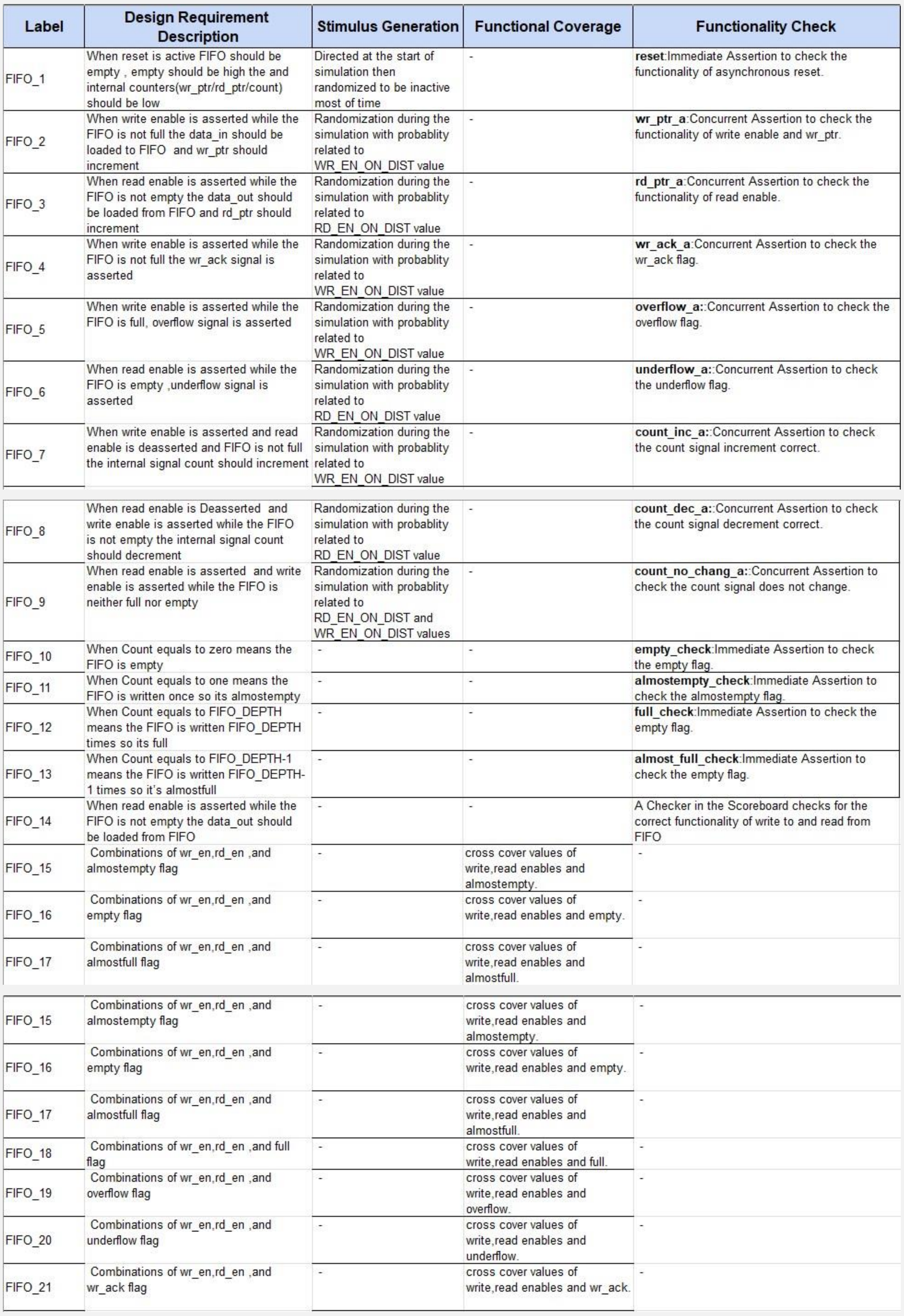
UVM-based Verification

Synchronous FIFO

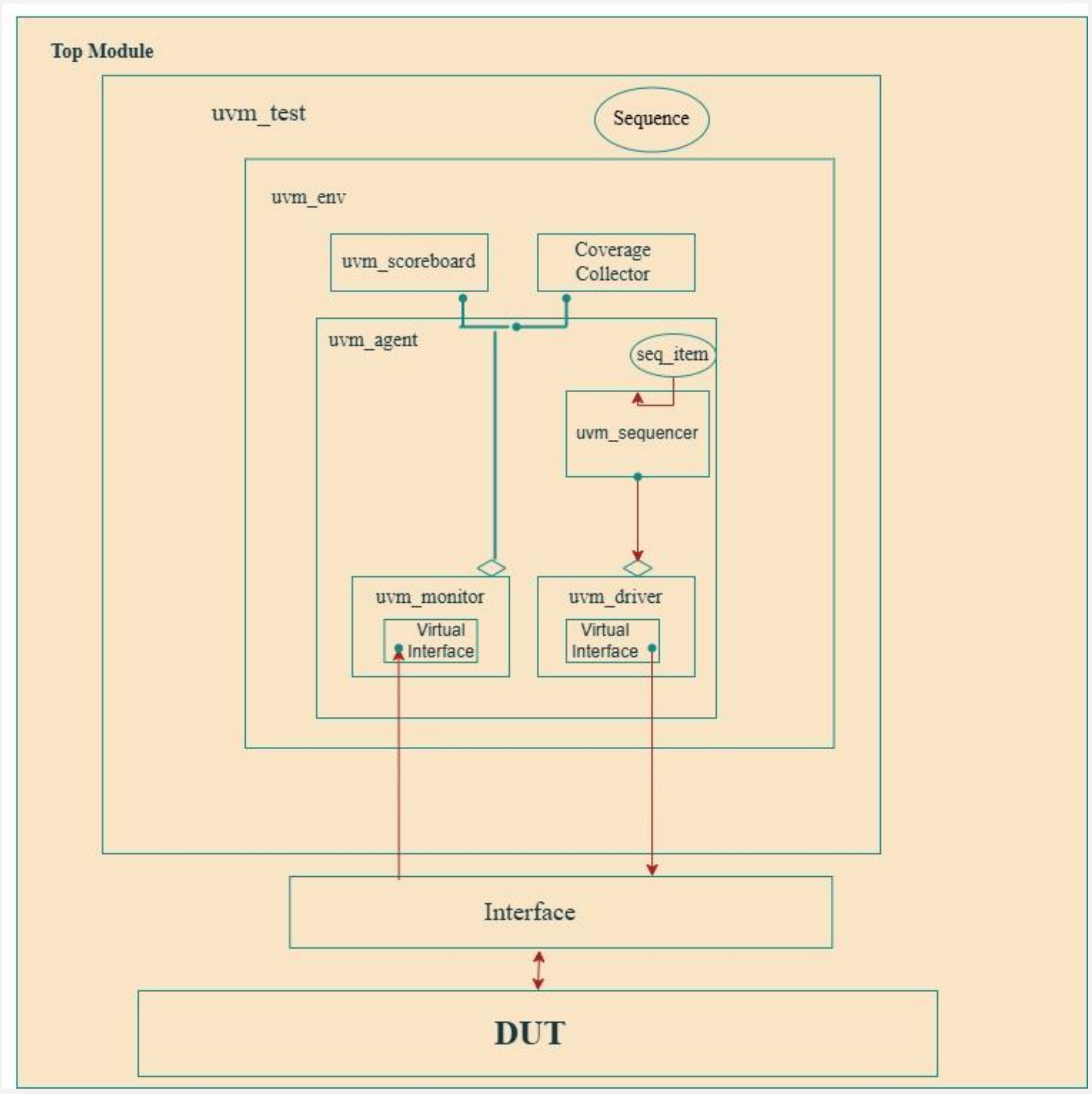
By Moustafa Mohammed

1- Verification Plan:



Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When reset is active FIFO should be empty , empty should be high the and internal counters(wr_ptr/rd_ptr/count) should be low	Directed at the start of simulation then randomized to be inactive most of time	-	**reset**:Immediate Assertion to check the functionality of asynchronous reset.
FIFO_2	When write enable is asserted while the FIFO is not full the data_in should be loaded to FIFO and wr_ptr should increment	Randomization during the simulation with probablity related to WR_EN_ON_DIST value	-	**wr_ptr_a**:Concurrent Assertion to check the functionality of write enable and wr_ptr.
FIFO_3	When read enable is asserted while the FIFO is not empty the data_out should be loaded from FIFO and rd_ptr should increment	Randomization during the simulation with probablity related to RD_EN_ON_DIST value	-	**rd_ptr_a**:Concurrent Assertion to check the functionality of read enable.
FIFO_4	When write enable is asserted while the FIFO is not full the wr_ack signal is asserted	Randomization during the simulation with probablity related to WR_EN_ON_DIST value	-	**wr_ack_a**:Concurrent Assertion to check the wr_ack flag.
FIFO_5	When write enable is asserted while the FIFO is full, overflow signal is asserted	Randomization during the simulation with probablity related to WR_EN_ON_DIST value	-	**overflow_a:**:Concurrent Assertion to check the overflow flag.
FIFO_6	When read enable is asserted while the FIFO is empty ,underflow signal is asserted	Randomization during the simulation with probablity related to RD_EN_ON_DIST value	-	**underflow_a:**:Concurrent Assertion to check the underflow flag.
FIFO_7	When write enable is asserted and read enable is deasserted and FIFO is not full the internal signal count should increment	Randomization during the simulation with probablity related to WR_EN_ON_DIST value	-	**count_inc_a:**:Concurrent Assertion to check the count signal increment correct.
FIFO_8	When read enable is Deasserted and write enable is asserted while the FIFO is not empty the internal signal count should decrement	Randomization during the simulation with probablity related to RD_EN_ON_DIST value	-	**count_dec_a:**:Concurrent Assertion to check the count signal decrement correct.
FIFO_9	When read enable is asserted and write enable is asserted while the FIFO is neither full nor empty	Randomization during the simulation with probablity related to RD_EN_ON_DIST and WR_EN_ON_DIST values	-	**count_no_chang_a:**:Concurrent Assertion to check the count signal does not change.
FIFO_10	When Count equals to zero means the FIFO is empty	-	-	**empty_check**:Immediate Assertion to check the empty flag.
FIFO_11	When Count equals to one means the FIFO is written once so its almostempty	-	-	**almostempty_check**:Immediate Assertion to check the almostempty flag.
FIFO_12	When Count equals to FIFO_DEPTH means the FIFO is written FIFO_DEPTH times so its full	-	-	**full_check**:Immediate Assertion to check the empty flag.
FIFO_13	When Count equals to FIFO_DEPTH-1 means the FIFO is written FIFO_DEPTH-1 times so it's almostfull	-	-	**almost_full_check**:Immediate Assertion to check the empty flag.
FIFO_14	When read enable is asserted while the FIFO is not empty the data_out should be loaded from FIFO	-	-	A Checker in the Scoreboard checks for the correct functionality of write to and read from FIFO
FIFO_15	Combinations of wr_en,rd_en ,and almostempty flag	-	cross cover values of write,read enables and almostempty.	-
FIFO_16	Combinations of wr_en,rd_en ,and empty flag	-	cross cover values of write,read enables and empty.	-
FIFO_17	Combinations of wr_en,rd_en ,and almostfull flag	-	cross cover values of write,read enables and almostfull.	-
FIFO_15	Combinations of wr_en,rd_en ,and almostempty flag	-	cross cover values of write,read enables and almostempty.	-
FIFO_16	Combinations of wr_en,rd_en ,and empty flag	-	cross cover values of write,read enables and empty.	-
FIFO_17	Combinations of wr_en,rd_en ,and almostfull flag	-	cross cover values of write,read enables and almostfull.	-
FIFO_18	Combinations of wr_en,rd_en ,and full flag	-	cross cover values of write,read enables and full.	-
FIFO_19	Combinations of wr_en,rd_en ,and overflow flag	-	cross cover values of write,read enables and overflow.	-
FIFO_20	Combinations of wr_en,rd_en ,and underflow flag	-	cross cover values of write,read enables and underflow.	-
FIFO_21	Combinations of wr_en,rd_en ,and wr_ack flag	-	cross cover values of write,read enables and wr_ack.	-

2- UVM Testbench Structure:



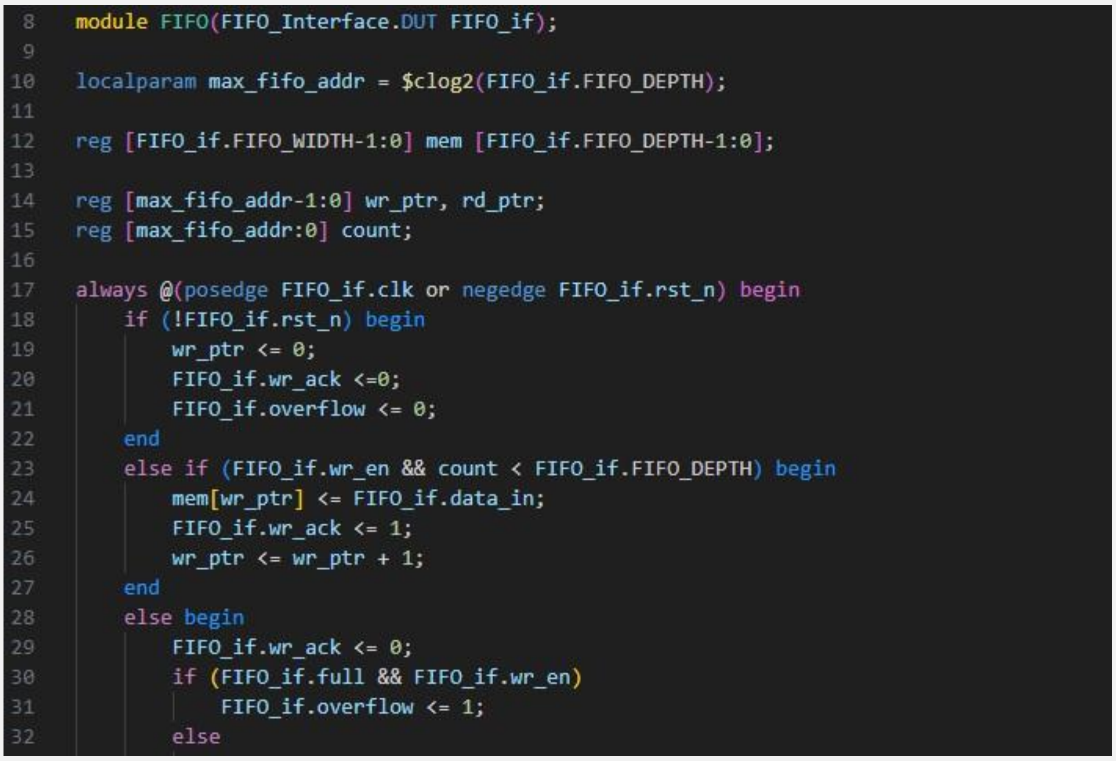
UVM testbench flow:

1. Top Module generates the clock, instantiates DUT Interface, DUT, and binds assertions module (which checks for FIFO flags and internal signals) then it passes virtual interface pointing to the DUT interface to Configuration database and call global task run_test to create and run the testbench.

