FIFO

UVM Project

By: Khaled Ahmed Hamed

Under supervision of Eng. Kareem Waseem

For more details please visit: Github repo

Bug report:

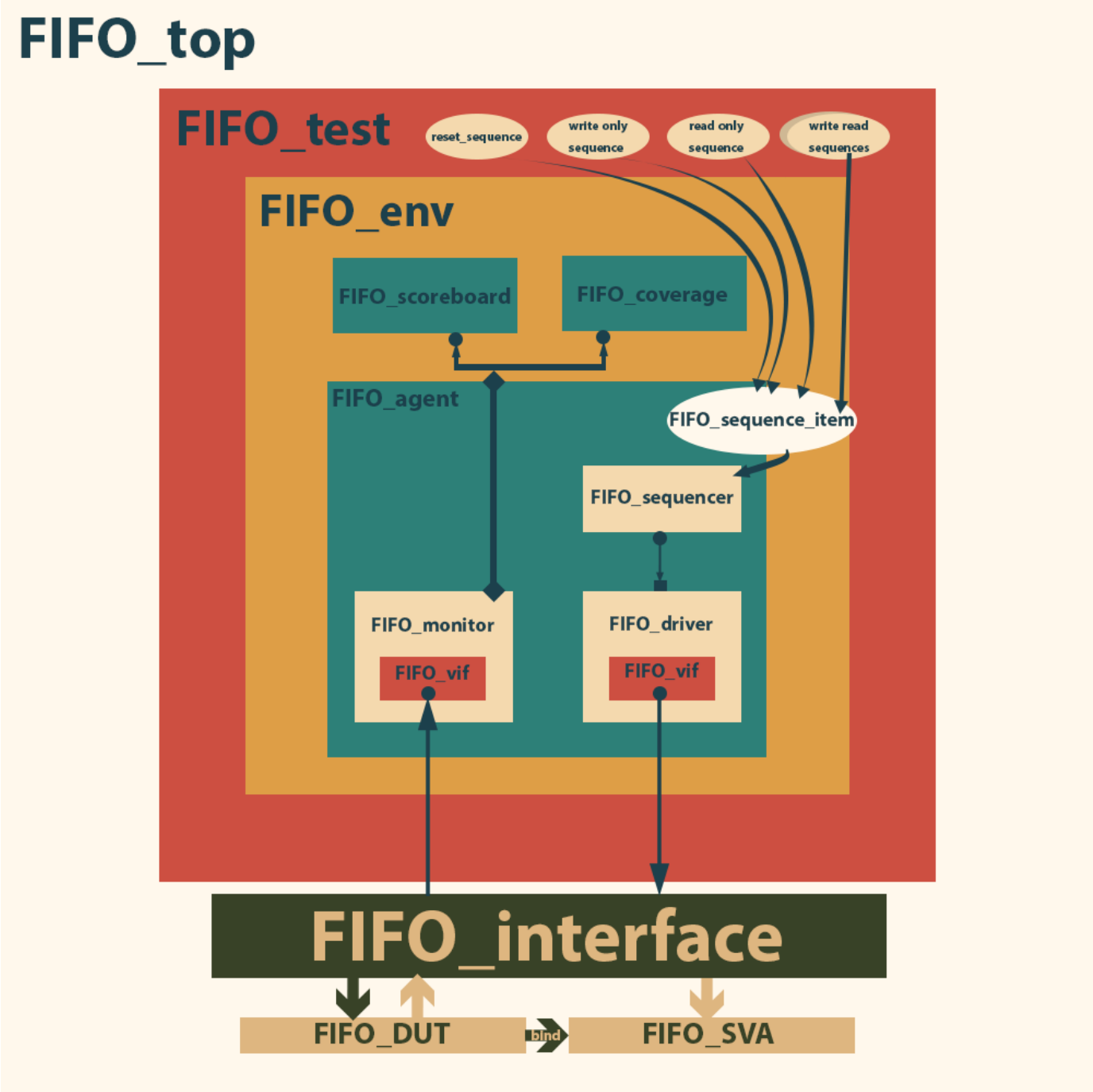
Design has 6 bugs:

1. Reset signals overflow, wr_ack and underflow (data_out not to be included)
2. Unhandled 2 cases:
    - If a read and write enables were high and the FIFO was empty, only writing will take place.
    - If a read and write enables were high and the FIFO was full, only reading will take place.
3. underflow is sequential not combinational.
4. almostfull flag: FIFO_DEPTH-2 corrected to FIFO_DEPTH-1.
5. When successful write operation has occured, overflow can't be high.
6. When successful read operation has occured, underflow can't be high.

Verification plan:

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
LABEL1	When the rst_n is asserted, All flags & internal signals should equal 0	Randomized less often 95 off & 5% on during the simulation	.	Immediate assertion to check async rst_n functionality as it is async
LABEL2	When rst_n is deactivated & the FIFO is full of elements (conut=depth), The full flag should be high	Randomized, Write enable to be high with distribution of the value WR_EN_ON_DIST and to be low with 100-WR_EN_ON_DISTduring the simulation	Cross coverage: WRITE_READ_FULL_CROSS, covers write & read enables and full flag	Immediate assertion to check full flag functionality as it is combinational
LABEL3	When rst_n is deactivated & the FIFO has one element left(conut=depth-1), The almostfull flag should be high	Randomized, Write enable to be high with distribution of the value WR_EN_ON_DIST and to be low with 100-WR_EN_ON_DISTduring the simulation	Cross coverage: WRITE_READ_ALMOST_FULL_CROSS, covers write & read enables and almostfull flag	Immediate assertion to check almostfull flag functionality as it is combinational
LABEL4	When rst_n is deactivated & the FIFO has no element inside (conut=0), The empty flag should be high	Randomized, Read enable to be high with distribution of the value RD_EN_ON_DIST and to be low with 100-RD_EN_ON_DIST during the simulation	Cross coverage: WRITE_READ_EMPTY_CROSS, covers write & read enables and empty flag	Immediate assertion to check empty flag functionality as it is combinational
LABEL5	When rst_n is deactivated & the FIFO has only one element inside (conut=1), The empty flag should be high	Randomized, Read enable to be high with distribution of the value RD_EN_ON_DIST and to be low with 100-RD_EN_ON_DIST during the simulation	Cross coverage: WRITE_READ_ALMSOT_EMPTY_CROSS, covers write & read enables and almostempty flag	Immediate assertion to check almostempty flag functionality as it is combinational
LABEL6	When rst_n is deactivated, You want to read & the FIFO is not full, The wr_ack flag should be high and wr_ptr should increment	Randomized, Write enable to be high with distribution of the value WR_EN_ON_DIST and to be low with 100-WR_EN_ON_DISTduring the simulation	Cross coverage: WRITE_READ_WR_ACK_CROSS, covers write & read enables and wr_ack flag	Concurrent assertion to check wr_ptr & wr_ack flag functionality as they are sequential
LABEL7	When rst_n is deactivated, You want to read & the FIFO is full, The overflow flag should be high	Randomized, Write enable to be high with distribution of the value WR_EN_ON_DIST and to be low with 100-WR_EN_ON_DISTduring the simulation	Cross coverage: WRITE_READ_OVERFLOW_CROSS, covers write & read enables and overflow flag	Concurrent assertion to check overflow flag functionality as it is sequential
LABEL8	When rst_n is deactivated, You want to read & the FIFO is not empty rd_ptr should increment and data_out should equal mem[rd_ptr]	Randomized, Read enable to be high with distribution of the value RD_EN_ON_DIST and to be low with 100-RD_EN_ON_DIST during the simulation	.	Concurrent assertion to check rd_ptr functionality as it is sequential, data_out checked against refernce model to check functionality
LABEL9	When rst_n is deactivated, You want to read & the FIFO is empty underflow flag should be high	Randomized, Read enable to be high with distribution of the value RD_EN_ON_DIST and to be low with 100-RD_EN_ON_DIST during the	Cross coverage: WRITE_READ_UNDERFLOW_CROSS, covers write & read enables and overflow flag	Concurrent assertion to check underflow flag functionality as it is sequential

