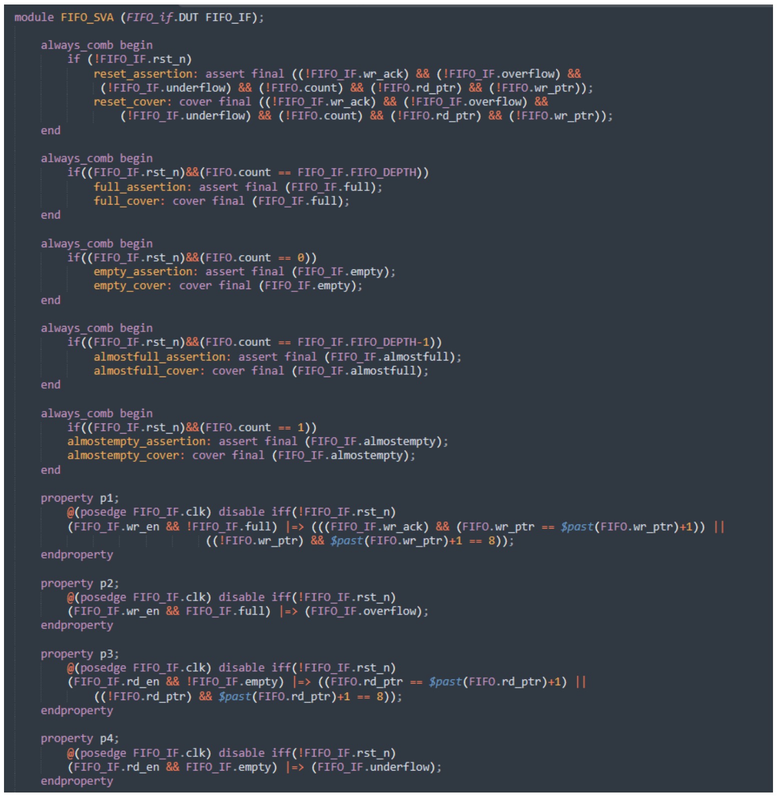


```
module FIFO(FIFO_if.DUT FIFO_IF);

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

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

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

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

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
        // Bug detected: sequential output neaded to be zero when reset asserted
        wr_ack <= 0;
        // Bug detected: sequential output neaded to be zero when reset asserted
        overflow <= 0;
    end
    else if (wr_en && count < FIFO_IF.FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        overflow <= 0;
    end
    else begin
        wr_ack <= 0;
        if (full & wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
end
```

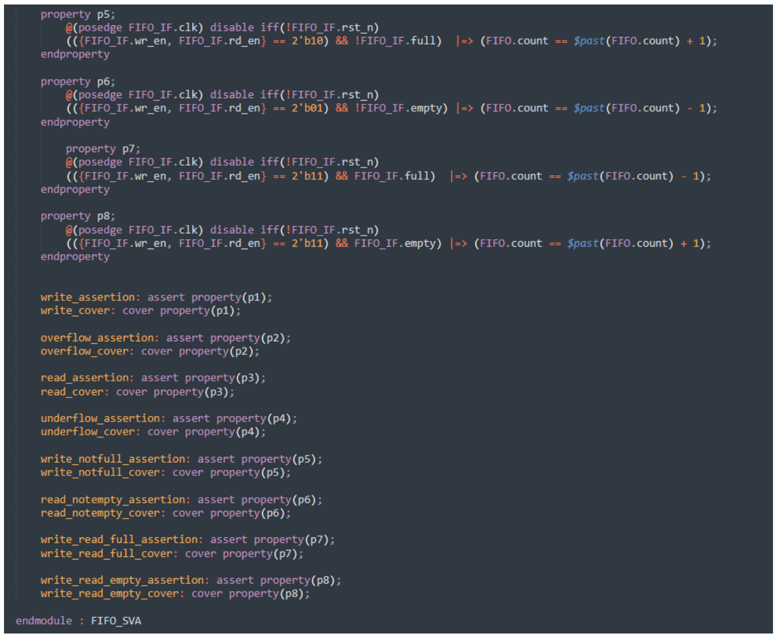


```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rd_ptr <= 0;
        // Bug detected: sequential output neaded to be zero when reset asserted
        underflow <= 0;
    end
    else if (rd_en && count != 0) begin
        data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        underflow <= 0;
    end
    else begin
        // Bug detected: sequential output underflow needed to be triggered with clk
        if (empty & rd_en)
            underflow <= 1;
        else
            underflow <= 0;
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
        // Bug detected: uncover case for If rd_en && wr_en high & FIFO empty,writing take place
        else if (({wr_en, rd_en} == 2'b11) && empty)
            count <= count + 1;
        // Bug detected: uncover case for If rd_en && wr_en high & FIFO full,reading take place
        else if (({wr_en, rd_en} == 2'b11) && full)
            count <= count - 1;
    end
end

assign full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
// Bug detected: almostfull high when fifo has one place empty
assign almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0;
assign almostempty = (count == 1)? 1 : 0;
endmodule
```

SVA code snippets:

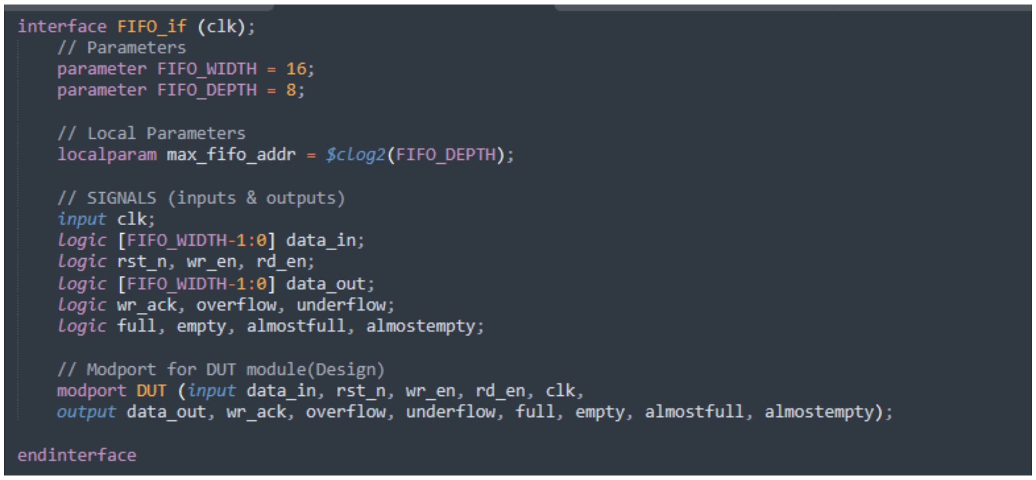


```systemverilog
module FIFO_SVA (FIFO_if.DUT FIFO_IF);

    always_comb begin
        if (!FIFO_IF.rst_n)
            reset_assertion: assert final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&
             (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));
            reset_cover: cover final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&
                (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));
    end

    always_comb begin
        if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH))
            full_assertion: assert final (FIFO_IF.full);
            full_cover: cover final (FIFO_IF.full);
    end

    always_comb begin
        if((FIFO_IF.rst_n)&&(FIFO.count == 0))
            empty_assertion: assert final (FIFO_IF.empty);
            empty_cover: cover final (FIFO_IF.empty);
    end

    always_comb begin
        if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH-1))
            almostfull_assertion: assert final (FIFO_IF.almostfull);
            almostfull_cover: cover final (FIFO_IF.almostfull);
    end

    always_comb begin
        if((FIFO_IF.rst_n)&&(FIFO.count == 1))
        almostempty_assertion: assert final (FIFO_IF.almostempty);
        almostempty_cover: cover final (FIFO_IF.almostempty);
    end

    property p1;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.wr_en && !FIFO_IF.full) |=> (((FIFO_IF.wr_ack) && (FIFO.wr_ptr == $past(FIFO.wr_ptr)+1)) ||
                           ((!FIFO.wr_ptr) && $past(FIFO.wr_ptr)+1 == 8));
    endproperty

    property p2;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.wr_en && FIFO_IF.full) |=> (FIFO_IF.overflow);
    endproperty

    property p3;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.rd_en && !FIFO_IF.empty) |=> ((FIFO.rd_ptr == $past(FIFO.rd_ptr)+1) ||
            ((!FIFO.rd_ptr) && $past(FIFO.rd_ptr)+1 == 8));
    endproperty

    property p4;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.rd_en && FIFO_IF.empty) |=> (FIFO_IF.underflow);
    endproperty
```



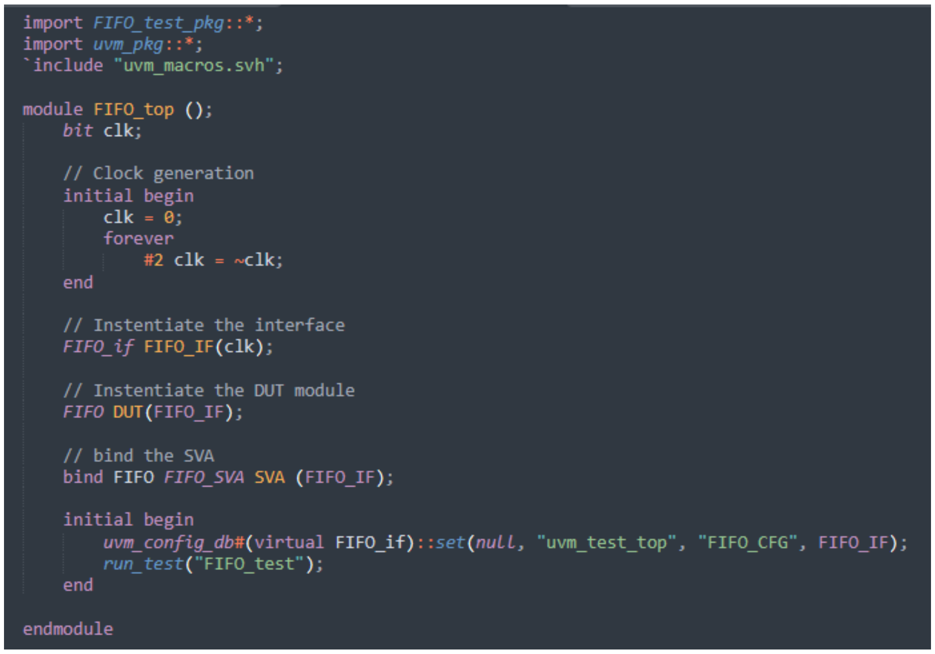
```systemverilog
    property p5;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full)  |=> (FIFO.count == $past(FIFO.count) + 1);
    endproperty

    property p6;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty) |=> (FIFO.count == $past(FIFO.count) - 1);
    endproperty

        property p7;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full)  |=> (FIFO.count == $past(FIFO.count) - 1);
    endproperty

    property p8;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty) |=> (FIFO.count == $past(FIFO.count) + 1);
    endproperty


    write_assertion: assert property(p1);
    write_cover: cover property(p1);

    overflow_assertion: assert property(p2);
    overflow_cover: cover property(p2);

    read_assertion: assert property(p3);
    read_cover: cover property(p3);

    underflow_assertion: assert property(p4);
    underflow_cover: cover property(p4);

    write_notfull_assertion: assert property(p5);
    write_notfull_cover: cover property(p5);

    read_notempty_assertion: assert property(p6);
    read_notempty_cover: cover property(p6);

    write_read_full_assertion: assert property(p7);
    write_read_full_cover: cover property(p7);

    write_read_empty_assertion: assert property(p8);
    write_read_empty_cover: cover property(p8);

endmodule : FIFO_SVA
```

Interface code snippet:



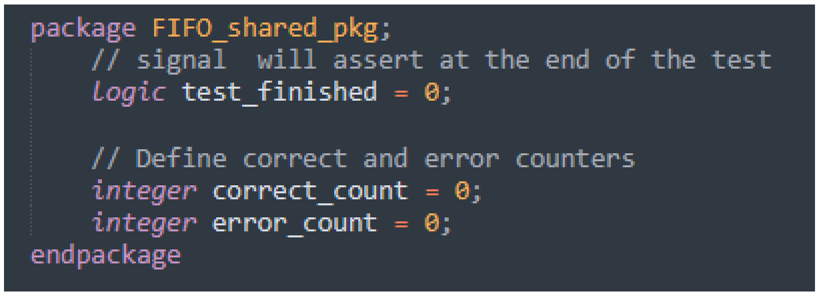
```systemverilog
interface FIFO_if (clk);
    // Parameters
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

    // Local Parameters
    localparam max_fifo_addr = $clog2(FIFO_DEPTH);

    // SIGNALS (inputs & outputs)
    input clk;
    logic [FIFO_WIDTH-1:0] data_in;
    logic rst_n, wr_en, rd_en;
    logic [FIFO_WIDTH-1:0] data_out;
    logic wr_ack, overflow, underflow;
    logic full, empty, almostfull, almostempty;

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

endinterface
```

Top code snippet:



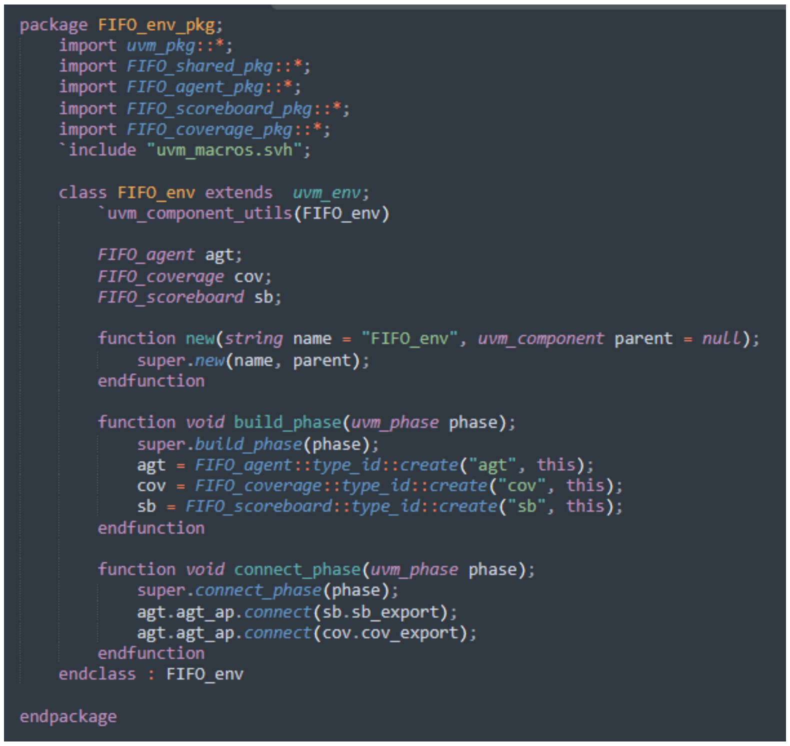
```systemverilog
import FIFO_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh";

module FIFO_top ();
    bit clk;

    // Clock generation
    initial begin
        clk = 0;
        forever
            #2 clk = ~clk;
    end

    // Instentiate the interface
    FIFO_if FIFO_IF(clk);

    // Instentiate the DUT module
    FIFO DUT(FIFO_IF);

    // bind the SVA
    bind FIFO FIFO_SVA SVA (FIFO_IF);

    initial begin
        uvm_config_db#(virtual FIFO_if)::set(null, "uvm_test_top", "FIFO_CFG", FIFO_IF);
        run_test("FIFO_test");
    end

endmodule
```

Shared package code snippet:



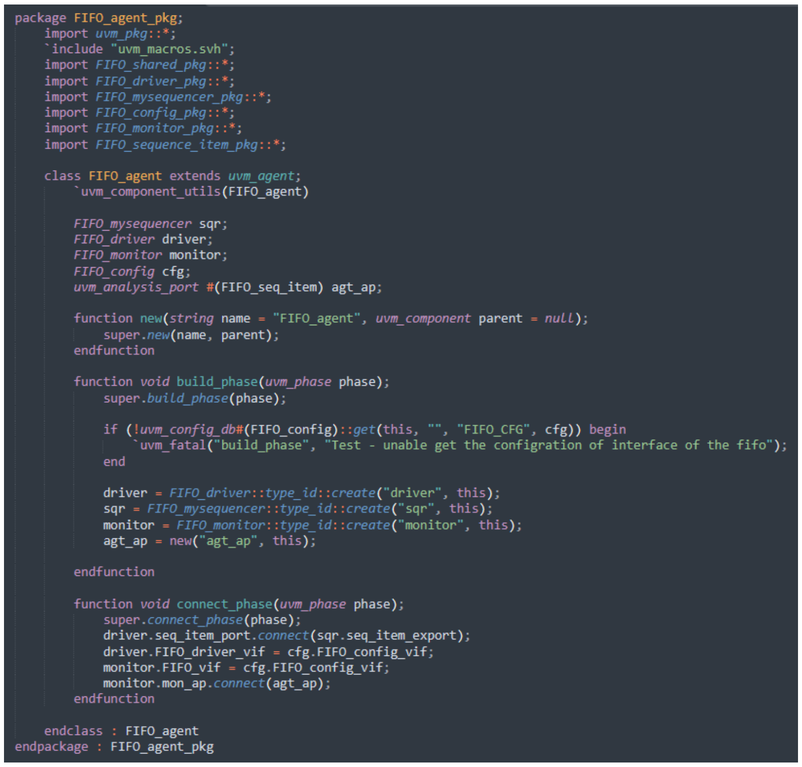
```systemverilog
package FIFO_shared_pkg;
    // signal  will assert at the end of the test
    logic test_finished = 0;

    // Define correct and error counters
    integer correct_count = 0;
    integer error_count = 0;
endpackage
```

Environment code snippet:



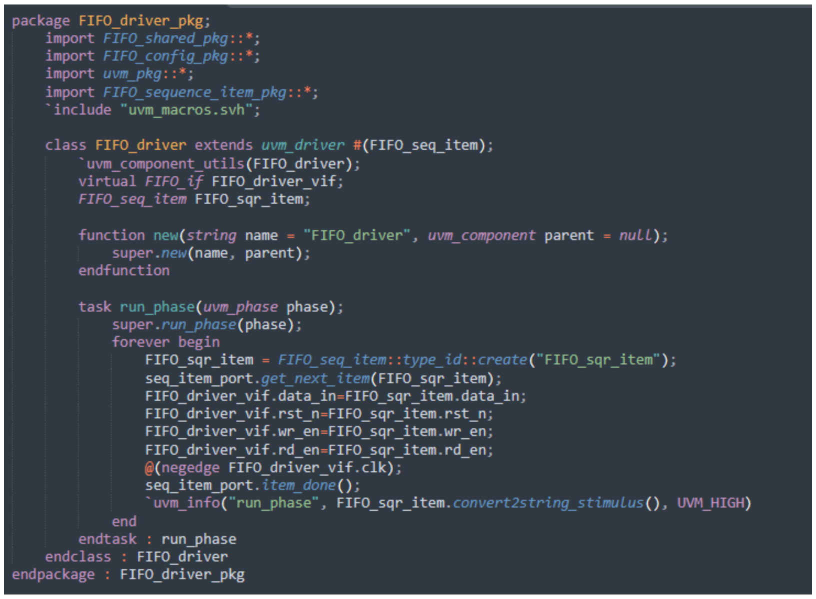
```
package FIFO_env_pkg;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    import FIFO_agent_pkg::*;
    import FIFO_scoreboard_pkg::*;
    import FIFO_coverage_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_env extends  uvm_env;
        `uvm_component_utils(FIFO_env)

        FIFO_agent agt;
        FIFO_coverage cov;
        FIFO_scoreboard sb;

        function new(string name = "FIFO_env", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            agt = FIFO_agent::type_id::create("agt", this);
            cov = FIFO_coverage::type_id::create("cov", this);
            sb = FIFO_scoreboard::type_id::create("sb", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agt.agt_ap.connect(sb.sb_export);
            agt.agt_ap.connect(cov.cov_export);
        endfunction
    endclass : FIFO_env

endpackage
```

Agent code snippet:



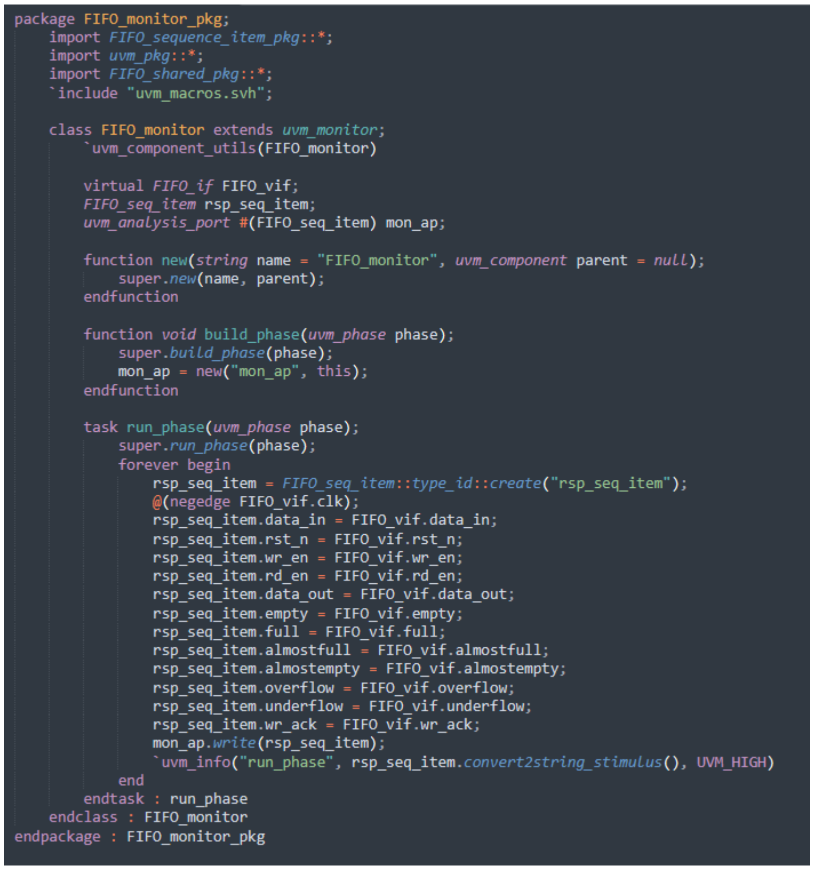
```
package FIFO_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh";
    import FIFO_shared_pkg::*;
    import FIFO_driver_pkg::*;
    import FIFO_mysequencer_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_monitor_pkg::*;
    import FIFO_sequence_item_pkg::*;

    class FIFO_agent extends uvm_agent;
        `uvm_component_utils(FIFO_agent)

        FIFO_mysequencer sqr;
        FIFO_driver driver;
        FIFO_monitor monitor;
        FIFO_config cfg;
        uvm_analysis_port #(FIFO_seq_item) agt_ap;

        function new(string name = "FIFO_agent", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);

            if (!uvm_config_db#(FIFO_config)::get(this, "", "FIFO_CFG", cfg)) begin
                `uvm_fatal("build_phase", "Test - unable get the configration of interface of the fifo");
            end

            driver = FIFO_driver::type_id::create("driver", this);
            sqr = FIFO_mysequencer::type_id::create("sqr", this);
            monitor = FIFO_monitor::type_id::create("monitor", this);
            agt_ap = new("agt_ap", this);

        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            driver.seq_item_port.connect(sqr.seq_item_export);
            driver.FIFO_driver_vif = cfg.FIFO_config_vif;
            monitor.FIFO_vif = cfg.FIFO_config_vif;
            monitor.mon_ap.connect(agt_ap);
        endfunction

    endclass : FIFO_agent
endpackage : FIFO_agent_pkg
```

Driver code snippet:



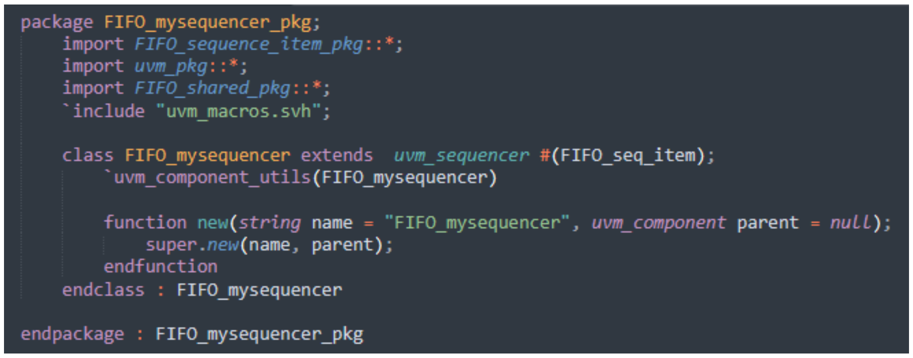
```
package FIFO_driver_pkg;
    import FIFO_shared_pkg::*;
    import FIFO_config_pkg::*;
    import uvm_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_driver extends uvm_driver #(FIFO_seq_item);
        `uvm_component_utils(FIFO_driver);
        virtual FIFO_if FIFO_driver_vif;
        FIFO_seq_item FIFO_sqr_item;

        function new(string name = "FIFO_driver", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                FIFO_sqr_item = FIFO_seq_item::type_id::create("FIFO_sqr_item");
                seq_item_port.get_next_item(FIFO_sqr_item);
                FIFO_driver_vif.data_in=FIFO_sqr_item.data_in;
                FIFO_driver_vif.rst_n=FIFO_sqr_item.rst_n;
                FIFO_driver_vif.wr_en=FIFO_sqr_item.wr_en;
                FIFO_driver_vif.rd_en=FIFO_sqr_item.rd_en;
                @(negedge FIFO_driver_vif.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase", FIFO_sqr_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask : run_phase
    endclass : FIFO_driver
endpackage : FIFO_driver_pkg
```

Monitor code snippet:



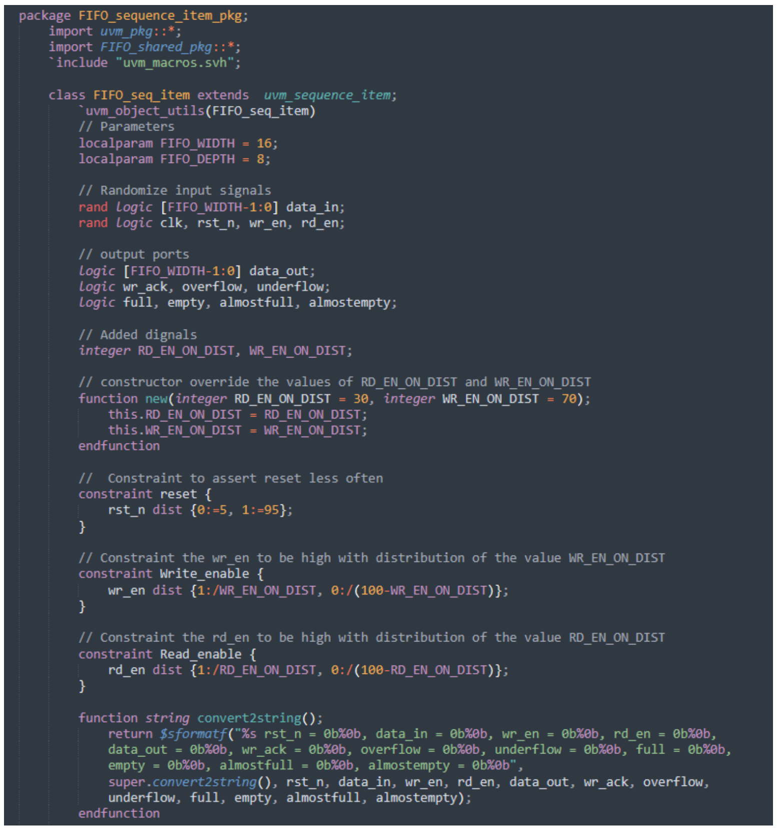
```
package FIFO_monitor_pkg;
    import FIFO_sequence_item_pkg::*;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_monitor extends uvm_monitor;
        `uvm_component_utils(FIFO_monitor)

        virtual FIFO_if FIFO_vif;
        FIFO_seq_item rsp_seq_item;
        uvm_analysis_port #(FIFO_seq_item) mon_ap;

        function new(string name = "FIFO_monitor", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            mon_ap = new("mon_ap", this);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = FIFO_seq_item::type_id::create("rsp_seq_item");
                @(negedge FIFO_vif.clk);
                rsp_seq_item.data_in = FIFO_vif.data_in;
                rsp_seq_item.rst_n = FIFO_vif.rst_n;
                rsp_seq_item.wr_en = FIFO_vif.wr_en;
                rsp_seq_item.rd_en = FIFO_vif.rd_en;
                rsp_seq_item.data_out = FIFO_vif.data_out;
                rsp_seq_item.empty = FIFO_vif.empty;
                rsp_seq_item.full = FIFO_vif.full;
                rsp_seq_item.almostfull = FIFO_vif.almostfull;
                rsp_seq_item.almostempty = FIFO_vif.almostempty;
                rsp_seq_item.overflow = FIFO_vif.overflow;
                rsp_seq_item.underflow = FIFO_vif.underflow;
                rsp_seq_item.wr_ack = FIFO_vif.wr_ack;
                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase", rsp_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask : run_phase
    endclass : FIFO_monitor
endpackage : FIFO_monitor_pkg
```