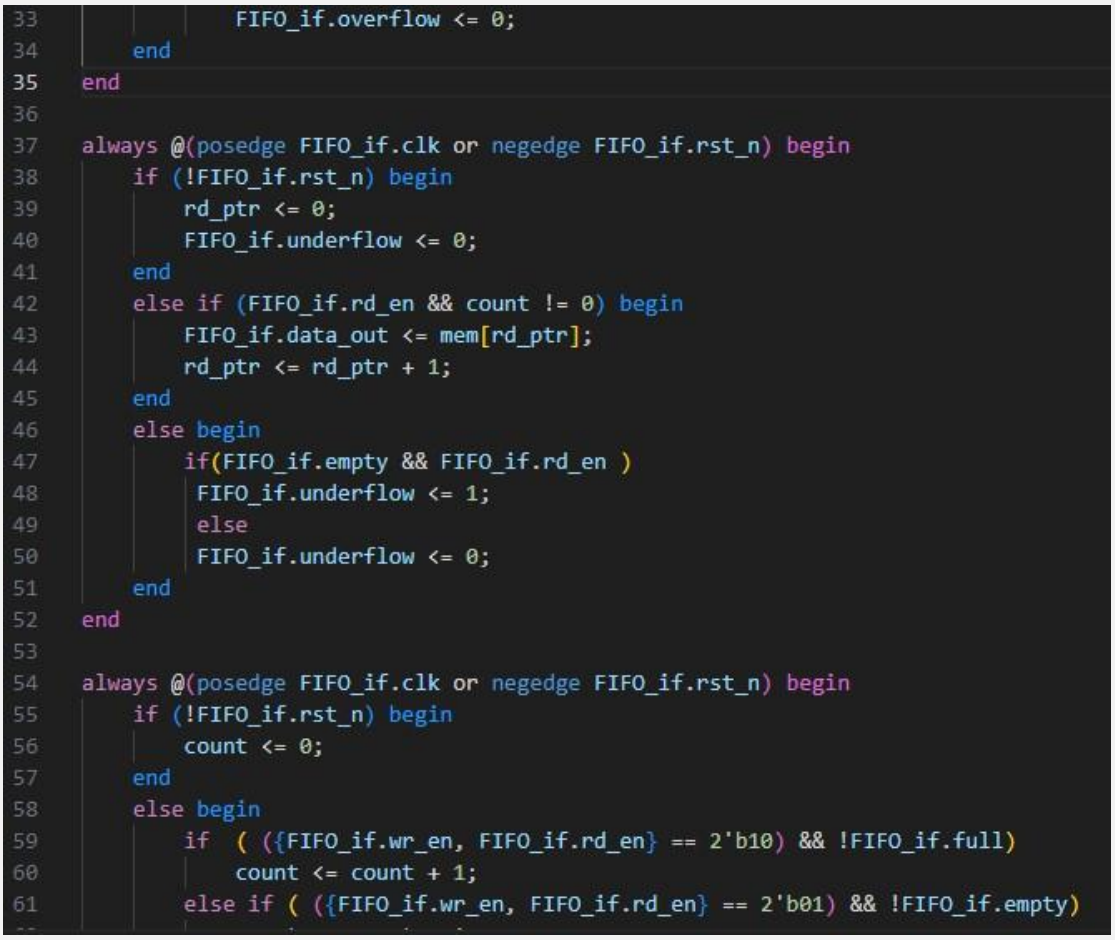
2. Test build the environment component and the sequences (Reset Sequence – Write Sequence – Read Sequence – Write Read Sequence). Then test retrieves the virtual interface to the virtual interface of Configuration

object and passes the configuration object to configuration database then it starts the sequences on the sequencer.
3. Environment Builds the Agent, coverage collector,and scoreboard (Reference Model) then connects the analysis port of the agent with scoreboard and coverage collector analysis ports.
4. Agent builds Driver , Sequencer ,and Monitor Components and connects the sequencer to the driver and it retrieves the configuration object interface and assign it to monitor and driver interfaces and connects the monitor analysis port with agent analysis port.
5. When Test starts sequence on the sequencer the sequence items sent to sequencer and the sequencer passes it to the driver when it requests the sequence item.
6. Driver Component requests sequence item from sequencer it drives it to the interface.
7. Monitor Component monitors the DUT Interface activity and translates it to transactions then it broadcasts them to the analysis component (coverage collector – scoreboard).
8. Coverage Collector get the sequence items from monitor then it samples the data for functional coverage.
9. Scoreboard gets the sequence items from monitor then checks correct functionality of data_out by reference model which calculates the data_out reference and compares it with the DUT data_out.

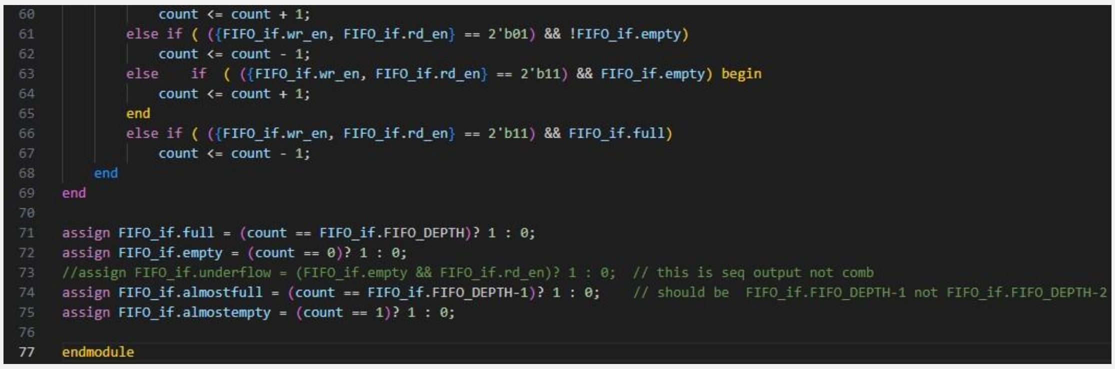
3- RTL:



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

localparam max_fifo_addr = $clog2(FIFO_if.FIFO_DEPTH);