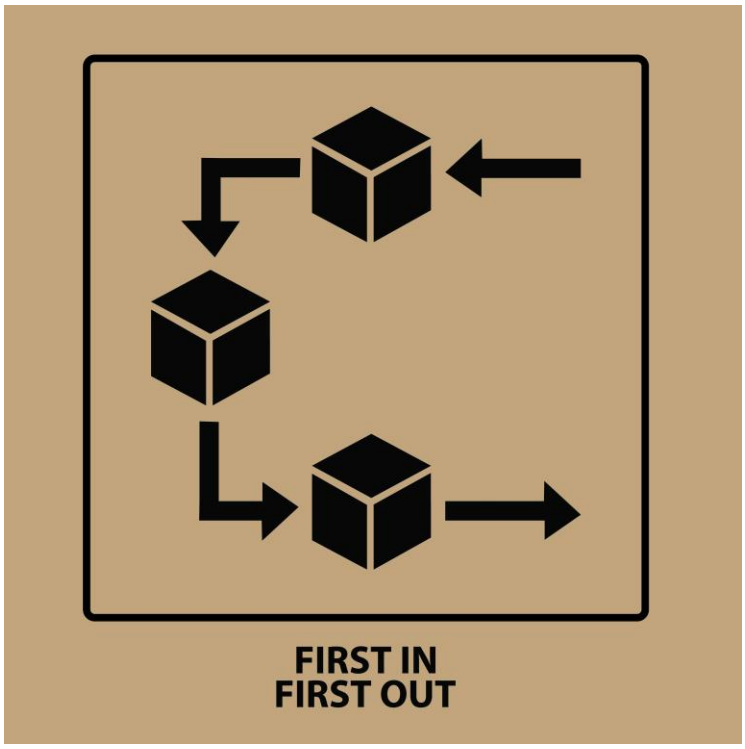
UVM testbench structure



Testbench flow

- FIFO_top module:
  1. instantiates the DUT, FIFO_interface & bind assertions (FIFO_SVA).
  2. Generate the clock.
  3. Passes interface (virtual FIFO_interface) using configuration database (Shared database between components).
  4. Runs test.
- FIFO_test:
  1. Build the FIFO_env & Sequences.
  2. Retrieve the virtual interface from the configuration database by configuration object (Object holds configuration settings and parameters for UVM components).
  3. Sets the configuration object into the configuration database.
  4. Builds the environment (FIFO_env)
  5. Starts sequences on the sequencer.
- Sequences:
  1. There are 5 sequences: Reset, Write only, Read only, Write Read & write read empty sequences.
  2. Core stimulus of any verification plan.
  3. Written within task body.
- FIFO_sequence_item:
  1. Data fields to communicate with DUT (Input & Output signals).
  2. Randomizes signals.
  3. Constraints blocks are added here to ensure verification plan.
- FIFO_env:
  Builds and connects scoreboard (FIFO_scoreboard), Coverage collector (FIFO_coverage), agent (FIFO_agent) and analysis components (Ports & Exports).
- FIFO_sequencer:
  Generates transactions as class objects and sends it to the driver (FIFO_driver) for execution.

- FIFO_driver:
  1. Pulls the next item from the sequencer.
  2. Drives the sequence item in the run_phase task using the virtual interface.
- FIFO_monitor:
  Captures signals information from DUT, translates it into sequence items and finally sends it analysis components (Ports & Exports).
- FIFO_scoreboard:
  1. Receives sequence items from the monitor.
  2. Runs input signals to the reference model (Task or Module but I have used a task) to compare the DUT output with the expected output to check the functionality of the FIFO.
- FIFO_coverage:
  1. Receives sequence items from the monitor.
  2. Contains the covergroups to ensure the verification plan.
  3. Samples the data fields for functional coverage.



Design code snippets:

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

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

always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    if (!FIFO_IF.rst_n) begin
        wr_ptr <= 0;
        // Bug detected: Reset signals FIFO_IF.overflow & FIFO_IF.wr_ack
        FIFO_IF.overflow <= 0;
        FIFO_IF.wr_ack <= 0;
    end
    else if (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH) begin
        mem[wr_ptr] <= FIFO_IF.data_in;
        FIFO_IF.wr_ack<=1;
        wr_ptr <= wr_ptr + 1;
        FIFO_IF.overflow <= 0;// Successful write operation, overflow can't be high
    end
    else begin
        FIFO_IF.wr_ack <= 0;
        if (FIFO_IF.full & FIFO_IF.wr_en)
            FIFO_IF.overflow <= 1;
        else
            FIFO_IF.overflow <= 0;
    end
end

always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    if (!FIFO_IF.rst_n) begin
        rd_ptr <= 0;
        // Bug detected: Reset signals FIFO_IF.underflow
        FIFO_IF.underflow <= 0;
    end
    else if (FIFO_IF.rd_en && count != 0) begin
        FIFO_IF.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        FIFO_IF.underflow <= 0;// Successful read operation, underflow can't be high
    end
    // Handled FIFO_IF.underflow behaviour when turned from combinational to sequential
    else begin
        if (FIFO_IF.empty & FIFO_IF.rd_en)
            FIFO_IF.underflow <= 1;
        else
            FIFO_IF.underflow <= 0;
    end
end
```

```
always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    if (!FIFO_IF.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full)
            count <= count + 1;
        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty)
            count <= count - 1;
        // Bug detected: Unhandled case,  If a read and write enables were high and the FIFO was FIFO_IF.empty, only writing will take place.
        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty)
            count <= count + 1;
        // Bug detected: Unhandled cases,  If a read and write enables were high and the FIFO was FIFO_IF.full, only reading will take place.
        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full)
            count <= count - 1;
    end
end