Sequencer code snippet:

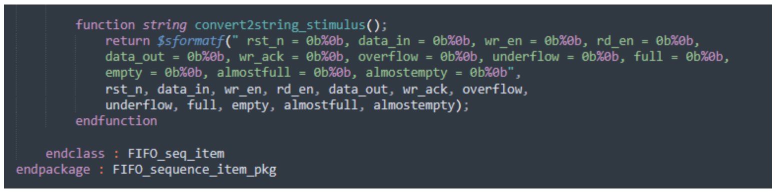


```
package FIFO_mysequencer_pkg;
    import FIFO_sequence_item_pkg::*;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_mysequencer extends  uvm_sequencer #(FIFO_seq_item);
        `uvm_component_utils(FIFO_mysequencer)

        function new(string name = "FIFO_mysequencer", uvm_component parent = null);
            super.new(name, parent);
        endfunction
    endclass : FIFO_mysequencer

endpackage : FIFO_mysequencer_pkg
```

Sequence Item code snippet:

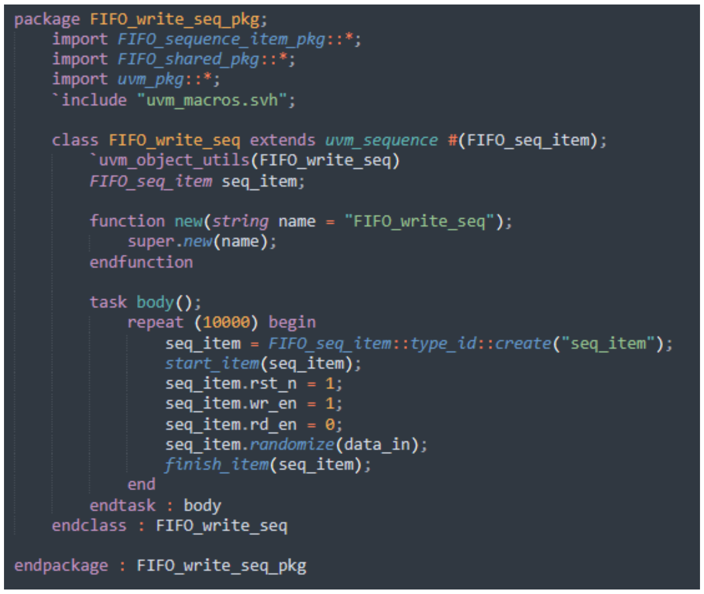


```
package FIFO_sequence_item_pkg;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_seq_item extends  uvm_sequence_item;
        `uvm_object_utils(FIFO_seq_item)
        // Parameters
        localparam FIFO_WIDTH = 16;
        localparam FIFO_DEPTH = 8;

        // Randomize input signals
        rand logic [FIFO_WIDTH-1:0] data_in;
        rand logic clk, rst_n, wr_en, rd_en;

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

        // Added dignals
        integer RD_EN_ON_DIST, WR_EN_ON_DIST;

        // constructor override the values of RD_EN_ON_DIST and WR_EN_ON_DIST
        function new(integer RD_EN_ON_DIST = 30, integer WR_EN_ON_DIST = 70);
            this.RD_EN_ON_DIST = RD_EN_ON_DIST;
            this.WR_EN_ON_DIST = WR_EN_ON_DIST;
        endfunction

        //  Constraint to assert reset less often
        constraint reset {
            rst_n dist {0:=5, 1:=95};
        }

        // Constraint the wr_en to be high with distribution of the value WR_EN_ON_DIST
        constraint Write_enable {
            wr_en dist {1:/WR_EN_ON_DIST, 0:/(100-WR_EN_ON_DIST)};
        }

        // Constraint the rd_en to be high with distribution of the value RD_EN_ON_DIST
        constraint Read_enable {
            rd_en dist {1:/RD_EN_ON_DIST, 0:/(100-RD_EN_ON_DIST)};
        }

        function string convert2string();
            return $sformatf("%s rst_n = 0b%0b, data_in = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b,
            data_out = 0b%0b, wr_ack = 0b%0b, overflow = 0b%0b, underflow = 0b%0b, full = 0b%0b,
            empty = 0b%0b, almostfull = 0b%0b, almostempty = 0b%0b",
            super.convert2string(), rst_n, data_in, wr_en, rd_en, data_out, wr_ack, overflow,
            underflow, full, empty, almostfull, almostempty);
        endfunction
```



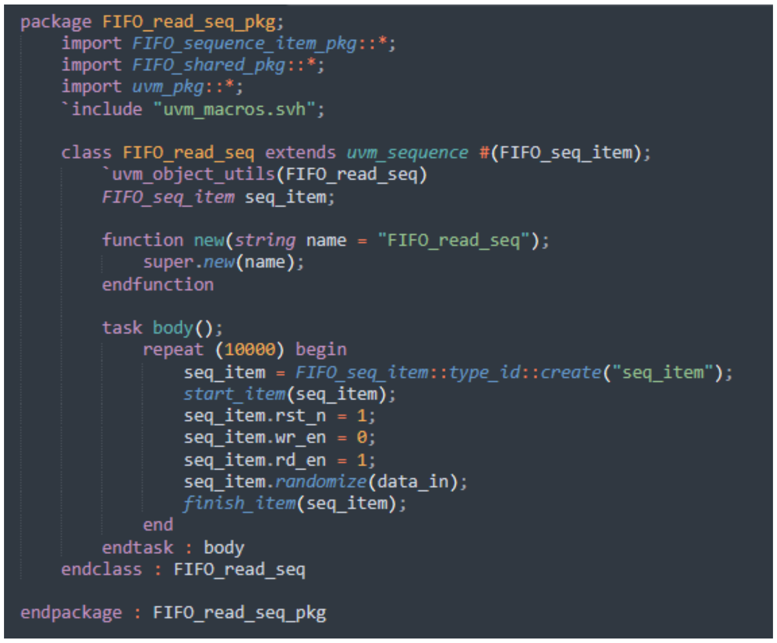
```
        function string convert2string_stimulus();
            return $sformatf(" rst_n = 0b%0b, data_in = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b,
            data_out = 0b%0b, wr_ack = 0b%0b, overflow = 0b%0b, underflow = 0b%0b, full = 0b%0b,
            empty = 0b%0b, almostfull = 0b%0b, almostempty = 0b%0b",
            rst_n, data_in, wr_en, rd_en, data_out, wr_ack, overflow,
            underflow, full, empty, almostfull, almostempty);
        endfunction

    endclass : FIFO_seq_item
endpackage : FIFO_sequence_item_pkg
```

Write Sequence code snippet:



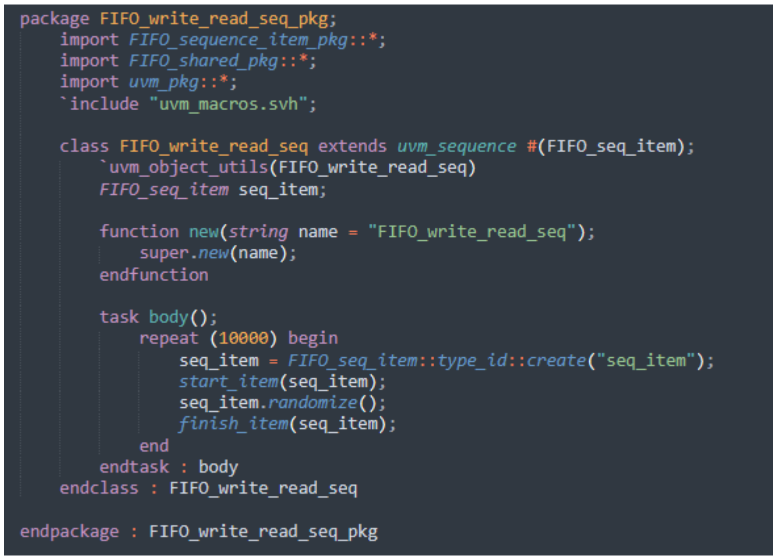
```
package FIFO_write_seq_pkg;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_write_seq extends uvm_sequence #(FIFO_seq_item);
        `uvm_object_utils(FIFO_write_seq)
        FIFO_seq_item seq_item;

        function new(string name = "FIFO_write_seq");
            super.new(name);
        endfunction

        task body();
            repeat (10000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n = 1;
                seq_item.wr_en = 1;
                seq_item.rd_en = 0;
                seq_item.randomize(data_in);
                finish_item(seq_item);
            end
        endtask : body
    endclass : FIFO_write_seq

endpackage : FIFO_write_seq_pkg
```

Read Sequence code snippet:



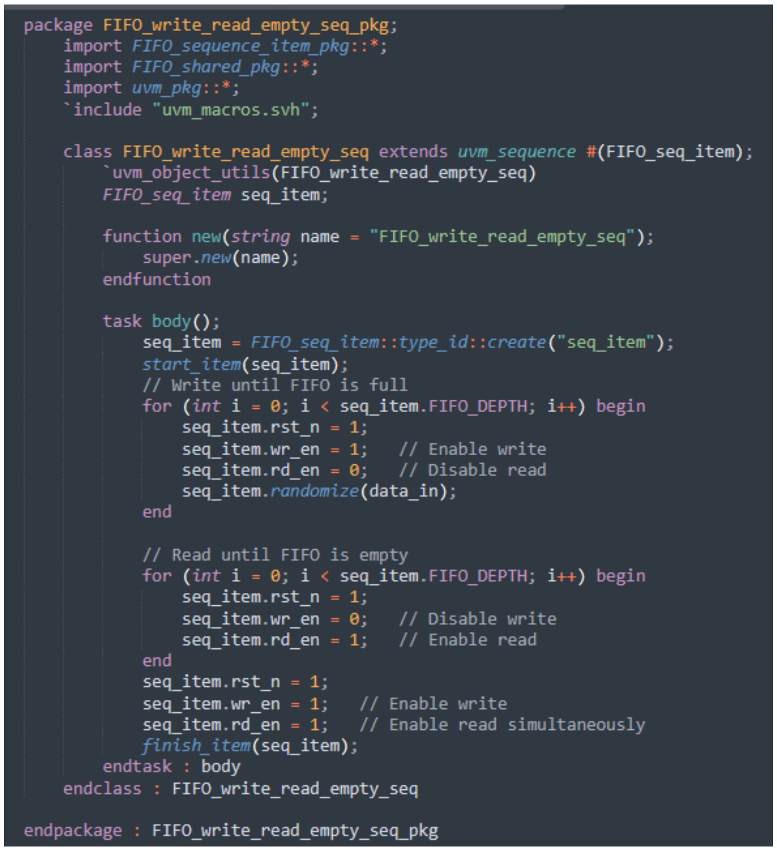
```
package FIFO_read_seq_pkg;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_read_seq extends uvm_sequence #(FIFO_seq_item);
        `uvm_object_utils(FIFO_read_seq)
        FIFO_seq_item seq_item;

        function new(string name = "FIFO_read_seq");
            super.new(name);
        endfunction

        task body();
            repeat (10000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n = 1;
                seq_item.wr_en = 0;
                seq_item.rd_en = 1;
                seq_item.randomize(data_in);
                finish_item(seq_item);
            end
        endtask : body
    endclass : FIFO_read_seq

endpackage : FIFO_read_seq_pkg
```

Write Read Sequence code snippet:



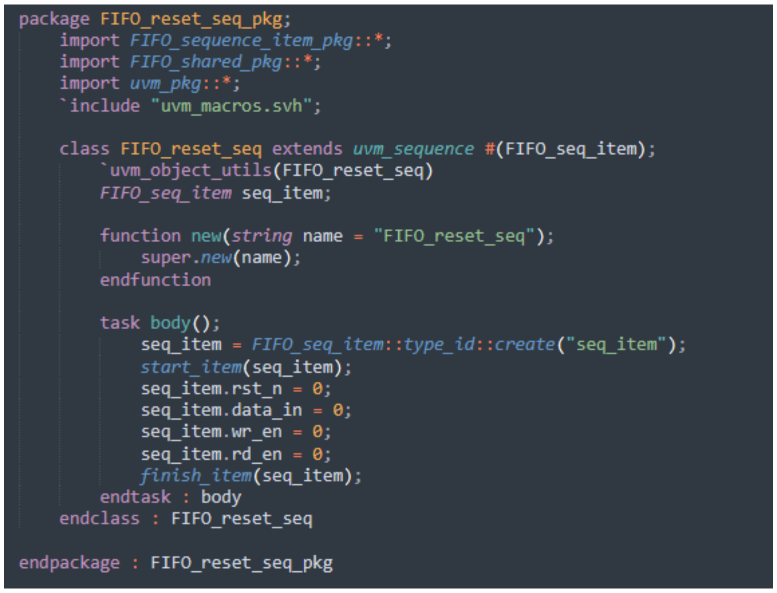
```
package FIFO_write_read_seq_pkg;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_write_read_seq extends uvm_sequence #(FIFO_seq_item);
        `uvm_object_utils(FIFO_write_read_seq)
        FIFO_seq_item seq_item;

        function new(string name = "FIFO_write_read_seq");
            super.new(name);
        endfunction

        task body();
            repeat (10000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.randomize();
                finish_item(seq_item);
            end
        endtask : body
    endclass : FIFO_write_read_seq

endpackage : FIFO_write_read_seq_pkg
```

Write Read Empty Sequence code snippet:



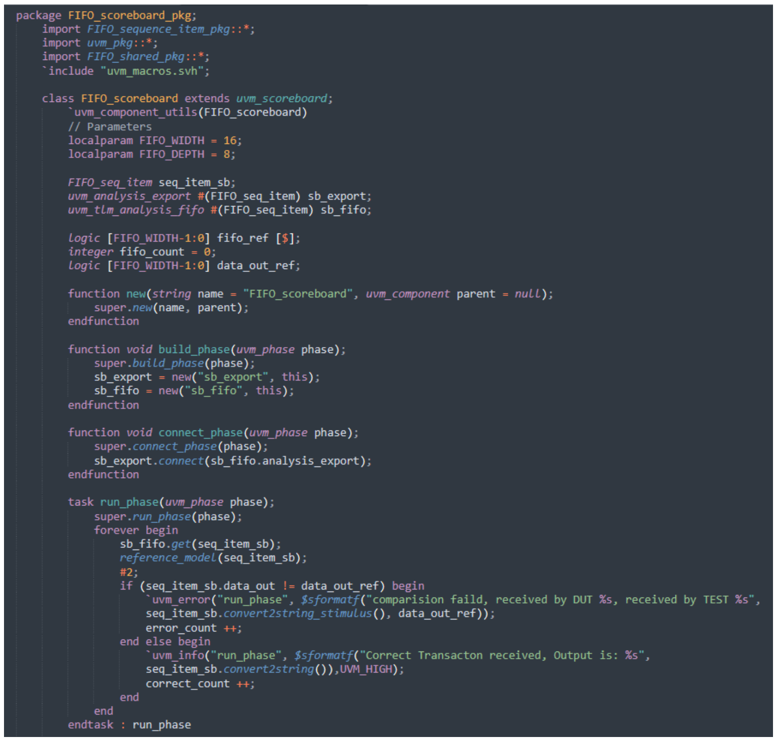
```
package FIFO_write_read_empty_seq_pkg;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_write_read_empty_seq extends uvm_sequence #(FIFO_seq_item);
        `uvm_object_utils(FIFO_write_read_empty_seq)
        FIFO_seq_item seq_item;

        function new(string name = "FIFO_write_read_empty_seq");
            super.new(name);
        endfunction

        task body();
            seq_item = FIFO_seq_item::type_id::create("seq_item");
            start_item(seq_item);
            // Write until FIFO is full
            for (int i = 0; i < seq_item.FIFO_DEPTH; i++) begin
                seq_item.rst_n = 1;
                seq_item.wr_en = 1;   // Enable write
                seq_item.rd_en = 0;   // Disable read
                seq_item.randomize(data_in);
            end

            // Read until FIFO is empty
            for (int i = 0; i < seq_item.FIFO_DEPTH; i++) begin
                seq_item.rst_n = 1;
                seq_item.wr_en = 0;   // Disable write
                seq_item.rd_en = 1;   // Enable read
            end
            seq_item.rst_n = 1;
            seq_item.wr_en = 1;   // Enable write
            seq_item.rd_en = 1;   // Enable read simultaneously
            finish_item(seq_item);
        endtask : body
    endclass : FIFO_write_read_empty_seq

endpackage : FIFO_write_read_empty_seq_pkg
```

Reset Sequence code snippet:

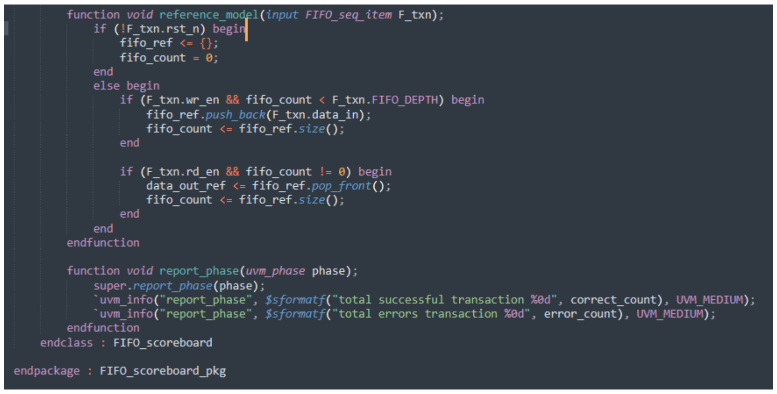


```systemverilog
package FIFO_reset_seq_pkg;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_reset_seq extends uvm_sequence #(FIFO_seq_item);
        `uvm_object_utils(FIFO_reset_seq)
        FIFO_seq_item seq_item;

        function new(string name = "FIFO_reset_seq");
            super.new(name);
        endfunction

        task body();
            seq_item = FIFO_seq_item::type_id::create("seq_item");
            start_item(seq_item);
            seq_item.rst_n = 0;
            seq_item.data_in = 0;
            seq_item.wr_en = 0;
            seq_item.rd_en = 0;
            finish_item(seq_item);
        endtask : body
    endclass : FIFO_reset_seq

endpackage : FIFO_reset_seq_pkg
```

Scoreboard code snippet:

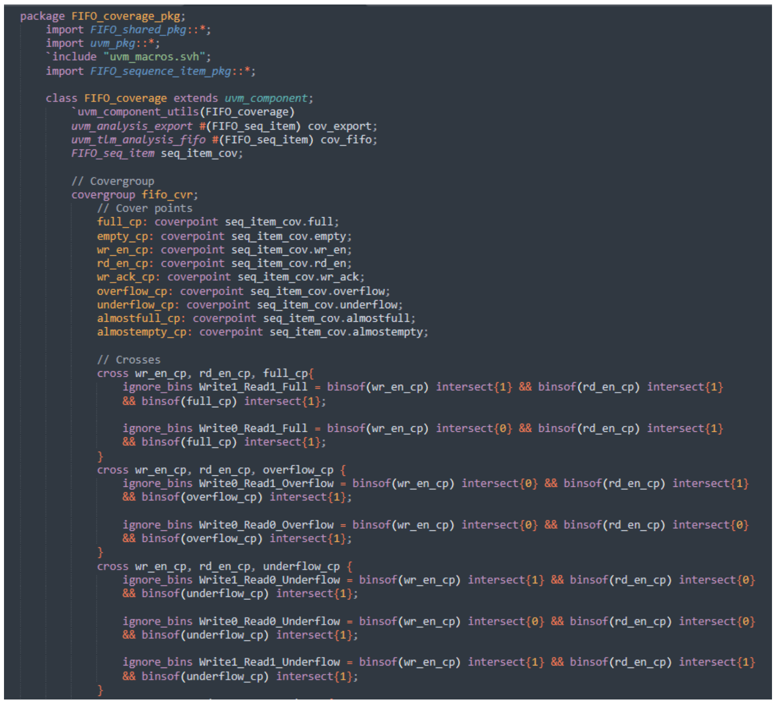


```systemverilog
package FIFO_scoreboard_pkg;
    import FIFO_sequence_item_pkg::*;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(FIFO_scoreboard)
        // Parameters
        localparam FIFO_WIDTH = 16;
        localparam FIFO_DEPTH = 8;

        FIFO_seq_item seq_item_sb;
        uvm_analysis_export #(FIFO_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(FIFO_seq_item) sb_fifo;

        logic [FIFO_WIDTH-1:0] fifo_ref [$];
        integer fifo_count = 0;
        logic [FIFO_WIDTH-1:0] data_out_ref;

        function new(string name = "FIFO_scoreboard", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            sb_export = new("sb_export", this);
            sb_fifo = new("sb_fifo", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            sb_export.connect(sb_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                reference_model(seq_item_sb);
                #2;
                if (seq_item_sb.data_out != data_out_ref) begin
                    `uvm_error("run_phase", $sformatf("comparision faild, received by DUT %s, received by TEST %s",
                    seq_item_sb.convert2string_stimulus(), data_out_ref));
                    error_count ++;
                end else begin
                    `uvm_info("run_phase", $sformatf("Correct Transacton received, Output is: %s",
                    seq_item_sb.convert2string()),UVM_HIGH);
                    correct_count ++;
                end
            end
        endtask : run_phase
```

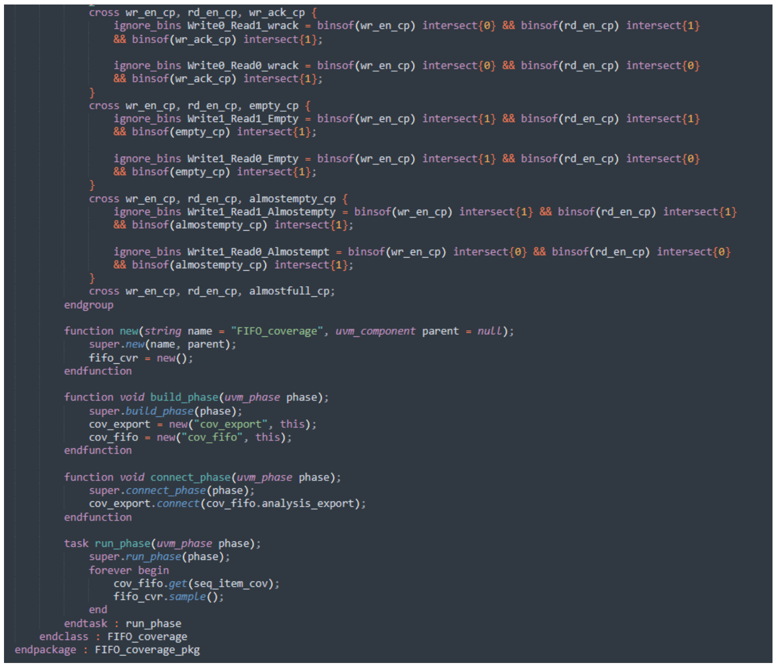


```
        function void reference_model(input FIFO_seq_item F_txn);
            if (!F_txn.rst_n) begin
                fifo_ref <= {};
                fifo_count = 0;
            end
            else begin
                if (F_txn.wr_en && fifo_count < F_txn.FIFO_DEPTH) begin
                    fifo_ref.push_back(F_txn.data_in);
                    fifo_count <= fifo_ref.size();
                end

                if (F_txn.rd_en && fifo_count != 0) begin
                    data_out_ref <= fifo_ref.pop_front();
                    fifo_count <= fifo_ref.size();
                end
            end
        endfunction

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("total successful transaction %0d", correct_count), UVM_MEDIUM);
            `uvm_info("report_phase", $sformatf("total errors transaction %0d", error_count), UVM_MEDIUM);
        endfunction
    endclass : FIFO_scoreboard

endpackage : FIFO_scoreboard_pkg
```

Coverage code snippet:

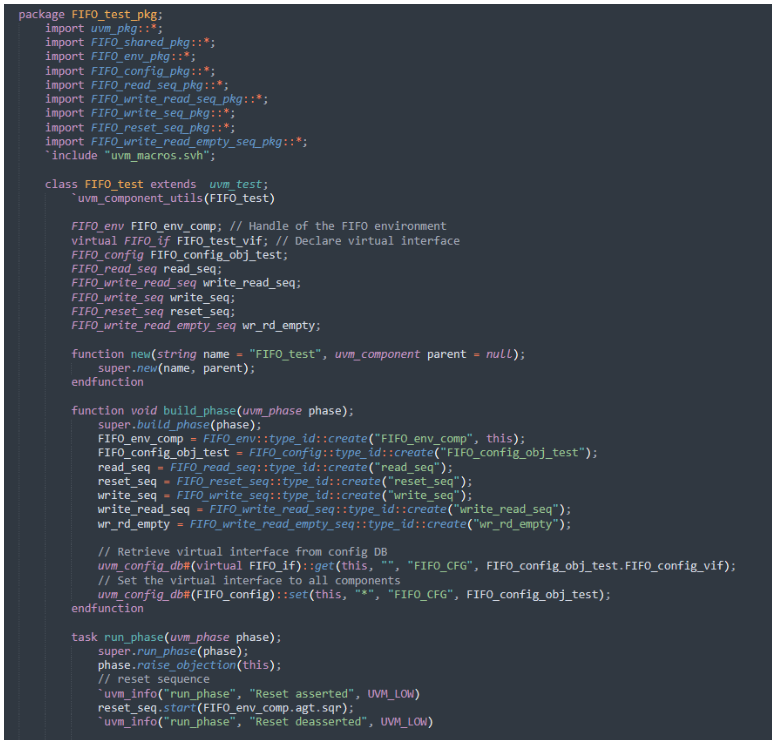


```
package FIFO_coverage_pkg;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";
    import FIFO_sequence_item_pkg::*;

    class FIFO_coverage extends uvm_component;
        `uvm_component_utils(FIFO_coverage)
        uvm_analysis_export #(FIFO_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(FIFO_seq_item) cov_fifo;
        FIFO_seq_item seq_item_cov;

        // Covergroup
        covergroup fifo_cvr;
            // Cover points
            full_cp: coverpoint seq_item_cov.full;
            empty_cp: coverpoint seq_item_cov.empty;
            wr_en_cp: coverpoint seq_item_cov.wr_en;
            rd_en_cp: coverpoint seq_item_cov.rd_en;
            wr_ack_cp: coverpoint seq_item_cov.wr_ack;
            overflow_cp: coverpoint seq_item_cov.overflow;
            underflow_cp: coverpoint seq_item_cov.underflow;
            almostfull_cp: coverpoint seq_item_cov.almostfull;
            almostempty_cp: coverpoint seq_item_cov.almostempty;

            // Crosses
            cross wr_en_cp, rd_en_cp, full_cp{
                ignore_bins Write1_Read1_Full = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{1}
                && binsof(full_cp) intersect{1};

                ignore_bins Write0_Read1_Full = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{1}
                && binsof(full_cp) intersect{1};
            }
            cross wr_en_cp, rd_en_cp, overflow_cp {
                ignore_bins Write0_Read1_Overflow = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{1}
                && binsof(overflow_cp) intersect{1};

                ignore_bins Write0_Read0_Overflow = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{0}
                && binsof(overflow_cp) intersect{1};
            }
            cross wr_en_cp, rd_en_cp, underflow_cp {
                ignore_bins Write1_Read0_Underflow = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{0}
                && binsof(underflow_cp) intersect{1};

                ignore_bins Write0_Read0_Underflow = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{0}
                && binsof(underflow_cp) intersect{1};

                ignore_bins Write1_Read1_Underflow = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{1}
                && binsof(underflow_cp) intersect{1};
            }
```