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

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

always @(posedge FIFO_if.clk or negedge FIFO_if.rst_n) begin
    if (!FIFO_if.rst_n) begin
        wr_ptr <= 0;
        FIFO_if.wr_ack <=0;
        FIFO_if.overflow <= 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;
    end
    else begin
        FIFO_if.wr_ack <= 0;
        if (FIFO_if.full && FIFO_if.wr_en)
            FIFO_if.overflow <= 1;
        else
```

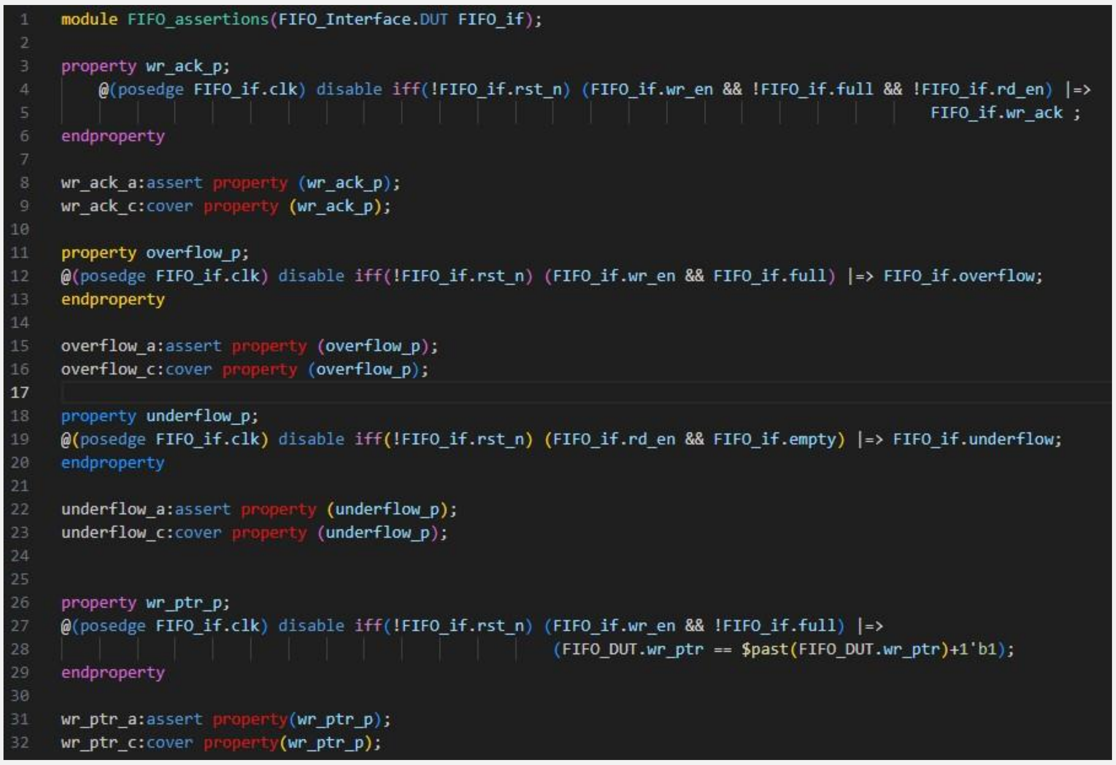


```
            FIFO_if.overflow <= 0;
    end
end

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

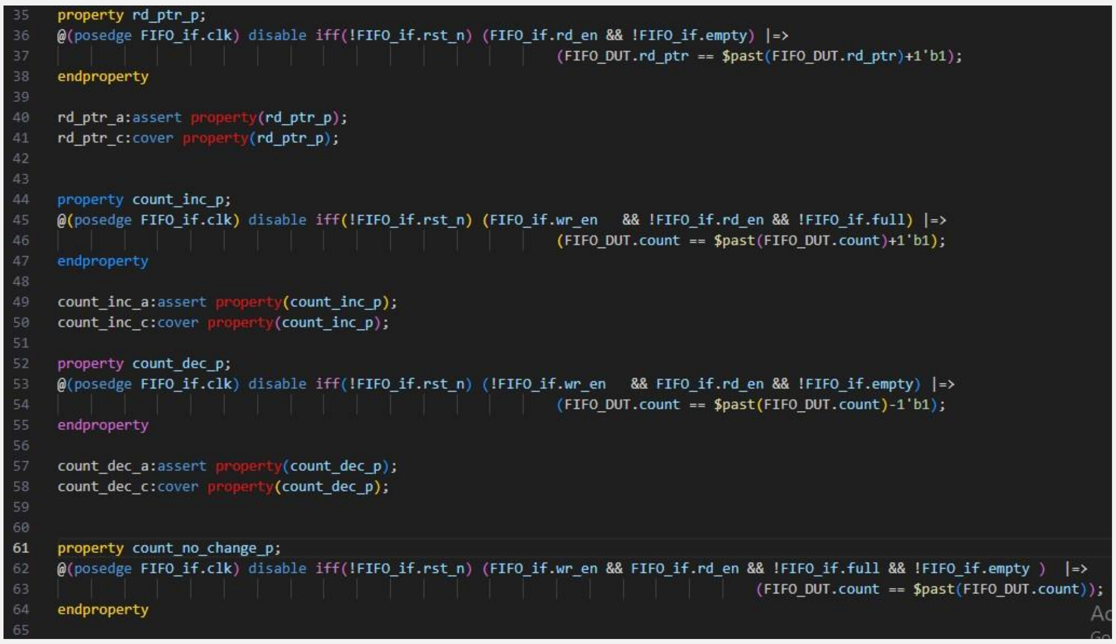


```
            count <= count + 1;
        else if ( ({FIFO_if.wr_en, FIFO_if.rd_en} == 2'b01) && !FIFO_if.empty)
            count <= count - 1;
        else    if  ( ({FIFO_if.wr_en, FIFO_if.rd_en} == 2'b11) && FIFO_if.empty) begin
            count <= count + 1;
        end
        else if ( ({FIFO_if.wr_en, 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.underflow = (FIFO_if.empty && FIFO_if.rd_en)? 1 : 0;  // this is seq output not comb
assign FIFO_if.almostfull = (count == FIFO_if.FIFO_DEPTH-1)? 1 : 0;    // should be  FIFO_if.FIFO_DEPTH-1 not FIFO_if.FIFO_DEPTH-2
assign FIFO_if.almostempty = (count == 1)? 1 : 0;

endmodule
```

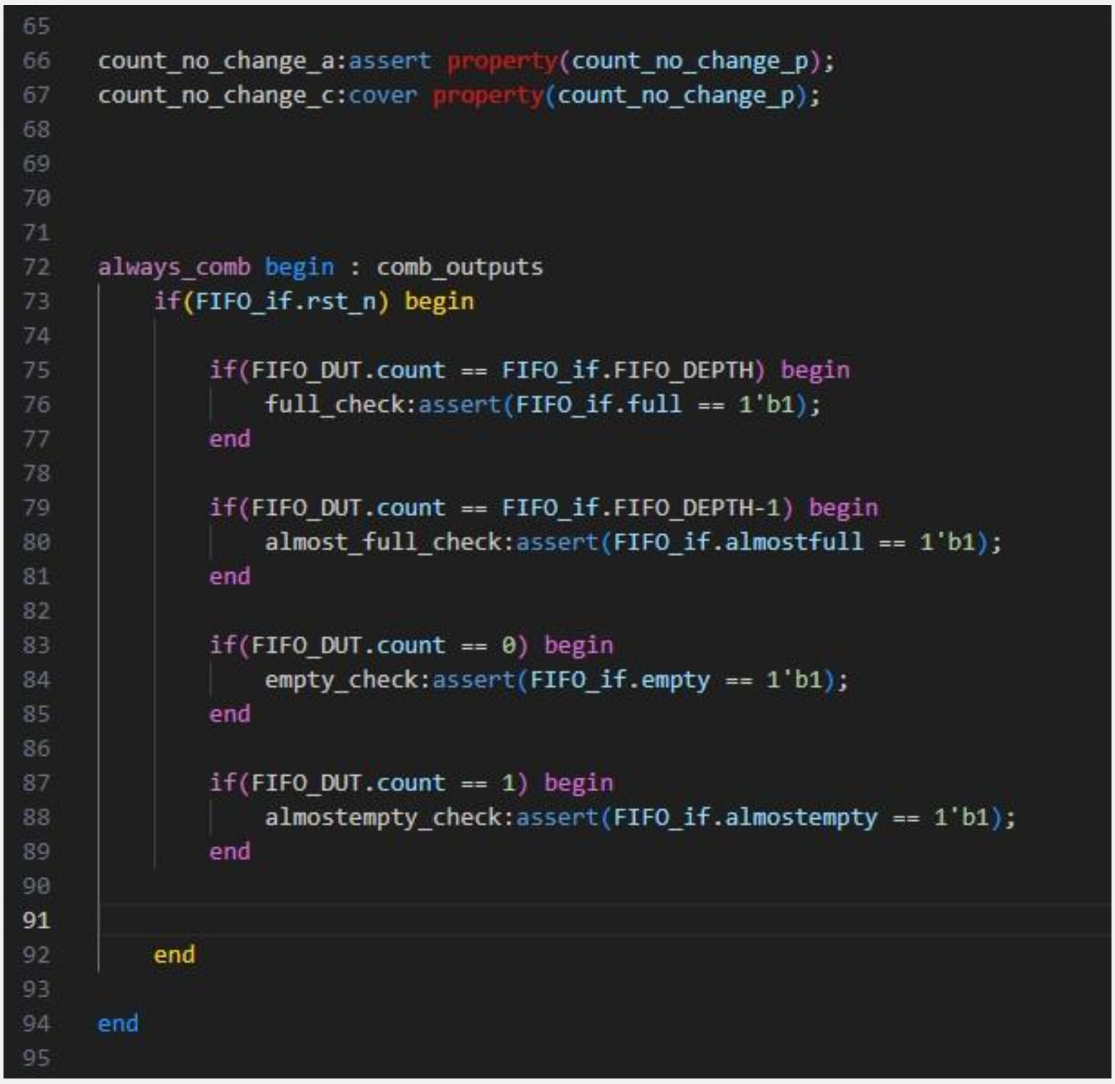
4- Assertions:



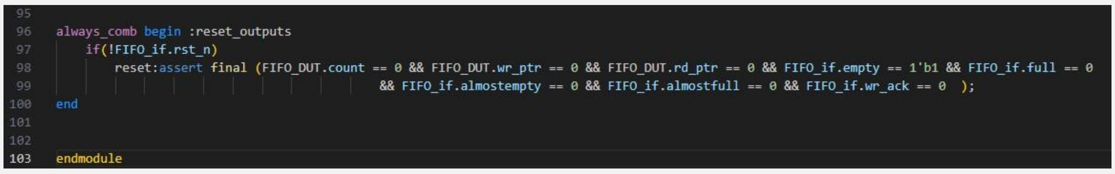
```
module FIFO_assertions(FIFO_Interface.DUT FIFO_if);

property wr_ack_p;
    @(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && !FIFO_if.full && !FIFO_if.rd_en) |=>
                                                                                                FIFO_if.wr_ack ;
endproperty

wr_ack_a:assert property (wr_ack_p);
wr_ack_c:cover property (wr_ack_p);

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

overflow_a:assert property (overflow_p);
overflow_c:cover property (overflow_p);

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

underflow_a:assert property (underflow_p);
underflow_c:cover property (underflow_p);


property wr_ptr_p;
@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && !FIFO_if.full) |=>
                                                    (FIFO_DUT.wr_ptr == $past(FIFO_DUT.wr_ptr)+1'b1);
endproperty

wr_ptr_a:assert property(wr_ptr_p);
wr_ptr_c:cover property(wr_ptr_p);
```



```
property rd_ptr_p;
@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.rd_en && !FIFO_if.empty) |=>
                                                    (FIFO_DUT.rd_ptr == $past(FIFO_DUT.rd_ptr)+1'b1);
endproperty

rd_ptr_a:assert property(rd_ptr_p);
rd_ptr_c:cover property(rd_ptr_p);


property count_inc_p;
@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en   && !FIFO_if.rd_en && !FIFO_if.full) |=>
                                                    (FIFO_DUT.count == $past(FIFO_DUT.count)+1'b1);
endproperty

count_inc_a:assert property(count_inc_p);
count_inc_c:cover property(count_inc_p);

property count_dec_p;
@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (!FIFO_if.wr_en   && FIFO_if.rd_en && !FIFO_if.empty) |=>
                                                    (FIFO_DUT.count == $past(FIFO_DUT.count)-1'b1);
endproperty

count_dec_a:assert property(count_dec_p);
count_dec_c:cover property(count_dec_p);


property count_no_change_p;
@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && FIFO_if.rd_en && !FIFO_if.full && !FIFO_if.empty )  |=>
                                                                (FIFO_DUT.count == $past(FIFO_DUT.count));
endproperty
```

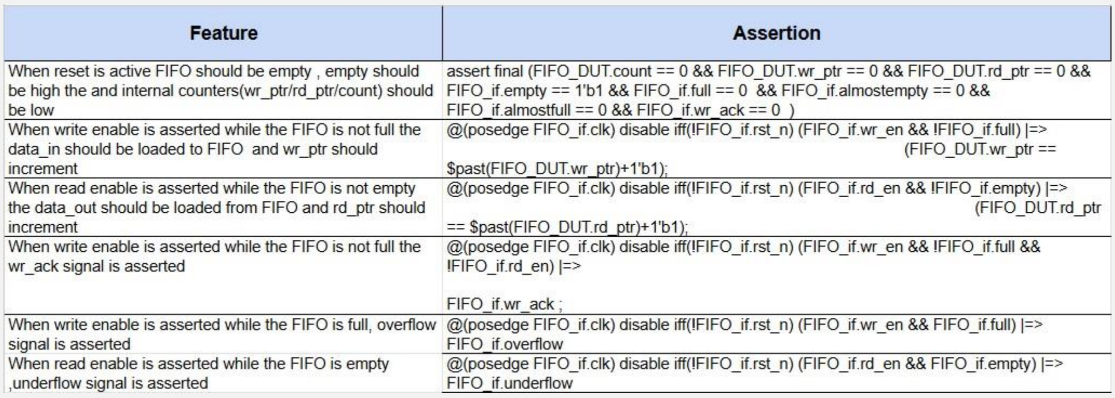


```systemverilog

count_no_change_a:assert property(count_no_change_p);
count_no_change_c:cover property(count_no_change_p);




always_comb begin : comb_outputs
    if(FIFO_if.rst_n) begin

        if(FIFO_DUT.count == FIFO_if.FIFO_DEPTH) begin
            full_check:assert(FIFO_if.full == 1'b1);
        end

        if(FIFO_DUT.count == FIFO_if.FIFO_DEPTH-1) begin
            almost_full_check:assert(FIFO_if.almostfull == 1'b1);
        end

        if(FIFO_DUT.count == 0) begin
            empty_check:assert(FIFO_if.empty == 1'b1);
        end

        if(FIFO_DUT.count == 1) begin
            almostempty_check:assert(FIFO_if.almostempty == 1'b1);
        end


    end

end

```



```systemverilog

always_comb begin :reset_outputs
    if(!FIFO_if.rst_n)
        reset:assert final (FIFO_DUT.count == 0 && FIFO_DUT.wr_ptr == 0 && FIFO_DUT.rd_ptr == 0 && FIFO_if.empty == 1'b1 && FIFO_if.full == 0
                                          && FIFO_if.almostempty == 0 && FIFO_if.almostfull == 0 && FIFO_if.wr_ack == 0  );
end


endmodule
```

5- Assertions Table:

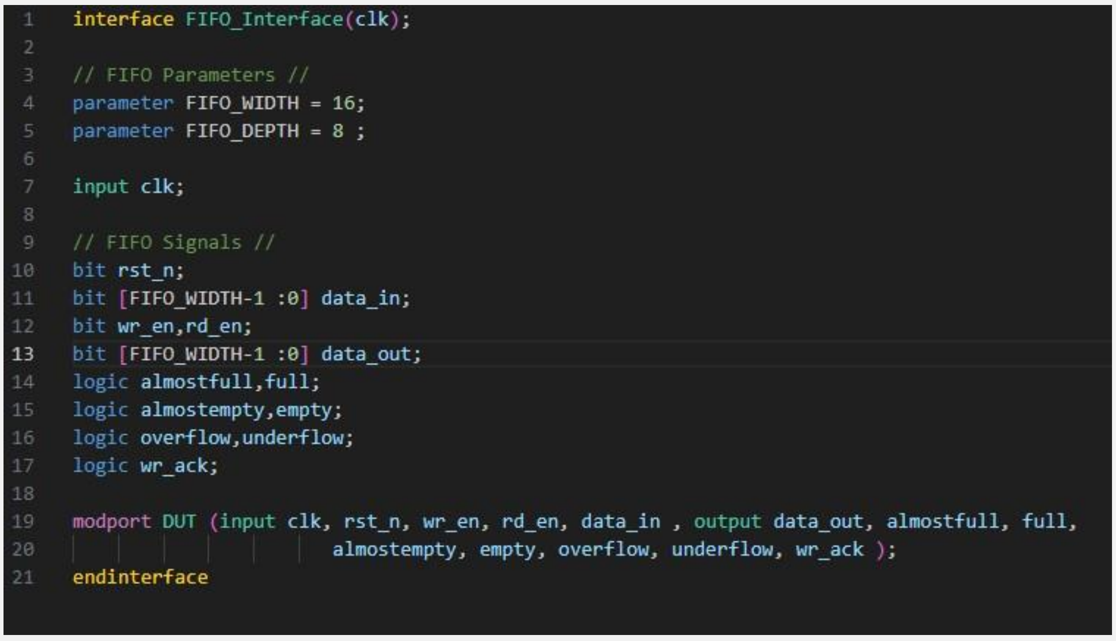


Feature	Assertion
When reset is active FIFO should be empty , empty should be high the and internal counters(wr_ptr/rd_ptr/count) should be low	assert final (FIFO_DUT.count == 0 && FIFO_DUT.wr_ptr == 0 && FIFO_DUT.rd_ptr == 0 && FIFO_if.empty == 1'b1 && FIFO_if.full == 0 && FIFO_if.almostempty == 0 && FIFO_if.almostfull == 0 && FIFO_if.wr_ack == 0)
When write enable is asserted while the FIFO is not full the data_in should be loaded to FIFO and wr_ptr should increment	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && !FIFO_if.full) |=> (FIFO_DUT.wr_ptr == $past(FIFO_DUT.wr_ptr)+1'b1);
When read enable is asserted while the FIFO is not empty the data_out should be loaded from FIFO and rd_ptr should increment	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.rd_en && !FIFO_if.empty) |=> (FIFO_DUT.rd_ptr == $past(FIFO_DUT.rd_ptr)+1'b1);
When write enable is asserted while the FIFO is not full the wr_ack signal is asserted	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && !FIFO_if.full && !FIFO_if.rd_en) |=> FIFO_if.wr_ack ;
When write enable is asserted while the FIFO is full, overflow signal is asserted	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && FIFO_if.full) |=> FIFO_if.overflow
When read enable is asserted while the FIFO is empty ,underflow signal is asserted	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.rd_en && FIFO_if.empty) |=> FIFO_if.underflow

When write enable is asserted and read enable is deasserted and FIFO is not full the internal signal count should increment	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && !FIFO_if.rd_en && !FIFO_if.full) |=> (FIFO_DUT.count == $past(FIFO_DUT.count)+1'b1)
When read enable is Deasserted and write enable is asserted while the FIFO is not empty the internal signal count should decrement	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (!FIFO_if.wr_en && FIFO_if.rd_en && !FIFO_if.empty) |=> (FIFO_DUT.count == $past(FIFO_DUT.count)-1'b1)
When read enable is asserted and write enable is asserted while the FIFO is neither full nor empty	@(posedge FIFO_if.clk) disable iff(!FIFO_if.rst_n) (FIFO_if.wr_en && FIFO_if.rd_en && !FIFO_if.full && !FIFO_if.empty) |=> (FIFO_DUT.count == $past(FIFO_DUT.count))
When Count equals to zero means the FIFO is empty	if(FIFO_DUT.count == 0) begin empty_check:assert(FIFO_if.empty == 1'b1); end
When Count equals to one means the FIFO is written once so its almostempty	if(FIFO_DUT.count == 1) begin almostempty_check:assert(FIFO_if.almostempty == 1'b1); end
When Count equals to FIFO_DEPTH means the FIFO is written FIFO_DEPTH times so its full	if(FIFO_DUT.count == FIFO_if.FIFO_DEPTH) begin full_check:assert(FIFO_if.full == 1'b1); end

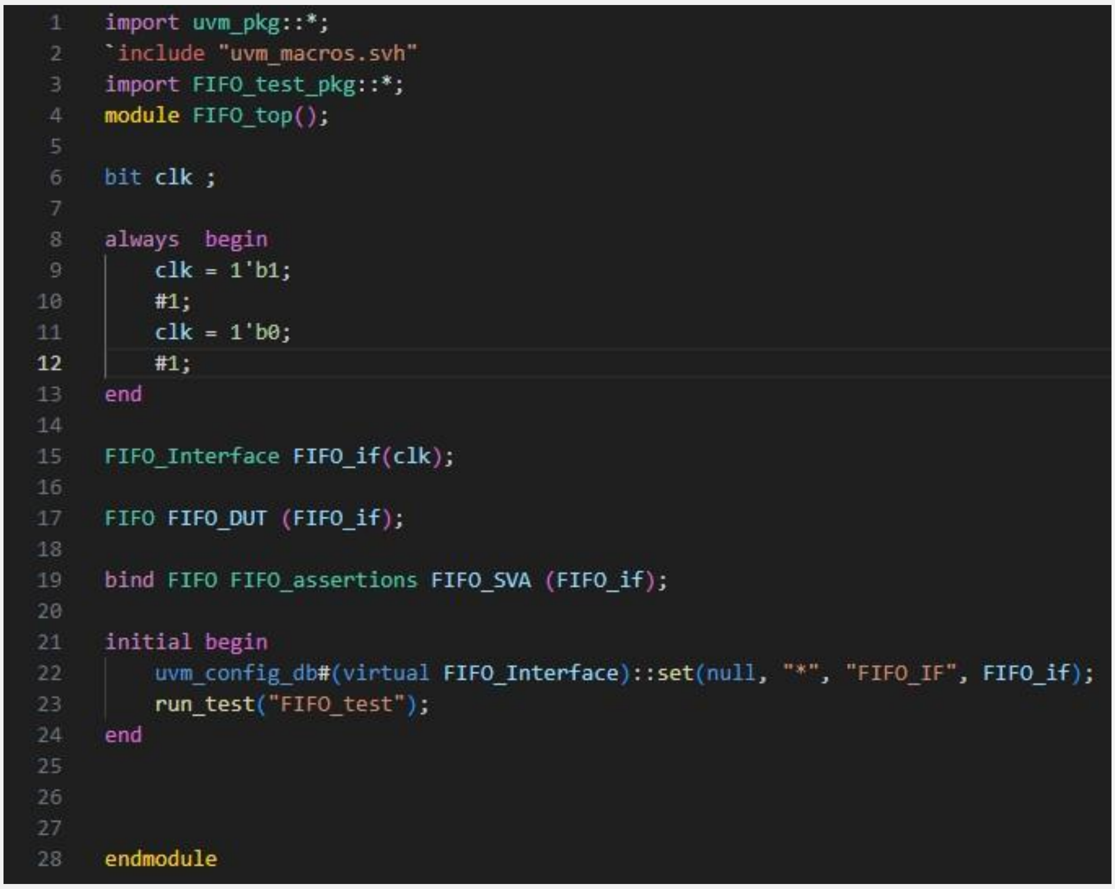
When Count equals to FIFO_DEPTH means the FIFO is written FIFO_DEPTH times so its full	if(FIFO_DUT.count == FIFO_if.FIFO_DEPTH) begin full_check:assert(FIFO_if.full == 1'b1); end
When Count equals to FIFO_DEPTH-1 means the FIFO is written FIFO_DEPTH-1 times so it's almostfull	if(FIFO_DUT.count == FIFO_if.FIFO_DEPTH-1) begin almost_full_check:assert(FIFO_if.almostfull == 1'b1); end