assign FIFO_IF.full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
assign FIFO_IF.empty = (count == 0)? 1 : 0;
assign FIFO_IF.almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0; // Bug detected: FIFO_IF.FIFO_DEPTH-2 --> FIFO_IF.FIFO_DEPTH-1
assign FIFO_IF.almostempty = (count == 1)? 1 : 0;
endmodule
```

Assertions 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)`
	`assert final ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr )&&(!FIFO.rd_ptr)&&(!FIFO.count));`
	`cover final ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!FIFO.count));`
	`end`
	
	`@(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n)`
	`(!FIFO_IF.rst_n) |-> ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!FIFO.count));`

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 (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 (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 (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 (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 P2;`
	`@(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'b1)||(FIFO.wr_ptr==0 && $past(FIFO.wr_ptr) +1'b1 == 8)));`
	`endproperty`

Whenever the rst_n is deactivated, Write enable is high & FIFO is full, overflow should be high.	`property P3;`
	`@(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`(FIFO_IF.full & FIFO_IF.wr_en) |=> (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 P4;`
	`  @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`  (FIFO_IF.rd_en && FIFO.count != 0) |=> ((FIFO.rd_ptr==$past(FIFO.rd_ptr)+1'b1)||(FIFO.rd_ptr==0 && $past(FIFO.rd_ptr) +1'b1 == 8));`
	`endproperty`

Whenever the rst_n is deactivated, Read enable is high & FIFO is empty, underflow should be high.	`property P5;`
	`  @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)`
	`  (FIFO_IF.empty & FIFO_IF.rd_en) |=> (FIFO_IF.underflow);`
	`endproperty`

Whenever the rst_n is deactivated, Write enable is high & FIFO is not full, Write 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'b10) && !FIFO_IF.full)  |=> ((FIFO.count==$past(FIFO.count)+1'b1));`
	`endproperty`

Whenever the rst_n is deactivated, read enable is high & FIFO is not empty, Read 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'b01) && !FIFO_IF.empty)  |=> ((FIFO.count==$past(FIFO.count)-1'b1));`
	`endproperty`

Whenever the rst_n is deactivated, Both of read & write enables are high & FIFO is empty, Write 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'b1));`
	`endproperty`

Whenever the rst_n is deactivated, Both of read & write enables are high & FIFO is not full, Read operation should take place.	`property P9;`
	`  @(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'b1));`
	`endproperty`

SVA code snippets:

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

    // Properties, Assertions & Covers
    always_comb begin
        if(!FIFO_IF.rst_n)
        reset_1_assertion: assert final ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr )&&(!FIFO.rd_ptr)&&(!FIFO.count));
        reset_1_cover: cover final ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!FIFO.count));
    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 (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 (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 (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 (FIFO_IF.almostempty);
    end

    property P1;
        @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n)
        (!FIFO_IF.rst_n) |-> ((!FIFO_IF.wr_ack)&&(!FIFO_IF.overflow)&&(!FIFO_IF.underflow)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!FIFO.count));
    endproperty

    property P2;
        @(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'b1)||(FIFO.wr_ptr==0 && $past(FIFO.wr_ptr) +1'b1 == 8)));
    endproperty

    property P3;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.full & FIFO_IF.wr_en) |=> (FIFO_IF.overflow);
    endproperty


    property P4;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.rd_en && FIFO.count != 0) |=> ((FIFO.rd_ptr==$past(FIFO.rd_ptr)+1'b1)||(FIFO.rd_ptr==0 && $past(FIFO.rd_ptr) +1'b1 == 8));
    endproperty

    property P5;
        @(posedge FIFO_IF.clk) disable iff(!FIFO_IF.rst_n)
        (FIFO_IF.empty & FIFO_IF.rd_en) |=> (FIFO_IF.underflow);
    endproperty

    property P6;
        @(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'b1));
    endproperty

    property P7;
        @(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'b1));
    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'b1));
    endproperty

    property P9;
        @(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'b1));
    endproperty

    reset_2_assertion: assert property(P1);
    reset_2_cover: cover property(P1);

    write_assertion: assert property(P2);
    write_cover: cover property(P2);

    overflow_assertion: assert property(P3);
    overflow_cover: cover property(P3);

    read_assertion: assert property(P4);
    read_cover: cover property(P4);

    underflow_assertion: assert property(P5);
    underflow_cover: cover property(P5);

    write_not_full_assertion: assert property(P6);
    write_not_full_cover: cover property(P6);

    read_not_empty_assertion: assert property(P7);
    read_not_empty_cover: cover property(P7);

    read_write_empty_assertion: assert property(P8);
    read_write_empty_cover: cover property(P8);

    read_write_full_assertion: assert property(P9);
    read_write_full_cover: cover property(P9);
endmodule : FIFO_SVA
```

Interface code snippet:

```
interface FIFO_interface (input bit clk);

    // Parameters
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    localparam max_fifo_addr = $clog2(FIFO_DEPTH);

    // Signals (All IO ports)
    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 Design & Assertion
    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 module code snippet:

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

module FIFO_top ();
    bit clk;

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

    // Instentiate the interface
    FIFO_if FIFO_IF(clk);

    // Instentiate the DUT module & pass the interface
    FIFO DUT(FIFO_IF);

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

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

endmodule
```

Shared package code snippet:

```
package FIFO_shared_pkg;

    // Counters to be used in scoreboard
    integer correct_count = 0;
    integer error_count = 0;

endpackage
```

Environment code snippet:

```
package FIFO_env_pkg;
    import uvm_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 : FIFO_env_pkg
```

Agent code snippet:

```
package FIFO_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh";
    import FIFO_shared_pkg::*;
    import FIFO_driver_pkg::*;
    import FIFO_sequencer_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_sequencer 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_sequencer::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_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::*;
    `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
```

Sequence Item code snippets:

```
package FIFO_sequence_item_pkg;
    import uvm_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;
        localparam max_fifo_addr = $clog2(FIFO_DEPTH);

        // Randomized 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 signals as integers**************/
        integer RD_EN_ON_DIST = 30;
        integer WR_EN_ON_DIST = 70;

        // Constructor
        function new(string name = "FIFO_seq_item");
            super.new(name);
        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 : convert2string

        function string convert2string_stimulus();
            return $sformatf(" rst_n = 0b%0b, data_in = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b",
            rst_n, data_in, wr_en, rd_en, );
        endfunction : convert2string_stimulus

    endclass : FIFO_seq_item
endpackage : FIFO_sequence_item_pkg
```

Sequencer code snippet:

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

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

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

endpackage : FIFO_sequencer_pkg
```

Reset Sequence code snippet:

```
package FIFO_reset_seq_pkg;
    import FIFO_sequence_item_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
```

Write only sequence code snippet:

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

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

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

        task body();
            repeat (10_000) 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_only_seq

endpackage : FIFO_write_only_seq_pkg
```

Read only sequence code snippet:

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

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

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

        task body();
            repeat (10_000) 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_only_seq

endpackage : FIFO_read_only_seq_pkg
```

Write & Read sequence code snippet:

```
package FIFO_write_read_seq_pkg;
    import FIFO_sequence_item_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 (10_000) 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 with empty sequence code snippet:

```
package FIFO_write_read_empty_seq_pkg;
    import FIFO_sequence_item_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;
                seq_item.rd_en = 0;
                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;
                seq_item.rd_en = 1;
            end
            seq_item.rst_n = 1;
            seq_item.wr_en = 1;
            seq_item.rd_en = 1;
            finish_item(seq_item);
        endtask : body
    endclass : FIFO_write_read_empty_seq

endpackage : FIFO_write_read_empty_seq_pkg
```

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
```

Scoreboard code snippets:

```
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 operations %0d", correct_count), UVM_MEDIUM);
            `uvm_info("report_phase", $sformatf("total errors operations %0d", error_count), UVM_MEDIUM);
        endfunction
    endclass : FIFO_scoreboard

endpackage : FIFO_scoreboard_pkg
```

Coverage code snippets:

```
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 cg;
            // Coverpoints
            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;
            full_cp: coverpoint seq_item_cov.full;
            almostfull_cp: coverpoint seq_item_cov.almostfull;
            empty_cp: coverpoint seq_item_cov.empty;
            almostempty_cp: coverpoint seq_item_cov.almostempty;

            // Crosses
            WRITE_READ_WR_ACK_CROSS: cross wr_en_cp, rd_en_cp, wr_ack_cp{
                ignore_bins WRITE0_READ1_WR_ACK1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{1} && binsof(wr_ack_cp)intersect{1};
                ignore_bins WRITE0_READ0_WR_ACK1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(wr_ack_cp)intersect{1};
            }
            WRITE_READ_OVERFLOW_CROSS: cross wr_en_cp, rd_en_cp, overflow_cp{
                ignore_bins WRITE0_READ1_OVERFLOW1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{1} && binsof(overflow_cp)intersect{1};
                ignore_bins WRITE0_READ0_OVERFLOW1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(overflow_cp)intersect{1};
            }
            WRITE_READ_UNDERFLOW_CROSS: cross wr_en_cp, rd_en_cp, underflow_cp {
                ignore_bins WRITE1_READ0_UNDERFLOW1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{0} && binsof(underflow_cp)intersect{1};
                ignore_bins WRITE0_READ0_UNDERFLOW1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(underflow_cp)intersect{1};
                ignore_bins WRITE1_READ1_UNDERFLOW1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{1} && binsof(underflow_cp)intersect{1};
            }
            WRITE_READ_FULL_CROSS: cross wr_en_cp, rd_en_cp, full_cp {
                ignore_bins WRITE1_READ1_FULL1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{1} && binsof(full_cp)intersect{1};
                ignore_bins WRITE0_READ1_FULL1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{1} && binsof(full_cp)intersect{1};
            }
            WRITE_READ_EMPTY_CROSS: cross wr_en_cp, rd_en_cp, empty_cp {
                ignore_bins WRITE1_READ1_EMPTY1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{1} && binsof(empty_cp)intersect{1};
                ignore_bins WRITE0_READ1_EMPTY1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{0} && binsof(empty_cp)intersect{1};
            }
            WRITE_READ_ALMOST_FULL_CROSS: cross wr_en_cp, rd_en_cp, almostfull_cp;
            WRITE_READ_ALMOST_EMPTY_CROSS: cross wr_en_cp, rd_en_cp, almostempty_cp {
                ignore_bins WRITE0_READ0_ALMOSTEMPTY1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(almostempty_cp)intersect{1};
                ignore_bins WRITE1_READ1_ALMOSTEMPTY1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{1} && binsof(almostempty_cp)intersect{1};
            }
        endgroup : cg
```