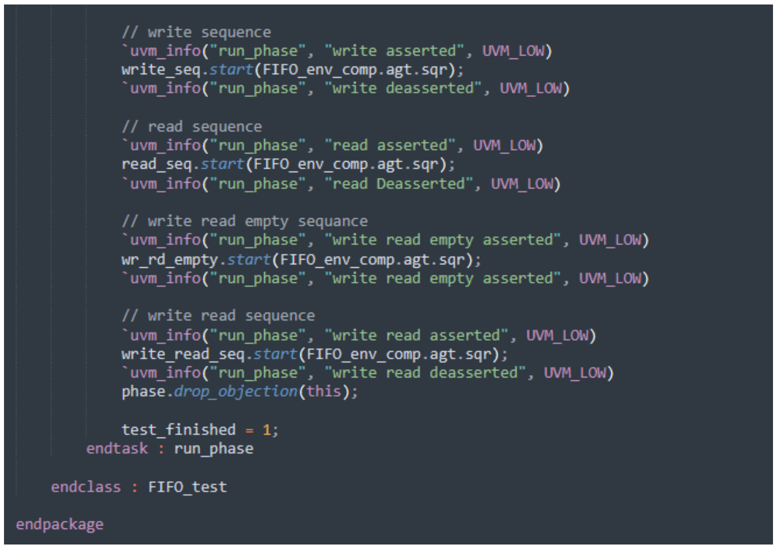


```
            cross wr_en_cp, rd_en_cp, wr_ack_cp {
                ignore_bins Write0_Read1_wrack = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{1}
                && binsof(wr_ack_cp) intersect{1};

                ignore_bins Write0_Read0_wrack = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{0}
                && binsof(wr_ack_cp) intersect{1};
            }
            cross wr_en_cp, rd_en_cp, empty_cp {
                ignore_bins Write1_Read1_Empty = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{1}
                && binsof(empty_cp) intersect{1};

                ignore_bins Write1_Read0_Empty = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{0}
                && binsof(empty_cp) intersect{1};
            }
            cross wr_en_cp, rd_en_cp, almostempty_cp {
                ignore_bins Write1_Read1_Almostempty = binsof(wr_en_cp) intersect{1} && binsof(rd_en_cp) intersect{1}
                && binsof(almostempty_cp) intersect{1};

                ignore_bins Write1_Read0_Almostempt = binsof(wr_en_cp) intersect{0} && binsof(rd_en_cp) intersect{0}
                && binsof(almostempty_cp) intersect{1};
            }
            cross wr_en_cp, rd_en_cp, almostfull_cp;
        endgroup

        function new(string name = "FIFO_coverage", uvm_component parent = null);
            super.new(name, parent);
            fifo_cvr = new();
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            cov_export = new("cov_export", this);
            cov_fifo = new("cov_fifo", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cov_export.connect(cov_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item_cov);
                fifo_cvr.sample();
            end
        endtask : run_phase
    endclass : FIFO_coverage
endpackage : FIFO_coverage_pkg
```

Test code snippet:

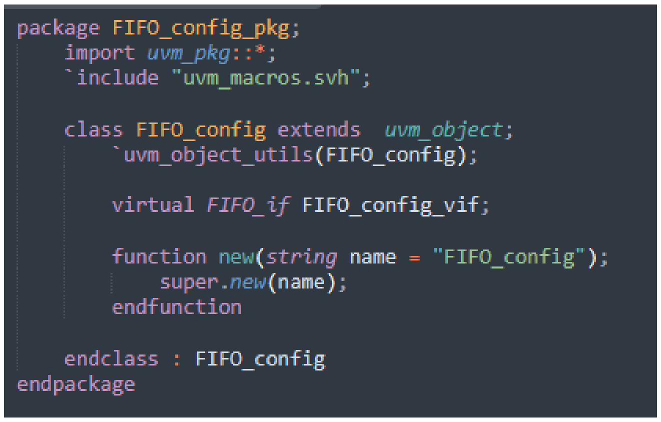


```
package FIFO_test_pkg;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    import FIFO_env_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_read_seq_pkg::*;
    import FIFO_write_read_seq_pkg::*;
    import FIFO_write_seq_pkg::*;
    import FIFO_reset_seq_pkg::*;
    import FIFO_write_read_empty_seq_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_test extends  uvm_test;
        `uvm_component_utils(FIFO_test)

        FIFO_env FIFO_env_comp; // Handle of the FIFO environment
        virtual FIFO_if FIFO_test_vif; // Declare virtual interface
        FIFO_config FIFO_config_obj_test;
        FIFO_read_seq read_seq;
        FIFO_write_read_seq write_read_seq;
        FIFO_write_seq write_seq;
        FIFO_reset_seq reset_seq;
        FIFO_write_read_empty_seq wr_rd_empty;

        function new(string name = "FIFO_test", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            FIFO_env_comp = FIFO_env::type_id::create("FIFO_env_comp", this);
            FIFO_config_obj_test = FIFO_config::type_id::create("FIFO_config_obj_test");
            read_seq = FIFO_read_seq::type_id::create("read_seq");
            reset_seq = FIFO_reset_seq::type_id::create("reset_seq");
            write_seq = FIFO_write_seq::type_id::create("write_seq");
            write_read_seq = FIFO_write_read_seq::type_id::create("write_read_seq");
            wr_rd_empty = FIFO_write_read_empty_seq::type_id::create("wr_rd_empty");

            // Retrieve virtual interface from config DB
            uvm_config_db#(virtual FIFO_if)::get(this, "", "FIFO_CFG", FIFO_config_obj_test.FIFO_config_vif);
            // Set the virtual interface to all components
            uvm_config_db#(FIFO_config)::set(this, "*", "FIFO_CFG", FIFO_config_obj_test);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            // reset sequence
            `uvm_info("run_phase", "Reset asserted", UVM_LOW)
            reset_seq.start(FIFO_env_comp.agt.sqr);
            `uvm_info("run_phase", "Reset deasserted", UVM_LOW)
```



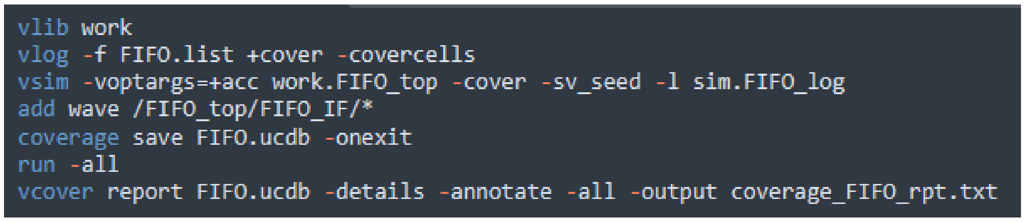
```
            // write sequence
            `uvm_info("run_phase", "write asserted", UVM_LOW)
            write_seq.start(FIFO_env_comp.agt.sqr);
            `uvm_info("run_phase", "write deasserted", UVM_LOW)

            // read sequence
            `uvm_info("run_phase", "read asserted", UVM_LOW)
            read_seq.start(FIFO_env_comp.agt.sqr);
            `uvm_info("run_phase", "read Deasserted", UVM_LOW)

            // write read empty sequance
            `uvm_info("run_phase", "write read empty asserted", UVM_LOW)
            wr_rd_empty.start(FIFO_env_comp.agt.sqr);
            `uvm_info("run_phase", "write read empty asserted", UVM_LOW)

            // write read sequence
            `uvm_info("run_phase", "write read asserted", UVM_LOW)
            write_read_seq.start(FIFO_env_comp.agt.sqr);
            `uvm_info("run_phase", "write read deasserted", UVM_LOW)
            phase.drop_objection(this);

            test_finished = 1;
        endtask : run_phase

    endclass : FIFO_test

endpackage
```

Configuration object code snippet:



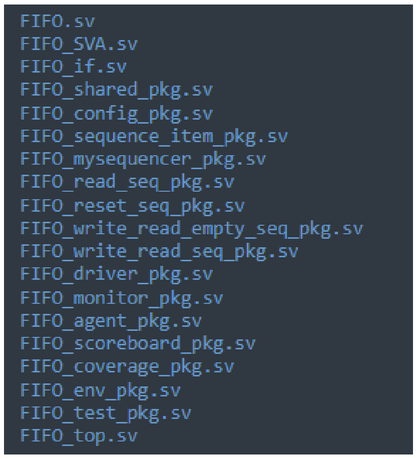
```
package FIFO_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class FIFO_config extends  uvm_object;
        `uvm_object_utils(FIFO_config);

        virtual FIFO_if FIFO_config_vif;

        function new(string name = "FIFO_config");
            super.new(name);
        endfunction

    endclass : FIFO_config
endpackage
```

DO File:



```
vlib work
vlog -f FIFO.list +cover -covercells
vsim -voptargs=+acc work.FIFO_top -cover -sv_seed -l sim.FIFO_log
add wave /FIFO_top/FIFO_IF/*
coverage save FIFO.ucdb -onexit
run -all
vcover report FIFO.ucdb -details -annotate -all -output coverage_FIFO_rpt.txt
```

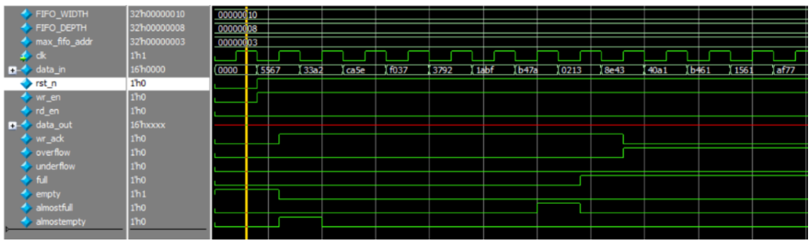
ALSU list of files:



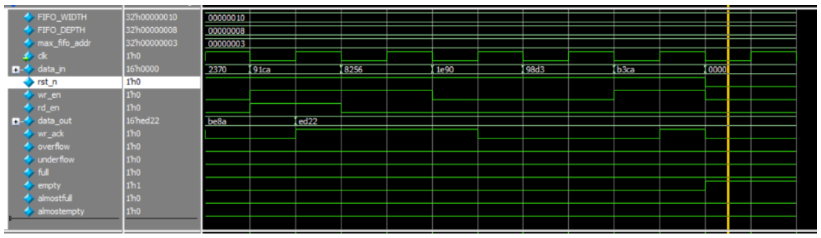
```
FIFO.sv
FIFO_SVA.sv
FIFO_if.sv
FIFO_shared_pkg.sv
FIFO_config_pkg.sv
FIFO_sequence_item_pkg.sv
FIFO_mysequencer_pkg.sv
FIFO_read_seq_pkg.sv
FIFO_reset_seq_pkg.sv
FIFO_write_read_empty_seq_pkg.sv
FIFO_write_read_seq_pkg.sv
FIFO_driver_pkg.sv
FIFO_monitor_pkg.sv
FIFO_agent_pkg.sv
FIFO_scoreboard_pkg.sv
FIFO_coverage_pkg.sv
FIFO_env_pkg.sv
FIFO_test_pkg.sv
FIFO_top.sv
```

Waveform snippets:

a) Reset seq:

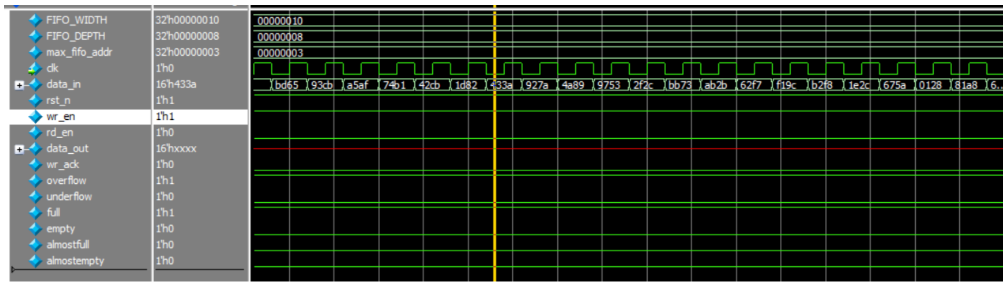


```
FIFO_WIDTH      32'h00000010
FIFO_DEPTH      32'h00000008
max_fifo_addr   32'h00000003
clk             1'h1
data_in         16'h0000
rst_n           1'h0
wr_en           1'h0
rd_en           1'h0
data_out        16'hxxxx
wr_ack          1'h0
overflow        1'h0
underflow       1'h0
full            1'h0
empty           1'h1
almostfull      1'h0
almostempty     1'h0
```