6- Interface :


```
interface FIFO_Interface(clk);

// FIFO Parameters //
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8 ;

input clk;

// FIFO Signals //
bit rst_n;
bit [FIFO_WIDTH-1 :0] data_in;
bit wr_en,rd_en;
bit [FIFO_WIDTH-1 :0] data_out;
logic almostfull,full;
logic almostempty,empty;
logic overflow,underflow;
logic wr_ack;

modport DUT (input clk, rst_n, wr_en, rd_en, data_in , output data_out, almostfull, full,
                        almostempty, empty, overflow, underflow, wr_ack );
endinterface
```

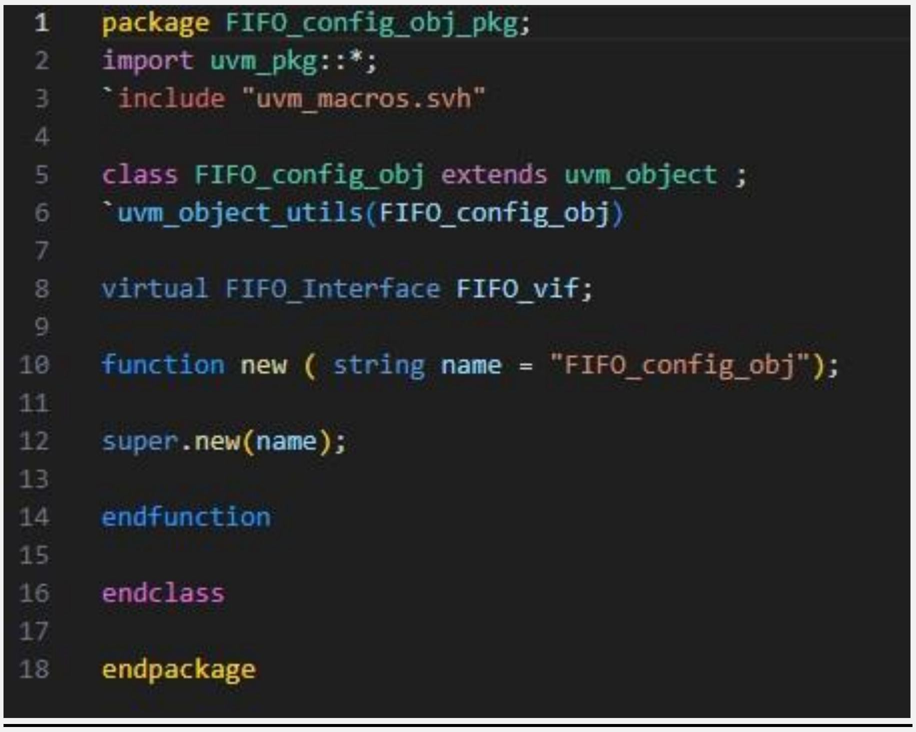
7- Verification Environment:

1.Top Module:



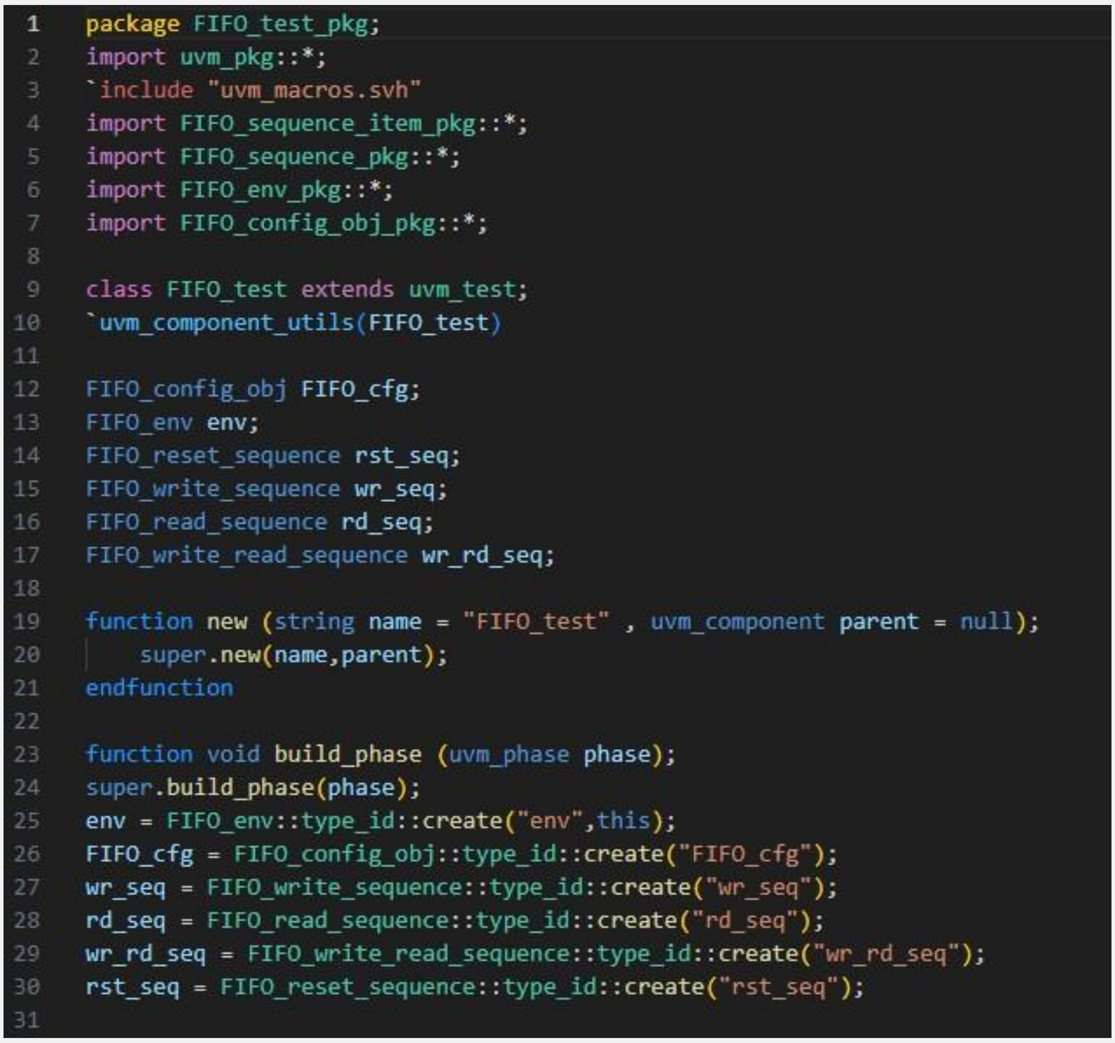
```
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_test_pkg::*;
module FIFO_top();

bit clk ;

always  begin
    clk = 1'b1;
    #1;
    clk = 1'b0;
    #1;
end

FIFO_Interface FIFO_if(clk);

FIFO FIFO_DUT (FIFO_if);

bind FIFO FIFO_assertions FIFO_SVA (FIFO_if);

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



endmodule
```