```
        function new(string name = "FIFO_coverage", uvm_component parent = null);
            super.new(name, parent);
            cg = 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);
                cg.sample();
            end
        endtask : run_phase
    endclass : FIFO_coverage
endpackage : FIFO_coverage_pkg
```

Test code snippets:

```
package FIFO_test_pkg;
    import uvm_pkg::*;
    import FIFO_env_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_read_only_seq_pkg::*;
    import FIFO_write_read_seq_pkg::*;
    import FIFO_write_only_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;
        virtual FIFO_if FIFO_test_vif;
        FIFO_config FIFO_config_obj_test;
        FIFO_read_only_seq read_seq;
        FIFO_write_read_seq write_read_seq;
        FIFO_write_only_seq write_seq;
        FIFO_reset_seq reset_seq;
        FIFO_write_read_empty_seq write_read_empty_seq;

        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_only_seq::type_id::create("read_seq");
            reset_seq = FIFO_reset_seq::type_id::create("reset_seq");
            write_seq = FIFO_write_only_seq::type_id::create("write_seq");
            write_read_seq = FIFO_write_read_seq::type_id::create("write_read_seq");
            write_read_empty_seq = FIFO_write_read_empty_seq::type_id::create("write_read_empty_seq");

            if (!uvm_config_db#(virtual FIFO_if)::get(this, "", "FIFO_IF", FIFO_config_obj_test.FIFO_config_vif)) begin
                `uvm_fatal("build_phase","Test - unable to get the virtual interface");
            end
            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);

            `uvm_info("run_phase", "Reset sequence asserted", UVM_LOW)
            reset_seq.start(FIFO_env_comp.agt.sqr);

            `uvm_info("run_phase","Write operation started", UVM_LOW)
            write_seq.start(FIFO_env_comp.agt.sqr);

            `uvm_info("run_phase","Read operation started", UVM_LOW)
            read_seq.start(FIFO_env_comp.agt.sqr);

            `uvm_info("run_phase","Write & Read operations together with empty started", UVM_LOW)
            write_read_empty_seq.start(FIFO_env_comp.agt.sqr);

            `uvm_info("run_phase","Write & Read operations together started", UVM_LOW)
            write_read_seq.start(FIFO_env_comp.agt.sqr);

            phase.drop_objection(this);
        endtask : run_phase

    endclass : FIFO_test

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
quit -sim
vcover report FIFO.ucdb -details -annotate -all -output coverage_FIFO_rpt.txt
```

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_sequencer_pkg.sv
FIFO_reset_seq_pkg.sv
FIFO_write_only_seq_pkg.sv
FIFO_read_only_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
```

Transcript snippet:

```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test FIFO_test...
# UVM_INFO FIFO_test_pkg.sv(48) @ 0: uvm_test_top [run_phase] Reset sequence asserted
# UVM_INFO FIFO_test_pkg.sv(51) @ 2: uvm_test_top [run_phase] Write operation started
# UVM_INFO FIFO_test_pkg.sv(54) @ 20002: uvm_test_top [run_phase] Read operation started
# UVM_INFO FIFO_test_pkg.sv(57) @ 40002: uvm_test_top [run_phase] Write & Read operations together started
# UVM_INFO FIFO_test_pkg.sv(60) @ 40004: uvm_test_top [run_phase] Write & Read operations together with empty started
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 60004: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_scoreboard_pkg.sv(74) @ 60004: uvm_test_top.FIFO_env_comp.sb [report_phase] total successful operations 30001
# UVM_INFO FIFO_scoreboard_pkg.sv(75) @ 60004: uvm_test_top.FIFO_env_comp.sb [report_phase] total errors operations 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   11
# 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]     5
# ** Note: $finish    : E:/DIGITAL/Programs/QuestaSim/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 60004 ns  Iteration: 61  Instance: /FIFO_top
```

Code coverage:

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------------------------------------
    5                                       9764     Count coming in to IF
    5              1                         959               if(!FIFO_IF.rst_n)
                                            8805     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    11                                      6432     Count coming in to IF
    11             1                         823               if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH))
                                            5609     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    17                                      6432     Count coming in to IF
    17             1                         577               if((FIFO_IF.rst_n)&&(FIFO.count == 0))
                                            5855     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    23                                      6432     Count coming in to IF
    23             1                        1056               if((FIFO_IF.rst_n)&&(FIFO.count == FIFO_IF.FIFO_DEPTH-1))
                                            5376     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    29                                      6432     Count coming in to IF
    29             1                         641               if((FIFO_IF.rst_n)&&(FIFO.count == 1))
                                            5791     All False Count
Branch totals: 2 hits of 2 branches = 100.00%
```