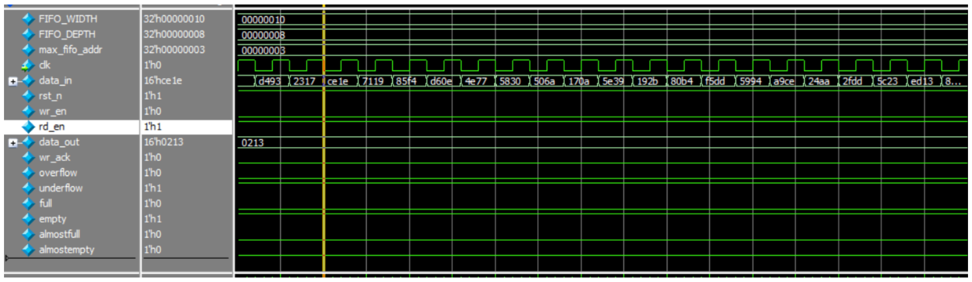
```
FIFO_WIDTH      32'h00000010
FIFO_DEPTH      32'h00000008
max_fifo_addr   32'h00000003
clk             1'h0
data_in         16'h0000
rst_n           1'h0
wr_en           1'h0
rd_en           1'h0
data_out        16'hed22
wr_ack          1'h0
overflow        1'h0
underflow       1'h0
full            1'h0
empty           1'h1
almostfull      1'h0
almostempty     1'h0
```

b) Write seq:



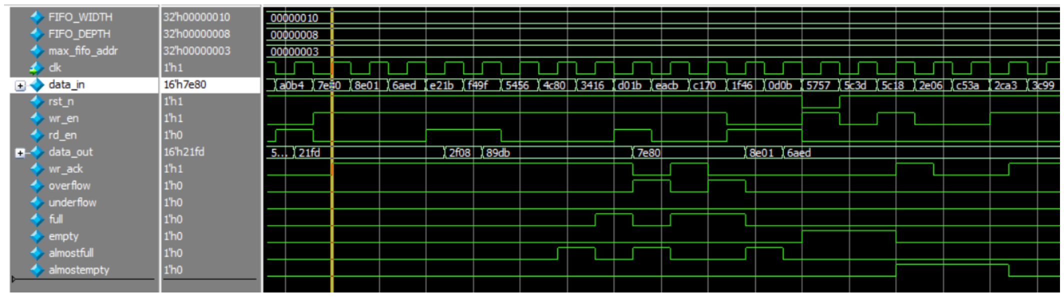
```
FIFO_WIDTH      32'h00000010
FIFO_DEPTH      32'h00000008
max_fifo_addr   32'h00000003
clk             1'h0
data_in         16'h433a
rst_n           1'h1
wr_en           1'h1
rd_en           1'h0
data_out        16'hxxxx
wr_ack          1'h0
overflow        1'h1
underflow       1'h0
full            1'h1
empty           1'h0
almostfull      1'h0
almostempty     1'h0
```

c) Read seq:

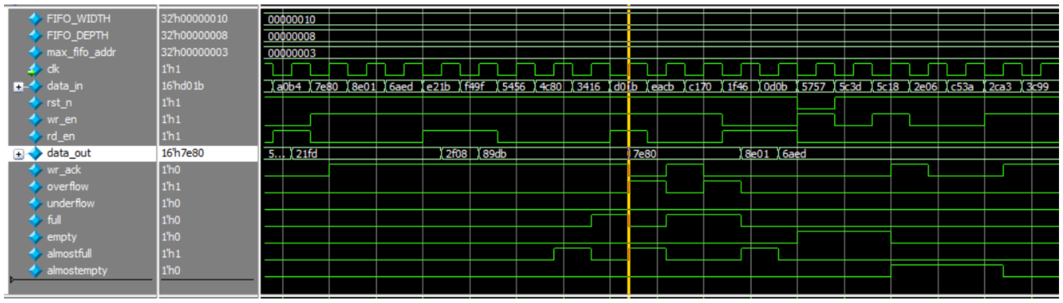


FIFO_WIDTH	32'h00000010
FIFO_DEPTH	32'h00000008
max_fifo_addr	32'h00000003
clk	1'h0
data_in	16'hce1e
rst_n	1'h1
wr_en	1'h0
rd_en	1'h1
data_out	16'h0213
wr_ack	1'h0
overflow	1'h0
underflow	1'h1
full	1'h0
empty	1'h1
almostfull	1'h0
almostempty	1'h0

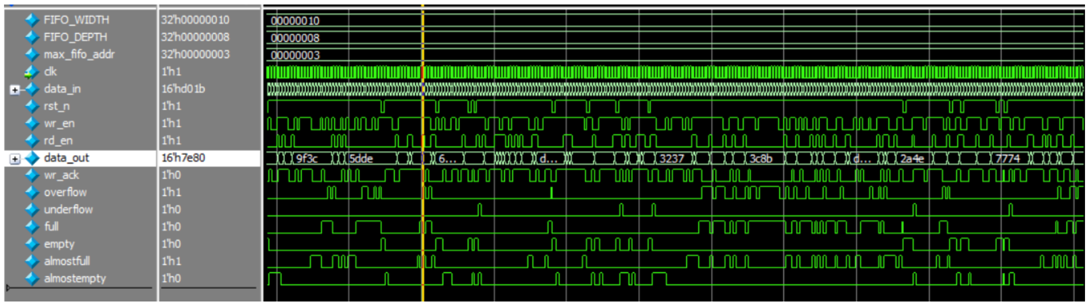
d) Write_Read seq:



FIFO_WIDTH	32'h00000010
FIFO_DEPTH	32'h00000008
max_fifo_addr	32'h00000003
clk	1'h1
data_in	16'h7e80
rst_n	1'h1
wr_en	1'h1
rd_en	1'h0
data_out	16'h21fd
wr_ack	1'h1
overflow	1'h0
underflow	1'h0
full	1'h0
empty	1'h0
almostfull	1'h0
almostempty	1'h0

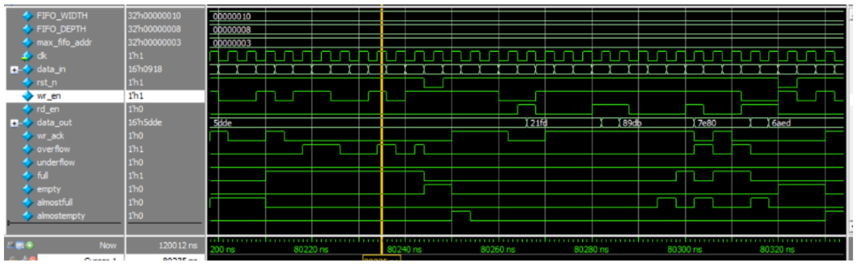


FIFO_WIDTH	32'h00000010
FIFO_DEPTH	32'h00000008
max_fifo_addr	32'h00000003
clk	1'h1
data_in	16'hd01b
rst_n	1'h1
wr_en	1'h1
rd_en	1'h1
data_out	16'h7e80
wr_ack	1'h0
overflow	1'h1
underflow	1'h0
full	1'h0
empty	1'h0
almostfull	1'h1
almostempty	1'h0



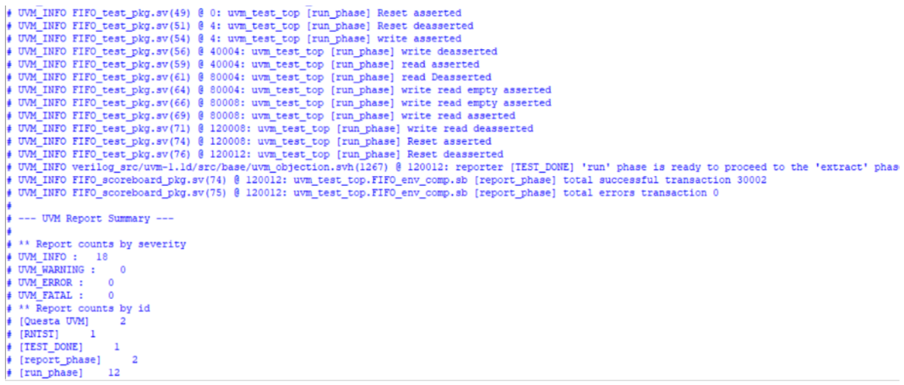
FIFO_WIDTH	32'h00000010
FIFO_DEPTH	32'h00000008
max_fifo_addr	32'h00000003
clk	1'h1
data_in	16'hd01b
rst_n	1'h1
wr_en	1'h1
rd_en	1'h1
data_out	16'h7e80
wr_ack	1'h0
overflow	1'h1
underflow	1'h0
full	1'h0
empty	1'h0
almostfull	1'h1
almostempty	1'h0

e) Write_Read_Empty seq:



FIFO_WIDTH	32'h00000010
FIFO_DEPTH	32'h00000008
max_fifo_addr	32'h00000003
clk	1'h1
data_in	16'h0918
rst_n	1'h1
wr_en	1'h1
rd_en	1'h0
data_out	16'h5dde
wr_ack	1'h0
overflow	1'h1
underflow	1'h0
full	1'h1
empty	1'h0
almostfull	1'h0
almostempty	1'h0
Now	120012 ns

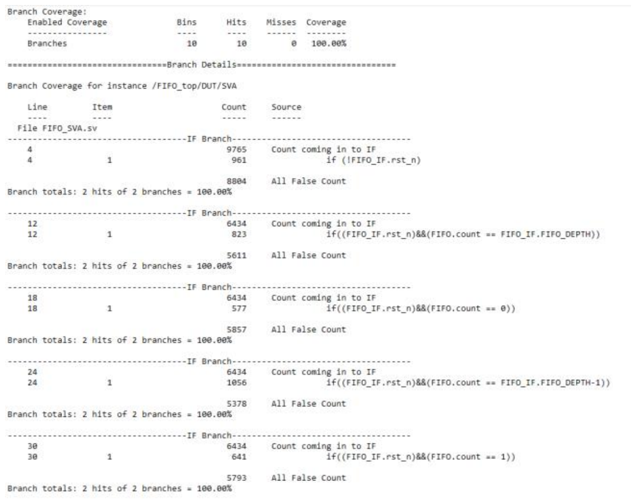
Transcript snippet:



```
# UVM_INFO FIFO_test_pkg.sv(49) @ 0: uvm_test_top [run_phase] Reset asserted
# UVM_INFO FIFO_test_pkg.sv(51) @ 4: uvm_test_top [run_phase] Reset deasserted
# UVM_INFO FIFO_test_pkg.sv(54) @ 4: uvm_test_top [run_phase] write asserted
# UVM_INFO FIFO_test_pkg.sv(56) @ 40004: uvm_test_top [run_phase] write deasserted
# UVM_INFO FIFO_test_pkg.sv(59) @ 40004: uvm_test_top [run_phase] read asserted
# UVM_INFO FIFO_test_pkg.sv(61) @ 80004: uvm_test_top [run_phase] read Deasserted
# UVM_INFO FIFO_test_pkg.sv(64) @ 80004: uvm_test_top [run_phase] write read empty asserted
# UVM_INFO FIFO_test_pkg.sv(66) @ 80008: uvm_test_top [run_phase] write read empty asserted
# UVM_INFO FIFO_test_pkg.sv(69) @ 80008: uvm_test_top [run_phase] write read asserted
# UVM_INFO FIFO_test_pkg.sv(71) @ 120008: uvm_test_top [run_phase] write read deasserted
# UVM_INFO FIFO_test_pkg.sv(74) @ 120008: uvm_test_top [run_phase] Reset asserted
# UVM_INFO FIFO_test_pkg.sv(76) @ 120012: uvm_test_top [run_phase] Reset deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 120012: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phas
# UVM_INFO FIFO_scoreboard_pkg.sv(74) @ 120012: uvm_test_top.FIFO_env_comp.sb [report_phase] total successful transaction 30002
# UVM_INFO FIFO_scoreboard_pkg.sv(75) @ 120012: uvm_test_top.FIFO_env_comp.sb [report_phase] total errors transaction 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   18
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
# [run_phase]    12
```

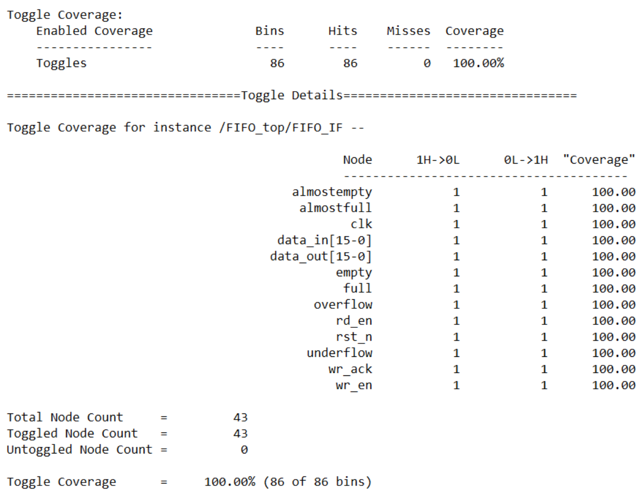
Code coverage report text file:

a) Branch coverage:



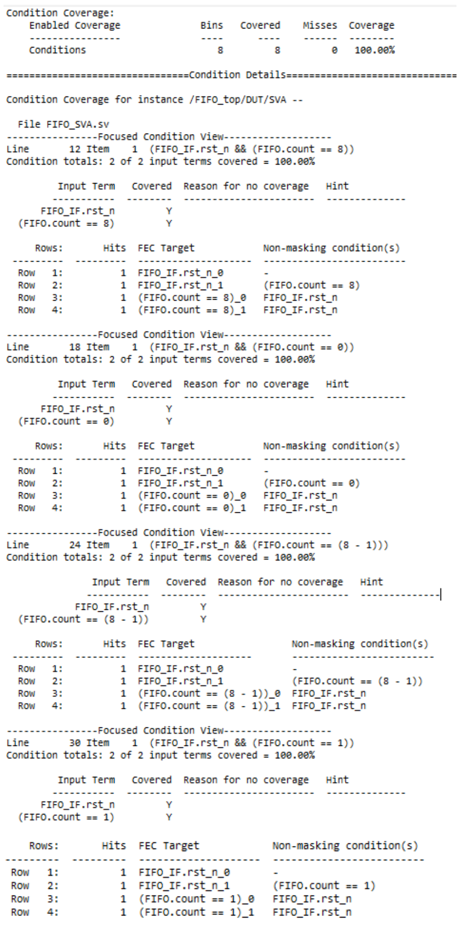
```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for instance /FIFO_top/DUT/SVA

    Line         Item                      Count     Source 
    ----         ----                      -----     ------ 
  File FIFO_SVA.sv
------------------------------------IF Branch------------------------------------
    4                                       9765     Count coming in to IF
    4               1                        961                 if (!FIFO_IF.rst_n)

                                            8804     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    12                                      6434     Count coming in to IF
    12              1                        823                 if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH))

                                            5611     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    18                                      6434     Count coming in to IF
    18              1                        577                 if((FIFO_IF.rst_n)&&(FIFO.count == 0))

                                            5857     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    24                                      6434     Count coming in to IF
    24              1                       1056                 if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH-1))

                                            5378     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    30                                      6434     Count coming in to IF
    30              1                        641                 if((FIFO_IF.rst_n)&&(FIFO.count == 1))

                                            5793     All False Count
Branch totals: 2 hits of 2 branches = 100.00%
```

b) Toggle coverage:



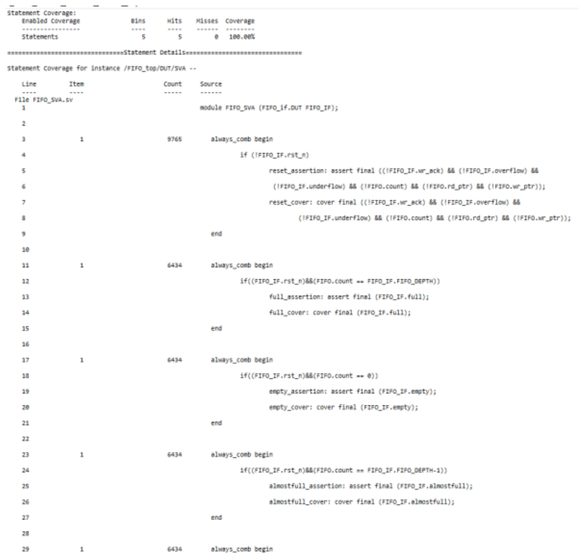
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

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

Toggle Coverage for instance /FIFO_top/FIFO_IF --

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

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

Toggle Coverage      =     100.00% (86 of 86 bins)
```

c) Condition coverage:



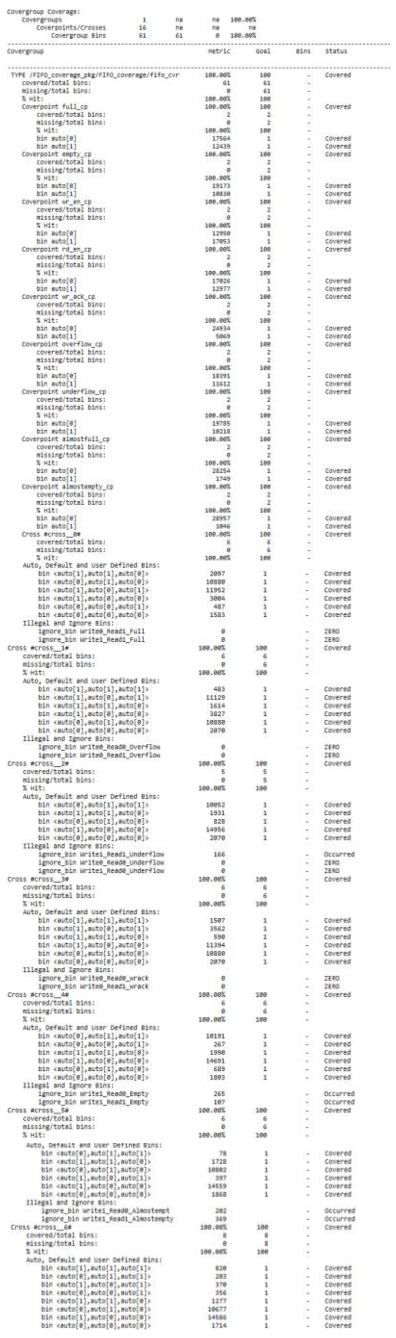
```
Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----    -------    ------  --------
    Conditions                       8          8         0   100.00%

================================Condition Details==============================

Condition Coverage for instance /FIFO_top/DUT/SVA --

  File FIFO_SVA.sv
----------------Focused Condition View-------------------
Line       12 Item    1  (FIFO_IF.rst_n && (FIFO.count == 8))
Condition totals: 2 of 2 input terms covered = 100.00%

         Input Term   Covered  Reason for no coverage   Hint
        -----------  --------  -----------------------  --------------
      FIFO_IF.rst_n         Y
  (FIFO.count == 8)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  FIFO_IF.rst_n_0       -
  Row   2:          1  FIFO_IF.rst_n_1       (FIFO.count == 8)
  Row   3:          1  (FIFO.count == 8)_0   FIFO_IF.rst_n
  Row   4:          1  (FIFO.count == 8)_1   FIFO_IF.rst_n

----------------Focused Condition View-------------------
Line       18 Item    1  (FIFO_IF.rst_n && (FIFO.count == 0))
Condition totals: 2 of 2 input terms covered = 100.00%

         Input Term   Covered  Reason for no coverage   Hint
        -----------  --------  -----------------------  --------------
      FIFO_IF.rst_n         Y
  (FIFO.count == 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  FIFO_IF.rst_n_0       -
  Row   2:          1  FIFO_IF.rst_n_1       (FIFO.count == 0)
  Row   3:          1  (FIFO.count == 0)_0   FIFO_IF.rst_n
  Row   4:          1  (FIFO.count == 0)_1   FIFO_IF.rst_n

----------------Focused Condition View-------------------
Line       24 Item    1  (FIFO_IF.rst_n && (FIFO.count == (8 - 1)))
Condition totals: 2 of 2 input terms covered = 100.00%

               Input Term   Covered  Reason for no coverage   Hint
              -----------  --------  -----------------------  --------------
            FIFO_IF.rst_n         Y
  (FIFO.count == (8 - 1))         Y

     Rows:       Hits  FEC Target                  Non-masking condition(s)
 ---------  ---------  --------------------        -------------------------
  Row   1:          1  FIFO_IF.rst_n_0             -
  Row   2:          1  FIFO_IF.rst_n_1             (FIFO.count == (8 - 1))
  Row   3:          1  (FIFO.count == (8 - 1))_0   FIFO_IF.rst_n
  Row   4:          1  (FIFO.count == (8 - 1))_1   FIFO_IF.rst_n

----------------Focused Condition View-------------------
Line       30 Item    1  (FIFO_IF.rst_n && (FIFO.count == 1))
Condition totals: 2 of 2 input terms covered = 100.00%

         Input Term   Covered  Reason for no coverage   Hint
        -----------  --------  -----------------------  --------------
      FIFO_IF.rst_n         Y
  (FIFO.count == 1)         Y

    Rows:       Hits  FEC Target             Non-masking condition(s)
---------  ---------  --------------------   -------------------------
 Row   1:          1  FIFO_IF.rst_n_0        -
 Row   2:          1  FIFO_IF.rst_n_1        (FIFO.count == 1)
 Row   3:          1  (FIFO.count == 1)_0    FIFO_IF.rst_n
 Row   4:          1  (FIFO.count == 1)_1    FIFO_IF.rst_n
```

d) Statement coverage:

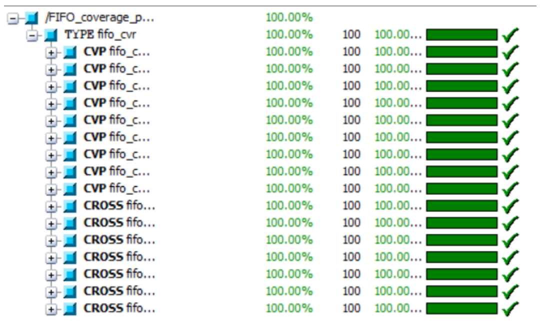


```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       5         5         0   100.00%

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

Statement Coverage for instance /FIFO_top/DUT/SVA --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO_SVA.sv
    1                                                module FIFO_SVA (FIFO_if.DUT FIFO_IF);

    2

    3              1                        9765        always_comb begin

    4                                                          if (!FIFO_IF.rst_n)

    5                                                                  reset_assertion: assert final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&

    6                                                                   (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));

    7                                                                  reset_cover: cover final ((!FIFO_IF.wr_ack) && (!FIFO_IF.overflow) &&

    8                                                                          (!FIFO_IF.underflow) && (!FIFO.count) && (!FIFO.rd_ptr) && (!FIFO.wr_ptr));

    9                                                   end

    10

    11             1                        6434        always_comb begin

    12                                                         if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH))

    13                                                                 full_assertion: assert final (FIFO_IF.full);

    14                                                                 full_cover: cover final (FIFO_IF.full);

    15                                                  end

    16

    17             1                        6434        always_comb begin

    18                                                         if((FIFO_IF.rst_n)&&(FIFO.count == 0))

    19                                                                 empty_assertion: assert final (FIFO_IF.empty);

    20                                                                 empty_cover: cover final (FIFO_IF.empty);

    21                                                  end

    22

    23             1                        6434        always_comb begin

    24                                                         if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH-1))

    25                                                                 almostfull_assertion: assert final (FIFO_IF.almostfull);

    26                                                                 almostfull_cover: cover final (FIFO_IF.almostfull);

    27                                                  end

    28

    29             1                        6434        always_comb begin
```

Functional coverage:

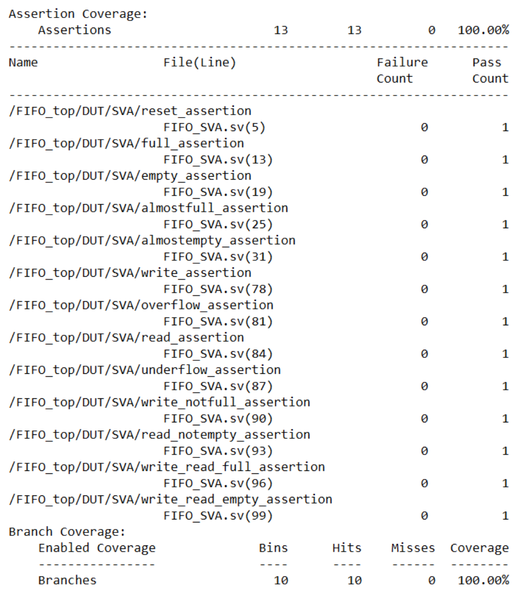


```
Covergroup Coverage:
    Covergroups                      1        na        na   100.00%
        Coverpoints/Crosses         16        na        na        na
            Covergroup Bins         61        61         0   100.00%
----------------------------------------------------------------------------------------------
Covergroup                                       Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pkg/FIFO_coverage/fifo_cvr  100.00%       100          -    Covered
    covered/total bins:                               61        61          -
    missing/total bins:                                0        61          -
    % Hit:                                       100.00%       100          -
    Coverpoint full_cp                           100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                17564         1          -    Covered
        bin auto[1]                                12439         1          -    Covered
    Coverpoint empty_cp                          100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                19173         1          -    Covered
        bin auto[1]                                10830         1          -    Covered
    Coverpoint wr_en_cp                          100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                12950         1          -    Covered
        bin auto[1]                                17053         1          -    Covered
    Coverpoint rd_en_cp                          100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                17026         1          -    Covered
        bin auto[1]                                12977         1          -    Covered
    Coverpoint wr_ack_cp                         100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                24934         1          -    Covered
        bin auto[1]                                 5069         1          -    Covered
    Coverpoint overflow_cp                       100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                18391         1          -    Covered
        bin auto[1]                                11612         1          -    Covered
    Coverpoint underflow_cp                      100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                19785         1          -    Covered
        bin auto[1]                                10218         1          -    Covered
    Coverpoint almostfull_cp                     100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                28254         1          -    Covered
        bin auto[1]                                 1749         1          -    Covered
    Coverpoint almostempty_cp                    100.00%       100          -    Covered
        covered/total bins:                            2         2          -
        missing/total bins:                            0         2          -
        % Hit:                                   100.00%       100          -
        bin auto[0]                                28957         1          -    Covered
        bin auto[1]                                 1046         1          -    Covered
    Cross #cross__0#                             100.00%       100          -    Covered
        covered/total bins:                            6         6          -
        missing/total bins:                            0         6          -
        % Hit:                                   100.00%       100          -
     Auto, Default and User Defined Bins:
         bin <auto[1],auto[1],auto[0]>             2097         1          -    Covered
         bin <auto[0],auto[1],auto[0]>            10880         1          -    Covered
         bin <auto[1],auto[0],auto[1]>            11952         1          -    Covered
         bin <auto[1],auto[0],auto[0]>             3004         1          -    Covered
         bin <auto[0],auto[0],auto[1]>              487         1          -    Covered
         bin <auto[0],auto[0],auto[0]>             1583         1          -    Covered
     Illegal and Ignore Bins:
         ignore_bin Write0_Read1_Full                 0                    -    ZERO
         ignore_bin Write1_Read1_Full                 0                    -    ZERO
Cross #cross__1#                                 100.00%       100          -    Covered
     covered/total bins:                              6         6          -
     missing/total bins:                              0         6          -
     % Hit:                                      100.00%       100          -
     Auto, Default and User Defined Bins:
         bin <auto[1],auto[1],auto[1]>              483         1          -    Covered
         bin <auto[1],auto[0],auto[1]>            11129         1          -    Covered
         bin <auto[1],auto[1],auto[0]>             1614         1          -    Covered
         bin <auto[1],auto[0],auto[0]>             3827         1          -    Covered
         bin <auto[0],auto[1],auto[0]>            10880         1          -    Covered
         bin <auto[0],auto[0],auto[0]>             2070         1          -    Covered
     Illegal and Ignore Bins:
         ignore_bin Write0_Read0_Overflow             0                    -    ZERO
         ignore_bin Write0_Read1_Overflow             0                    -    ZERO
Cross #cross__2#                                 100.00%       100          -    Covered
     covered/total bins:                              5         5          -
     missing/total bins:                              0         5          -
     % Hit:                                      100.00%       100          -
     Auto, Default and User Defined Bins:
         bin <auto[0],auto[1],auto[1]>            10052         1          -    Covered
         bin <auto[1],auto[1],auto[0]>             1931         1          -    Covered
         bin <auto[0],auto[1],auto[0]>              828         1          -    Covered
         bin <auto[1],auto[0],auto[0]>            14956         1          -    Covered
         bin <auto[0],auto[0],auto[0]>             2070         1          -    Covered
     Illegal and Ignore Bins:
         ignore_bin Write1_Read1_Underflow          166                    -    Occurred
         ignore_bin Write0_Read0_Underflow            0                    -    ZERO
         ignore_bin Write1_Read0_Underflow            0                    -    ZERO
Cross #cross__3#                                 100.00%       100          -    Covered
     covered/total bins:                              6         6          -
     missing/total bins:                              0         6          -
     % Hit:                                      100.00%       100          -
     Auto, Default and User Defined Bins:
         bin <auto[1],auto[1],auto[1]>             1507         1          -    Covered
         bin <auto[1],auto[0],auto[1]>             3562         1          -    Covered
         bin <auto[1],auto[1],auto[0]>              590         1          -    Covered
         bin <auto[1],auto[0],auto[0]>            11394         1          -    Covered
         bin <auto[0],auto[1],auto[0]>            10880         1          -    Covered
         bin <auto[0],auto[0],auto[0]>             2070         1          -    Covered
     Illegal and Ignore Bins:
         ignore_bin Write0_Read0_wrack                0                    -    ZERO
         ignore_bin Write0_Read1_wrack                0                    -    ZERO
Cross #cross__4#                                 100.00%       100          -    Covered
     covered/total bins:                              6         6          -
     missing/total bins:                              0         6          -
     % Hit:                                      100.00%       100          -
     Auto, Default and User Defined Bins:
         bin <auto[0],auto[1],auto[1]>            10191         1          -    Covered
         bin <auto[0],auto[0],auto[1]>              267         1          -    Covered
         bin <auto[1],auto[1],auto[0]>             1990         1          -    Covered
         bin <auto[1],auto[0],auto[0]>            14691         1          -    Covered
         bin <auto[0],auto[1],auto[0]>              689         1          -    Covered
         bin <auto[0],auto[0],auto[0]>             1803         1          -    Covered
     Illegal and Ignore Bins:
         ignore_bin Write1_Read0_Empty              265                    -    Occurred
         ignore_bin Write1_Read1_Empty              107                    -    Occurred
Cross #cross__5#                                 100.00%       100          -    Covered
     covered/total bins:                              6         6          -
     missing/total bins:                              0         6          -
     % Hit:                                      100.00%       100          -
      Auto, Default and User Defined Bins:
          bin <auto[0],auto[1],auto[1]>              78         1          -    Covered
          bin <auto[1],auto[1],auto[0]>            1728         1          -    Covered
          bin <auto[0],auto[1],auto[0]>           10802         1          -    Covered
          bin <auto[1],auto[0],auto[1]>             397         1          -    Covered
          bin <auto[1],auto[0],auto[0]>           14559         1          -    Covered
          bin <auto[0],auto[0],auto[0]>            1868         1          -    Covered
      Illegal and Ignore Bins:
          ignore_bin Write1_Read0_Almostempt        202                    -    Occurred
          ignore_bin Write1_Read1_Almostempty       369                    -    Occurred
 Cross #cross__6#                                100.00%       100          -    Covered
      covered/total bins:                             8         8          -
      missing/total bins:                             0         8          -
      % Hit:                                     100.00%       100          -
      Auto, Default and User Defined Bins:
          bin <auto[1],auto[1],auto[1]>             820         1          -    Covered
          bin <auto[0],auto[1],auto[1]>             203         1          -    Covered
          bin <auto[1],auto[0],auto[1]>             370         1          -    Covered
          bin <auto[0],auto[0],auto[1]>             356         1          -    Covered
          bin <auto[1],auto[1],auto[0]>            1277         1          -    Covered
          bin <auto[0],auto[1],auto[0]>           10677         1          -    Covered
          bin <auto[1],auto[0],auto[0]>           14586         1          -    Covered
          bin <auto[0],auto[0],auto[0]>            1714         1          -    Covered
```

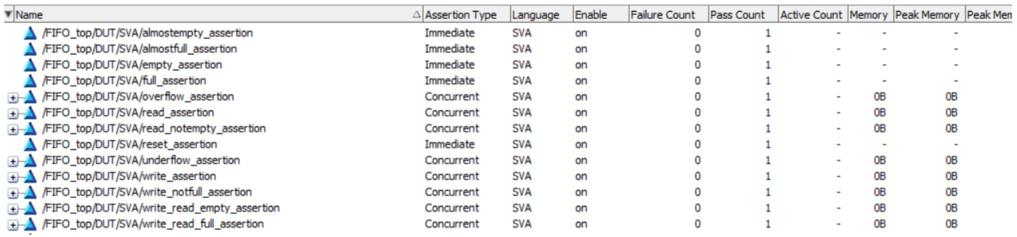


```
/FIFO_coverage_p...      100.00%
  TYPE fifo_cvr          100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CVP fifo_c...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
    CROSS fifo...        100.00%   100   100.00...  ✓
```

Assertions coverage:

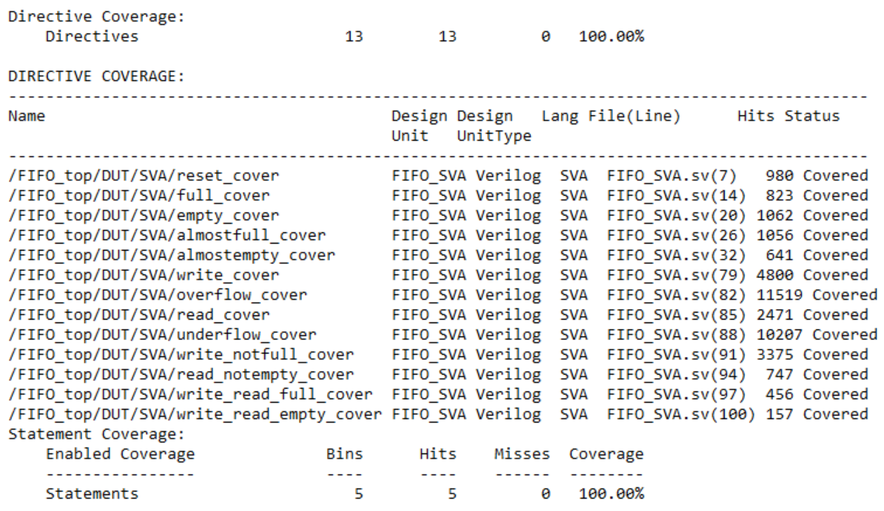


```
Assertion Coverage:
    Assertions                    13        13         0   100.00%
-------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
-------------------------------------------------------------------
/FIFO_top/DUT/SVA/reset_assertion
                     FIFO_SVA.sv(5)                    0          1
/FIFO_top/DUT/SVA/full_assertion
                     FIFO_SVA.sv(13)                   0          1
/FIFO_top/DUT/SVA/empty_assertion
                     FIFO_SVA.sv(19)                   0          1
/FIFO_top/DUT/SVA/almostfull_assertion
                     FIFO_SVA.sv(25)                   0          1
/FIFO_top/DUT/SVA/almostempty_assertion
                     FIFO_SVA.sv(31)                   0          1
/FIFO_top/DUT/SVA/write_assertion
                     FIFO_SVA.sv(78)                   0          1
/FIFO_top/DUT/SVA/overflow_assertion
                     FIFO_SVA.sv(81)                   0          1
/FIFO_top/DUT/SVA/read_assertion
                     FIFO_SVA.sv(84)                   0          1
/FIFO_top/DUT/SVA/underflow_assertion
                     FIFO_SVA.sv(87)                   0          1
/FIFO_top/DUT/SVA/write_notfull_assertion
                     FIFO_SVA.sv(90)                   0          1
/FIFO_top/DUT/SVA/read_notempty_assertion
                     FIFO_SVA.sv(93)                   0          1
/FIFO_top/DUT/SVA/write_read_full_assertion
                     FIFO_SVA.sv(96)                   0          1
/FIFO_top/DUT/SVA/write_read_empty_assertion
                     FIFO_SVA.sv(99)                   0          1
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%
```

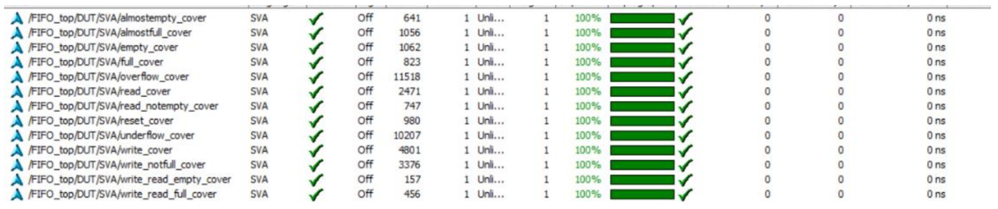


Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Men
/FIFO_top/DUT/SVA/almostempty_assertion	Immediate	SVA	on	0	1	-	-	-	
/FIFO_top/DUT/SVA/almostfull_assertion	Immediate	SVA	on	0	1	-	-	-	
/FIFO_top/DUT/SVA/empty_assertion	Immediate	SVA	on	0	1	-	-	-	
/FIFO_top/DUT/SVA/full_assertion	Immediate	SVA	on	0	1	-	-	-	
/FIFO_top/DUT/SVA/overflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/read_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/read_notempty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/reset_assertion	Immediate	SVA	on	0	1	-	-	-	
/FIFO_top/DUT/SVA/underflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/write_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/write_notfull_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/write_read_empty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	
/FIFO_top/DUT/SVA/write_read_full_assertion	Concurrent	SVA	on	0	1	-	0B	0B	

Directive coverage:



```
Directive Coverage:
    Directives                    13        13         0   100.00%

DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------
Name                                    Design Design    Lang File(Line)      Hits Status
                                        Unit   UnitType
-------------------------------------------------------------------------------------------
/FIFO_top/DUT/SVA/reset_cover           FIFO_SVA Verilog  SVA  FIFO_SVA.sv(7)    980 Covered
/FIFO_top/DUT/SVA/full_cover            FIFO_SVA Verilog  SVA  FIFO_SVA.sv(14)  823 Covered
/FIFO_top/DUT/SVA/empty_cover           FIFO_SVA Verilog  SVA  FIFO_SVA.sv(20) 1062 Covered
/FIFO_top/DUT/SVA/almostfull_cover      FIFO_SVA Verilog  SVA  FIFO_SVA.sv(26) 1056 Covered
/FIFO_top/DUT/SVA/almostempty_cover     FIFO_SVA Verilog  SVA  FIFO_SVA.sv(32)  641 Covered
/FIFO_top/DUT/SVA/write_cover           FIFO_SVA Verilog  SVA  FIFO_SVA.sv(79) 4800 Covered
/FIFO_top/DUT/SVA/overflow_cover        FIFO_SVA Verilog  SVA  FIFO_SVA.sv(82) 11519 Covered
/FIFO_top/DUT/SVA/read_cover            FIFO_SVA Verilog  SVA  FIFO_SVA.sv(85) 2471 Covered
/FIFO_top/DUT/SVA/underflow_cover       FIFO_SVA Verilog  SVA  FIFO_SVA.sv(88) 10207 Covered
/FIFO_top/DUT/SVA/write_notfull_cover   FIFO_SVA Verilog  SVA  FIFO_SVA.sv(91) 3375 Covered
/FIFO_top/DUT/SVA/read_notempty_cover   FIFO_SVA Verilog  SVA  FIFO_SVA.sv(94)  747 Covered
/FIFO_top/DUT/SVA/write_read_full_cover FIFO_SVA Verilog  SVA  FIFO_SVA.sv(97)  456 Covered
/FIFO_top/DUT/SVA/write_read_empty_cover FIFO_SVA Verilog SVA  FIFO_SVA.sv(100) 157 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       5         5         0   100.00%
```



/FIFO_top/DUT/SVA/almostempty_cover	SVA	✓	Off	641	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/almostfull_cover	SVA	✓	Off	1056	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/empty_cover	SVA	✓	Off	1062	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/full_cover	SVA	✓	Off	823	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/overflow_cover	SVA	✓	Off	11518	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/read_cover	SVA	✓	Off	2471	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/read_notempty_cover	SVA	✓	Off	747	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/reset_cover	SVA	✓	Off	980	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/underflow_cover	SVA	✓	Off	10207	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/write_cover	SVA	✓	Off	4801	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/write_notfull_cover	SVA	✓	Off	3376	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/write_read_empty_cover	SVA	✓	Off	157	1	Unli...	1	100%	🟩	✓	0	0	0 ns
/FIFO_top/DUT/SVA/write_read_full_cover	SVA	✓	Off	456	1	Unli...	1	100%	🟩	✓	0	0	0 ns