```
Branches - by instance (/FIFO_top/DUT)                                    Branch
FIFO.sv
    8 if (!FIFO_IF.rst_n) begin
   14 else if (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH) begin
   20 else begin
   22 if (FIFO_IF.full & FIFO_IF.wr_en)
   24 else
   30 if (!FIFO_IF.rst_n) begin
   35 else if (FIFO_IF.rd_en && count != 0) begin
   41 else begin
   42 if (FIFO_IF.empty & FIFO_IF.rd_en)
   44 else
   50 if (!FIFO_IF.rst_n) begin
   53 else begin
   54 if       ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full)
   56 else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty)
   59 else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty)
   62 else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full)
   67 assign FIFO_IF.full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
   68 assign FIFO_IF.empty = (count == 0)? 1 : 0;
   69 assign FIFO_IF.almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0; // Bug detected: FIFO_IF.FIFO_DEPTH-2 --> FIFO_IF.FIFO_DEPTH-1
   70 assign FIFO_IF.almostempty = (count == 1)? 1 : 0;
FIFO_top.sv
```

Statement coverage:

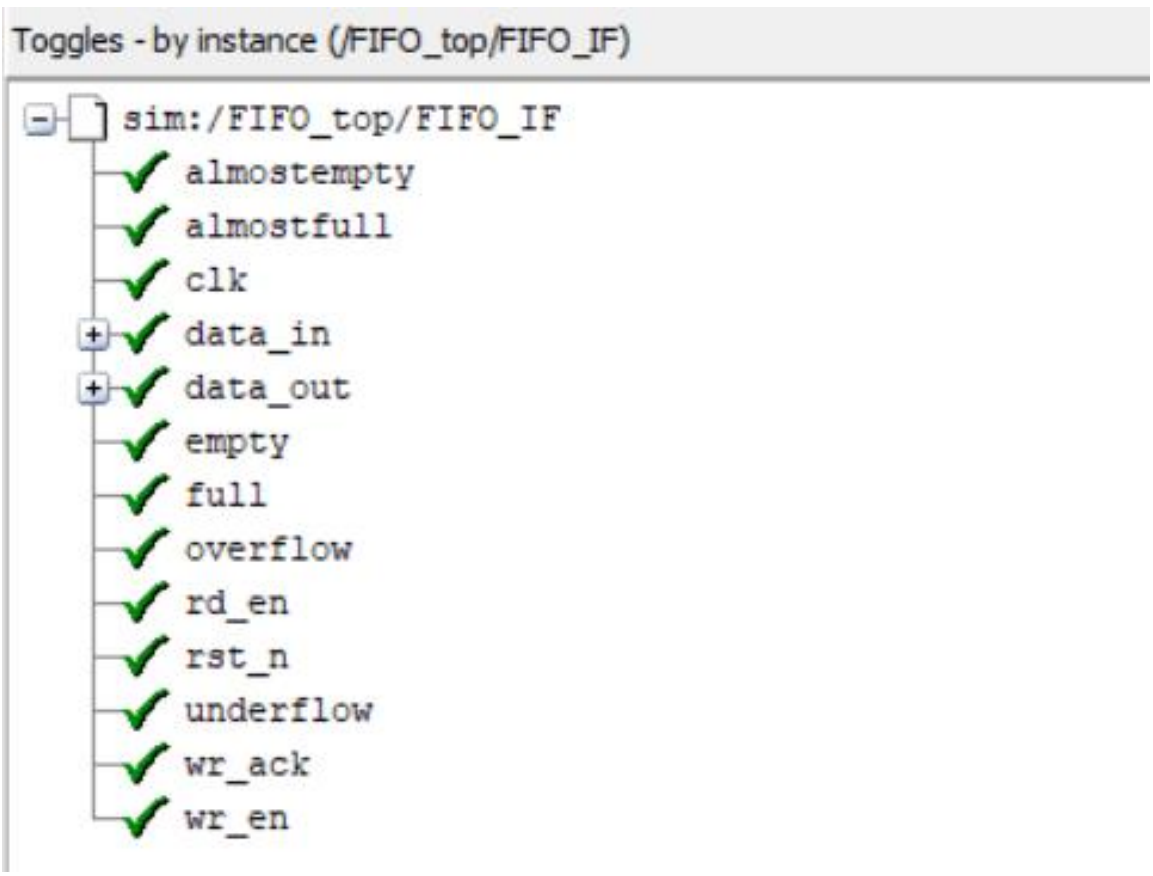
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      29        29         0   100.00%

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

Statement Coverage for instance /FIFO_top/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
    1                                                module FIFO (FIFO_if.DUT FIFO_IF);
    2
    3                                                reg [FIFO_IF.max_fifo_addr-1:0] wr_ptr, rd_ptr;
    4                                                reg [FIFO_IF.max_fifo_addr:0] count;
    5                                                reg [FIFO_IF.FIFO_WIDTH-1:0] mem [FIFO_IF.FIFO_DEPTH-1:0];
    6
    7              1                         30486   always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    8                                                  if (!FIFO_IF.rst_n) begin
    9              1                          1003           wr_ptr <= 0;
    10                                                        // Bug detected: Reset signals FIFO_IF.overflow & FIFO_IF.wr_ack
    11             1                          1003           FIFO_IF.overflow <= 0;
    12             1                          1003           FIFO_IF.wr_ack <= 0;
    13                                                 end
    14                                                 else if (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH) begin
    15             1                          5070           mem[wr_ptr] <= FIFO_IF.data_in;
    16             1                          5070           FIFO_IF.wr_ack<=1;
    17             1                          5070           wr_ptr <= wr_ptr + 1;
    18             1                          5070           FIFO_IF.overflow <= 0;// Successful write operation, overflow can't be high
    19                                                 end
    20                                                 else begin
    21             1                         24413           FIFO_IF.wr_ack <= 0;
    22                                                        if (FIFO_IF.full & FIFO_IF.wr_en)
    23             1                         11611                   FIFO_IF.overflow <= 1;
    24                                                        else
    25             1                         12802                   FIFO_IF.overflow <= 0;
    26                                                 end
    27                                               end
    28
    29             1                         30486   always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    30                                                 if (!FIFO_IF.rst_n) begin
    31             1                          1003           rd_ptr <= 0;
    32                                                        // Bug detected: Reset signals FIFO_IF.underflow
    33             1                          1003           FIFO_IF.underflow <= 0;
    34                                                 end
    35                                                 else if (FIFO_IF.rd_en && count != 0) begin
    36             1                          2607           FIFO_IF.data_out <= mem[rd_ptr];
    37             1                          2607           rd_ptr <= rd_ptr + 1;
    38             1                          2607           FIFO_IF.underflow <= 0;// Successful read operation, underflow can't be high
    39                                                 end
    40                                                 // Handled FIFO_IF.underflow behaviour when turned from combinational to sequential
    41                                                 else begin
    42                                                        if (FIFO_IF.empty & FIFO_IF.rd_en)
    43             1                         10217                   FIFO_IF.underflow <= 1;
    44                                                        else
    45             1                         16659                   FIFO_IF.underflow <= 0;
    46                                                 end
    47                                               end
    48
    49             1                          9306   always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    50                                                 if (!FIFO_IF.rst_n) begin
    51             1                           992           count <= 0;
    52                                                 end
    53                                                 else begin
    54                                                        if      ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full)
    55             1                          3563                   count <= count + 1;
    56                                                        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty)
    57             1                           782                   count <= count - 1;
    58                                                        // Bug detected: Unhandled case,  If a read and write enables were high and the FIFO was FIFO_IF.empty, only writing will take place.
    59                                                        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty)
    60             1                           165                   count <= count + 1;
    61                                                        // Bug detected: Unhandled cases,  If a read and write enables were high and the FIFO was FIFO_IF.full, only reading will take place.
    62                                                        else if ( ({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full)
    63             1                           483                   count <= count - 1;
    64                                                 end
    65                                               end
    66
    67             1                          5464   assign FIFO_IF.full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
    68             1                          5464   assign FIFO_IF.empty = (count == 0)? 1 : 0;
    69             1                          5464   assign FIFO_IF.almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0; // Bug detected: FIFO_IF.FIFO_DEPTH-2 --> FIFO_IF.FIFO_DEPTH-1
    70             1                          5464   assign FIFO_IF.almostempty = (count == 1)? 1 : 0;
```

Statements - by instance (/FIFO_top/DUT)

```
FIFO.sv
    7 always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
    9 wr_ptr <= 0;
   11 FIFO_IF.overflow <= 0;
   12 FIFO_IF.wr_ack <= 0;
   15 mem[wr_ptr] <= FIFO_IF.data_in;
   16 FIFO_IF.wr_ack<=1;
   17 wr_ptr <= wr_ptr + 1;
   18 FIFO_IF.overflow <= 0;// Successful write operation, overflow can't be high
   21 FIFO_IF.wr_ack <= 0;
   23 FIFO_IF.overflow <= 1;
   25 FIFO_IF.overflow <= 0;
   29 always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
   31 rd_ptr <= 0;
   33 FIFO_IF.underflow <= 0;
   36 FIFO_IF.data_out <= mem[rd_ptr];
   37 rd_ptr <= rd_ptr + 1;
   38 FIFO_IF.underflow <= 0;// Successful read operation, underflow can't be high
   43 FIFO_IF.underflow <= 1;
   45 FIFO_IF.underflow <= 0;
   49 always @(posedge FIFO_IF.clk or negedge FIFO_IF.rst_n) begin
   51 count <= 0;
   55 count <= count + 1;
   57 count <= count - 1;
   60 count <= count + 1;
   63 count <= count - 1;
   67 assign FIFO_IF.full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
   68 assign FIFO_IF.empty = (count == 0)? 1 : 0;
   69 assign FIFO_IF.almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0; // Bug detected: FIFO_IF.FIFO_DEPTH-2 --> FIFO_IF.FIFO_DEPTH-1
   70 assign FIFO_IF.almostempty = (count == 1)? 1 : 0;
FIFO_top.sv
```

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)
```



Functional coverage:

```
Covergroup Coverage:
    Covergroups                    1        na        na   100.00%
        Coverpoints/Crosses       18        na        na        na
            Covergroup Bins       61        61         0   100.00%
----------------------------------------------------------------------------------------------
Covergroup                                        Metric       Goal       Bins    Status
----------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pkg/FIFO_coverage/cg         100.00%        100          -    Covered
     covered/total bins:                               61         61          -
     missing/total bins:                                0         61          -
     % Hit:                                       100.00%        100          -
     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]                                12049          1          -    Covered
         bin auto[1]                                17853          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]                                17825          1          -    Covered
         bin auto[1]                                12377          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]                                24932          1          -    Covered
         bin auto[1]                                 5070          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]                                11611          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]                                19755          1          -    Covered
         bin auto[1]                                10217          1          -    Covered
     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]                                12428          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]                                28253          1          -    Covered
         bin auto[1]                                 1749          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]                                10829          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]                                28955          1          -    Covered
         bin auto[1]                                 1047          1          -    Covered
     Cross WRITE_READ_WR_ACK_CROSS                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]>           3563          1          -    Covered
             bin <auto[1],auto[1],auto[0]>            590          1          -    Covered
             bin <auto[1],auto[0],auto[0]>          11393          1          -    Covered
             bin <auto[0],auto[1],auto[0]>          10850          1          -    Covered
             bin <auto[0],auto[0],auto[0]>           2069          1          -    Covered
         Illegal and Ignore Bins:
             ignore_bin WRITE0_READ0_WR_ACK1            0                     -    ZERO
             ignore_bin WRITE0_READ1_WR_ACK1            0                     -    ZERO
     Cross WRITE_READ_OVERFLOW_CROSS              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]>         11128          1          -    Covered
             bin <auto[1],auto[1],auto[0]>          1614          1          -    Covered
             bin <auto[1],auto[0],auto[0]>          3528          1          -    Covered
             bin <auto[0],auto[1],auto[0]>         10850          1          -    Covered
```

```
             bin <auto[0],auto[1],auto[0]>         10850          1          -    Covered
             bin <auto[0],auto[0],auto[0]>          2069          1          -    Covered
         Illegal and Ignore Bins:
             ignore_bin WRITE0_READ0_OVERFLOW1          0                     -    ZERO
             ignore_bin WRITE0_READ1_OVERFLOW1          0                     -    ZERO
     Cross WRITE_READ_UNDERFLOW_CROSS             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]>           1932          1          -    Covered
             bin <auto[0],auto[1],auto[0]>            828          1          -    Covered
             bin <auto[1],auto[0],auto[0]>          14056          1          -    Covered
             bin <auto[0],auto[0],auto[0]>           2069          1          -    Covered
         Illegal and Ignore Bins:
             ignore_bin WRITE1_READ1_UNDERFLOW1       165                     -    Occurred
             ignore_bin WRITE0_READ0_UNDERFLOW1         0                     -    ZERO
             ignore_bin WRITE1_READ0_UNDERFLOW1         0                     -    ZERO
     Cross WRITE_READ_FULL_CROSS                  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]>          10850          1          -    Covered
             bin <auto[1],auto[0],auto[1]>          11951          1          -    Covered
             bin <auto[1],auto[0],auto[0]>           3005          1          -    Covered
             bin <auto[0],auto[0],auto[1]>            487          1          -    Covered
             bin <auto[0],auto[0],auto[0]>           1582          1          -    Covered
         Illegal and Ignore Bins:
             ignore_bin WRITE0_READ1_FULL1              0                     -    ZERO
             ignore_bin WRITE1_READ1_FULL1              0                     -    ZERO
     Cross WRITE_READ_EMPTY_CROSS                 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]>            266          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 WRITE0_READ1_EMPTY1           265                     -    Occurred
             ignore_bin WRITE1_READ1_EMPTY1           107                     -    Occurred
     Cross WRITE_READ_ALMOST_FULL_CROSS           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]>          1715          1          -    Covered
     Cross WRITE_READ_ALMOST_EMPTY_CROSS          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]>         10021          1          -    Covered
             bin <auto[1],auto[0],auto[1]>           398          1          -    Covered
             bin <auto[1],auto[0],auto[0]>         14558          1          -    Covered
             bin <auto[0],auto[0],auto[0]>          1867          1          -    Covered
         Illegal and Ignore Bins:
             ignore_bin WRITE1_READ1_ALMOSTEMPTY1       360                     -    Occurred
             ignore_bin WRITE0_READ0_ALMOSTEMPTY1       202                     -    Occurred
```

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_coverage_pkg/FIFO_coverage		100.00%							
TYPE cg		100.00%	100	100.00...	✓		auto(1)		
CVP cg::wr_en_cp		100.00%	100	100.00...	✓				
CVP cg::rd_en_cp		100.00%	100	100.00...	✓				
CVP cg::wr_ack_cp		100.00%	100	100.00...	✓				
CVP cg::overflow_cp		100.00%	100	100.00...	✓				
CVP cg::underflow_cp		100.00%	100	100.00...	✓				
CVP cg::full_cp		100.00%	100	100.00...	✓				
CVP cg::almostfull_cp		100.00%	100	100.00...	✓				
CVP cg::empty_cp		100.00%	100	100.00...	✓				
CVP cg::almostempty_cp		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_WR_ACK_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_OVERFLOW_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_UNDERFLOW_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_FULL_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_EMPTY_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_ALMOST_FULL_CROSS		100.00%	100	100.00...	✓				
CROSS cg::WRITE_READ_ALMOST_EMPTY_CROSS		100.00%	100	100.00...	✓				

Assertions coverage (Sequential Domain):

```
Assertion Coverage:
    Assertions                      14        14         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/FIFO_top/DUT/SVA/reset_1_assertion
                     FIFO_SVA.sv(6)                     0          1
/FIFO_top/DUT/SVA/full_assertion
                     FIFO_SVA.sv(12)                    0          1
/FIFO_top/DUT/SVA/empty_assertion
                     FIFO_SVA.sv(18)                    0          1
/FIFO_top/DUT/SVA/almostfull_assertion
                     FIFO_SVA.sv(24)                    0          1
/FIFO_top/DUT/SVA/almostempty_assertion
                     FIFO_SVA.sv(30)                    0          1
/FIFO_top/DUT/SVA/reset_2_assertion
                     FIFO_SVA.sv(80)                    0          1
/FIFO_top/DUT/SVA/write_assertion
                     FIFO_SVA.sv(83)                    0          1
/FIFO_top/DUT/SVA/overflow_assertion
                     FIFO_SVA.sv(86)                    0          1
/FIFO_top/DUT/SVA/read_assertion
                     FIFO_SVA.sv(89)                    0          1
/FIFO_top/DUT/SVA/underflow_assertion
                     FIFO_SVA.sv(92)                    0          1
/FIFO_top/DUT/SVA/write_not_full_assertion
                     FIFO_SVA.sv(95)                    0          1
/FIFO_top/DUT/SVA/read_not_empty_assertion
                     FIFO_SVA.sv(98)                    0          1
/FIFO_top/DUT/SVA/read_write_empty_assertion
                     FIFO_SVA.sv(101)                   0          1
/FIFO_top/DUT/SVA/read_write_full_assertion
                     FIFO_SVA.sv(104)                   0          1
```

Name	Assertion Type	Language	Enable	Failure Count	Pass Count
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__17...	Immediate	SVA	on	0	0
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__17...	Immediate	SVA	on	0	0
/FIFO_top/DUT/SVA/reset_1_assertion	Immediate	SVA	on	0	1
/FIFO_top/DUT/SVA/full_assertion	Immediate	SVA	on	0	1
/FIFO_top/DUT/SVA/empty_assertion	Immediate	SVA	on	0	1
/FIFO_top/DUT/SVA/almostfull_assertion	Immediate	SVA	on	0	1
/FIFO_top/DUT/SVA/almostempty_assertion	Immediate	SVA	on	0	1
/FIFO_top/DUT/SVA/reset_2_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/write_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/overflow_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/read_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/underflow_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/write_not_full_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/read_not_empty_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/read_write_empty_assertion	Concurrent	SVA	on	0	1
/FIFO_top/DUT/SVA/read_write_full_assertion	Concurrent	SVA	on	0	1

Directive coverage:

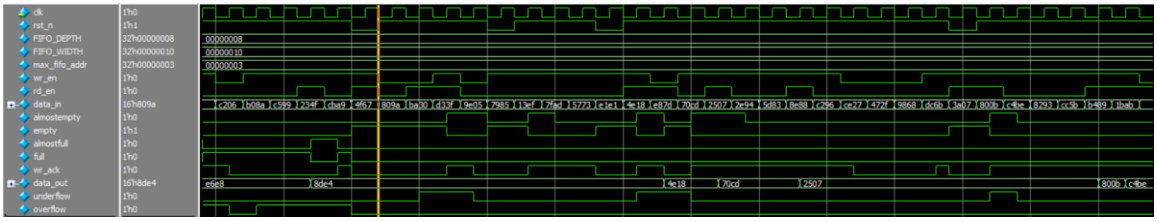
```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/FIFO_top/DUT/SVA/reset_1_cover         FIFO_SVA Verilog  SVA  FIFO_SVA.sv(7)   979 Covered
/FIFO_top/DUT/SVA/full_cover            FIFO_SVA Verilog  SVA  FIFO_SVA.sv(13)  960 Covered
/FIFO_top/DUT/SVA/empty_cover           FIFO_SVA Verilog  SVA  FIFO_SVA.sv(19) 1061 Covered
/FIFO_top/DUT/SVA/almostfull_cover      FIFO_SVA Verilog  SVA  FIFO_SVA.sv(25) 1140 Covered
/FIFO_top/DUT/SVA/almostempty_cover     FIFO_SVA Verilog  SVA  FIFO_SVA.sv(31)  705 Covered
/FIFO_top/DUT/SVA/reset_2_cover         FIFO_SVA Verilog  SVA  FIFO_SVA.sv(81)  519 Covered
/FIFO_top/DUT/SVA/write_cover           FIFO_SVA Verilog  SVA  FIFO_SVA.sv(84) 4801 Covered
/FIFO_top/DUT/SVA/overflow_cover        FIFO_SVA Verilog  SVA  FIFO_SVA.sv(87) 11518 Covered
/FIFO_top/DUT/SVA/read_cover            FIFO_SVA Verilog  SVA  FIFO_SVA.sv(90) 2471 Covered
/FIFO_top/DUT/SVA/underflow_cover       FIFO_SVA Verilog  SVA  FIFO_SVA.sv(93) 10207 Covered
/FIFO_top/DUT/SVA/write_not_full_cover  FIFO_SVA Verilog  SVA  FIFO_SVA.sv(96) 3376 Covered
/FIFO_top/DUT/SVA/read_not_empty_cover  FIFO_SVA Verilog  SVA  FIFO_SVA.sv(99)  747 Covered
/FIFO_top/DUT/SVA/read_write_empty_cover FIFO_SVA Verilog  SVA  FIFO_SVA.sv(102) 157 Covered
/FIFO_top/DUT/SVA/read_write_full_cover  FIFO_SVA Verilog  SVA  FIFO_SVA.sv(105) 456 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 14
```

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/FIFO_top/DUT/SVA/reset_1_cover	SVA	✓	Off	979	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/full_cover	SVA	✓	Off	960	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/empty_cover	SVA	✓	Off	1061	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/almostfull_cover	SVA	✓	Off	1140	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/almostempty_cover	SVA	✓	Off	705	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/reset_2_cover	SVA	✓	Off	519	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/write_cover	SVA	✓	Off	4801	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/overflow_cover	SVA	✓	Off	11518	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/read_cover	SVA	✓	Off	2471	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/underflow_cover	SVA	✓	Off	10207	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/write_not_full_cover	SVA	✓	Off	3376	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/read_not_empty_cover	SVA	✓	Off	747	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/read_write_empty_cover	SVA	✓	Off	157	1	Unlimited	1	100%		✓
/FIFO_top/DUT/SVA/read_write_full_cover	SVA	✓	Off	456	1	Unlimited	1	100%		✓

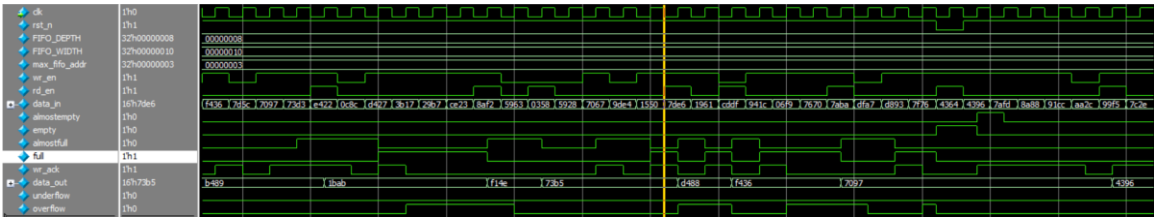
QuestaSim simulation waveform & transcript snippets:

Labels snippets:

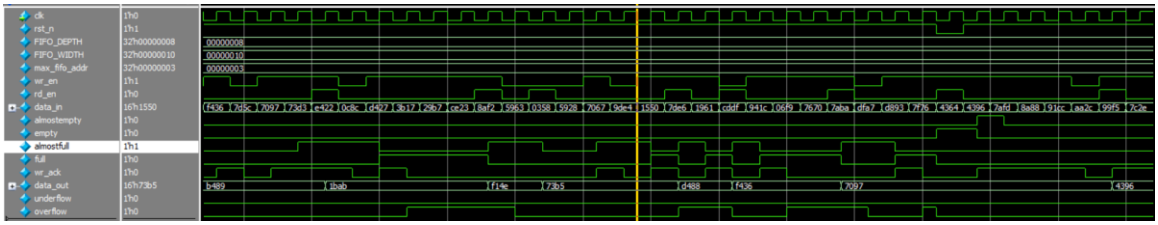
Label1(When the rst_n is asserted, All flags & internal signals should equal 0):



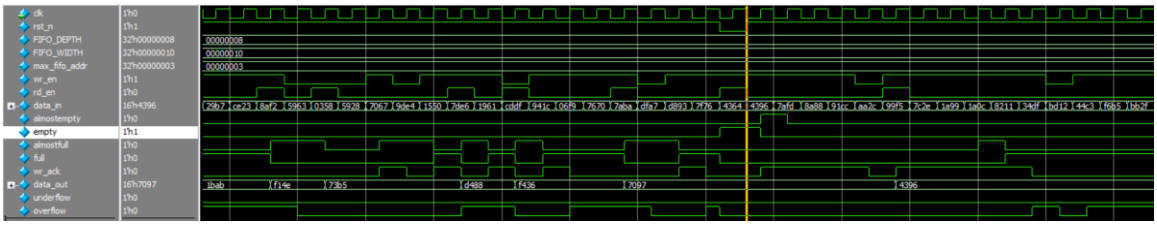
Label2(When rst_n is deactivated & the FIFO is full of elements (conut=depth), The full flag should be high):



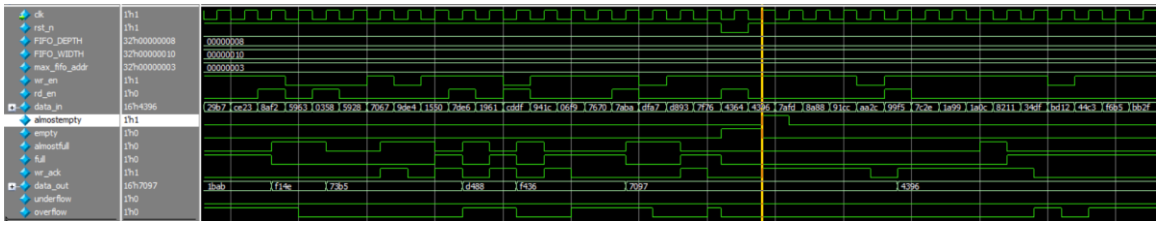
Label3(When rst_n is deactivated & the FIFO has one element left free(conut=depth-1), The almostfull flag should be high):



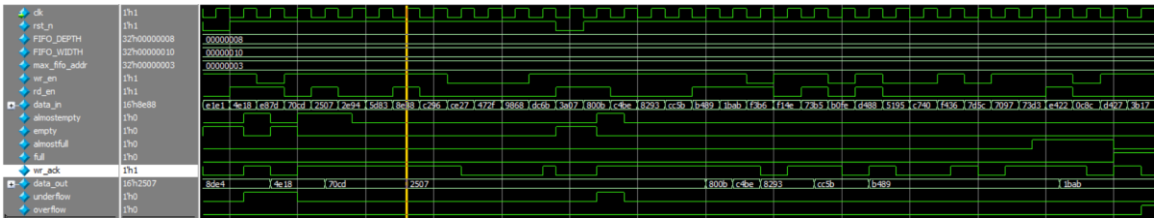
Label4(When rst_n is deactivated & the FIFO has no element inside (conut=0), The empty flag should be high):



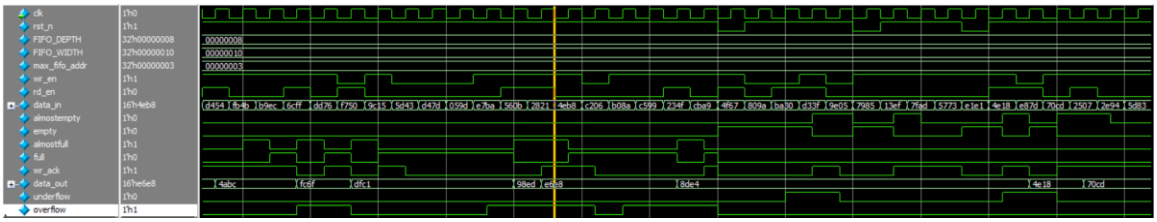
Label5(When rst_n is deactivated & the FIFO has only one element inside (conut=1), The empty flag should be high):



Label6(When rst_n is deactivated, You want to read & the FIFO is not full, The wr_ack flag should be high and wr_ptr should increment):



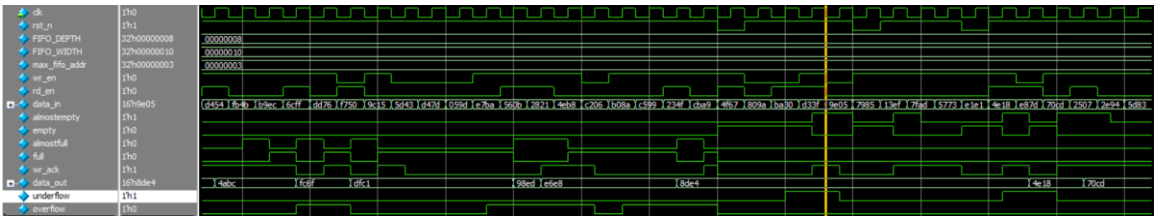
Label7(When rst_n is deactivated, You want to read & the FIFO is full, The overflow flag should be high):



Label8(When rst_n is deactivated, You want to read & the FIFO is not empty rd_ptr should increment and data_out should equal mem[rd_ptr]):

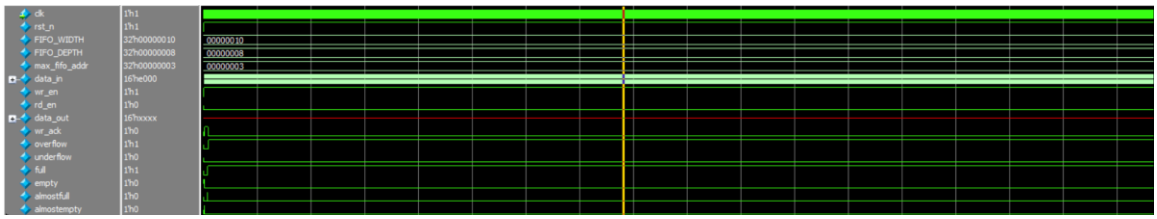
```
# UVM_INFO FIFO_scoreboard_pkg.sv(47) @ 59778: uvm_test_top.FIFO_env_comp.sb [run_phase] Correct Transacton received, Output is:  rst_n = 0b1, data_in = 0b1110011101101000, wr_en = 0b0, rd_en = 0b0,
#
#                    data_out = 0b1010000100001000, wr_ack = 0b0, overflow = 0b0, underflow = 0b0, full = 0b0,
#
#                    empty = 0b0, almostfull = 0b0, almostempty = 0b0
# UVM_INFO FIFO_scoreboard_pkg.sv(47) @ 59780: uvm_test_top.FIFO_env_comp.sb [run_phase] Correct Transacton received, Output is:  rst_n = 0b1, data_in = 0b1010101101111, wr_en = 0b1, rd_en = 0b0,
#
#                    data_out = 0b1010000100001000, wr_ack = 0b1, overflow = 0b0, underflow = 0b0, full = 0b0,
#
#                    empty = 0b0, almostfull = 0b1, almostempty = 0b0
# UVM_INFO FIFO_scoreboard_pkg.sv(47) @ 59782: uvm_test_top.FIFO_env_comp.sb [run_phase] Correct Transacton received, Output is:  rst_n = 0b1, data_in = 0b1001110110010011, wr_en = 0b1, rd_en = 0b0,
#
#                    data_out = 0b1010000100001000, wr_ack = 0b1, overflow = 0b0, underflow = 0b0, full = 0b1,
#
#                    empty = 0b0, almostfull = 0b0, almostempty = 0b0
```

Label9(When rst_n is deactivated, You want to read & the FIFO is empty underflow flag should be high):

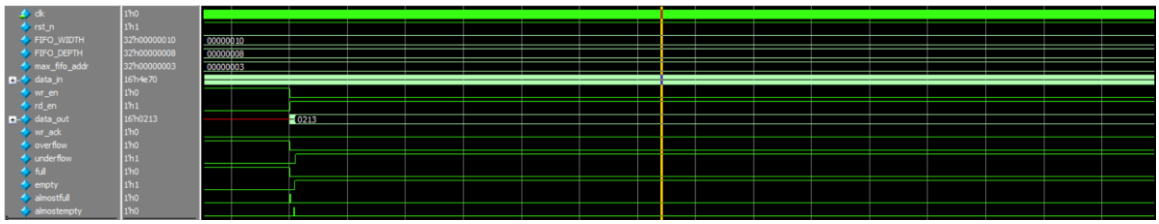


Sequences snippets:

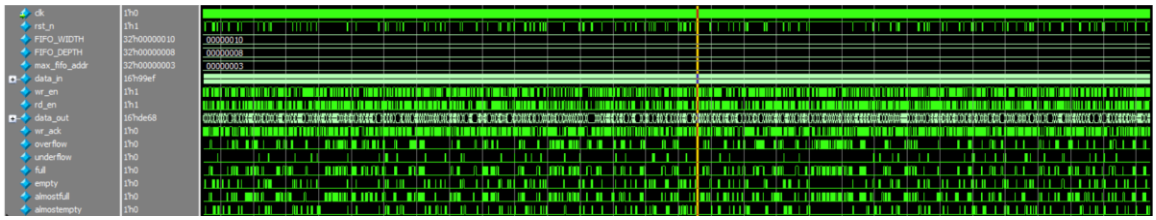
Reset & Write only sequences:



Read only sequence:



Write & Read together sequences:



تم بحمد الله

سُورَةُ النَّجۡمِ

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

وَأَن لَّيۡسَ لِلۡإِنسَٰنِ إِلَّا مَا سَعَىٰ ٣٩

سُورَةُ التَّوۡبَةِ

وَقُلِ ٱعۡمَلُواْ فَسَيَرَى ٱللَّهُ عَمَلَكُمۡ وَرَسُولُهُۥ وَٱلۡمُؤۡمِنُونَۖ
وَسَتُرَدُّونَ إِلَىٰ عَٰلِمِ ٱلۡغَيۡبِ وَٱلشَّهَٰدَةِ فَيُنَبِّئُكُم بِمَا كُنتُمۡ
تَعۡمَلُونَ ١٠٥