2. Configuration Object:

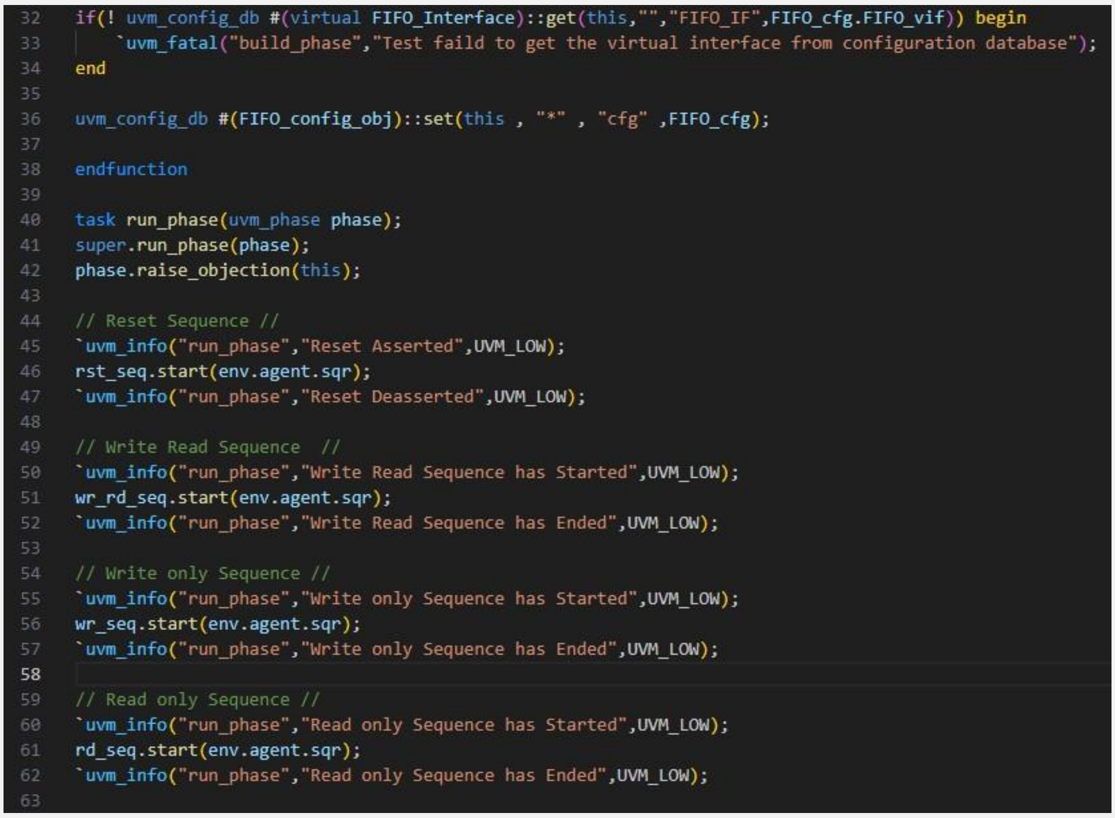


```
package FIFO_config_obj_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_config_obj extends uvm_object ;
`uvm_object_utils(FIFO_config_obj)

virtual FIFO_Interface FIFO_vif;

function new ( string name = "FIFO_config_obj");

super.new(name);

endfunction

endclass

endpackage
```

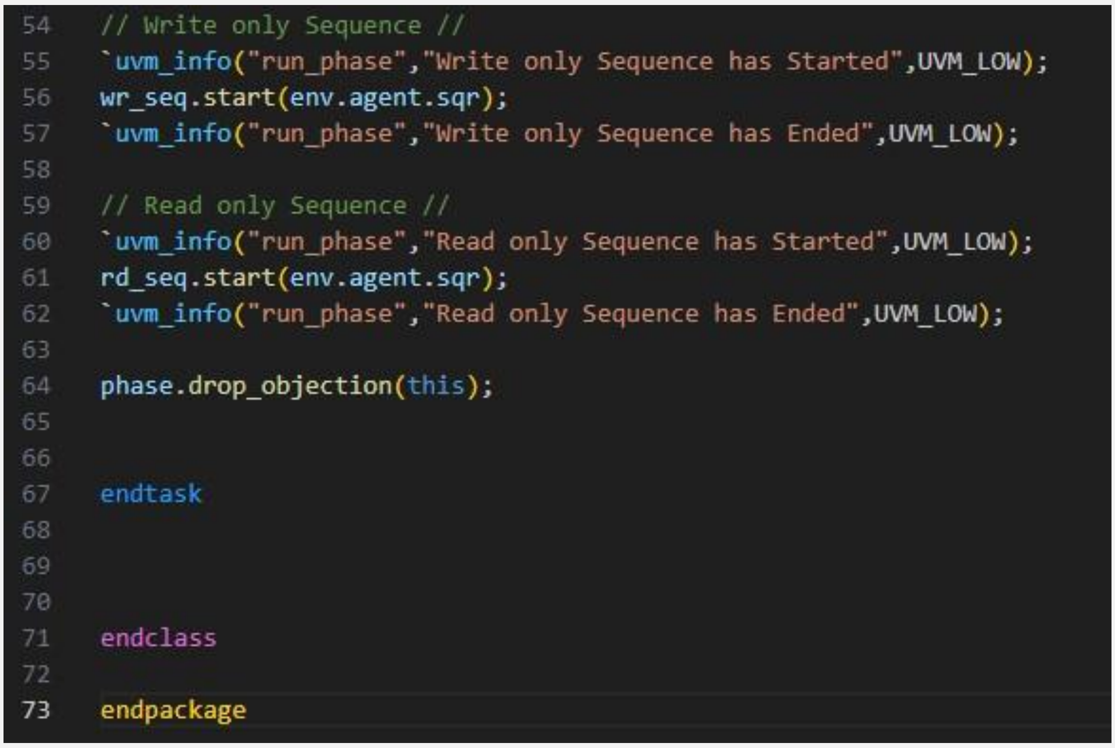
3. Test:



```
package FIFO_test_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;
import FIFO_sequence_pkg::*;
import FIFO_env_pkg::*;
import FIFO_config_obj_pkg::*;

class FIFO_test extends uvm_test;
`uvm_component_utils(FIFO_test)

FIFO_config_obj FIFO_cfg;
FIFO_env env;
FIFO_reset_sequence rst_seq;
FIFO_write_sequence wr_seq;
FIFO_read_sequence rd_seq;
FIFO_write_read_sequence wr_rd_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);
env = FIFO_env::type_id::create("env",this);
FIFO_cfg = FIFO_config_obj::type_id::create("FIFO_cfg");
wr_seq = FIFO_write_sequence::type_id::create("wr_seq");
rd_seq = FIFO_read_sequence::type_id::create("rd_seq");
wr_rd_seq = FIFO_write_read_sequence::type_id::create("wr_rd_seq");
rst_seq = FIFO_reset_sequence::type_id::create("rst_seq");

```



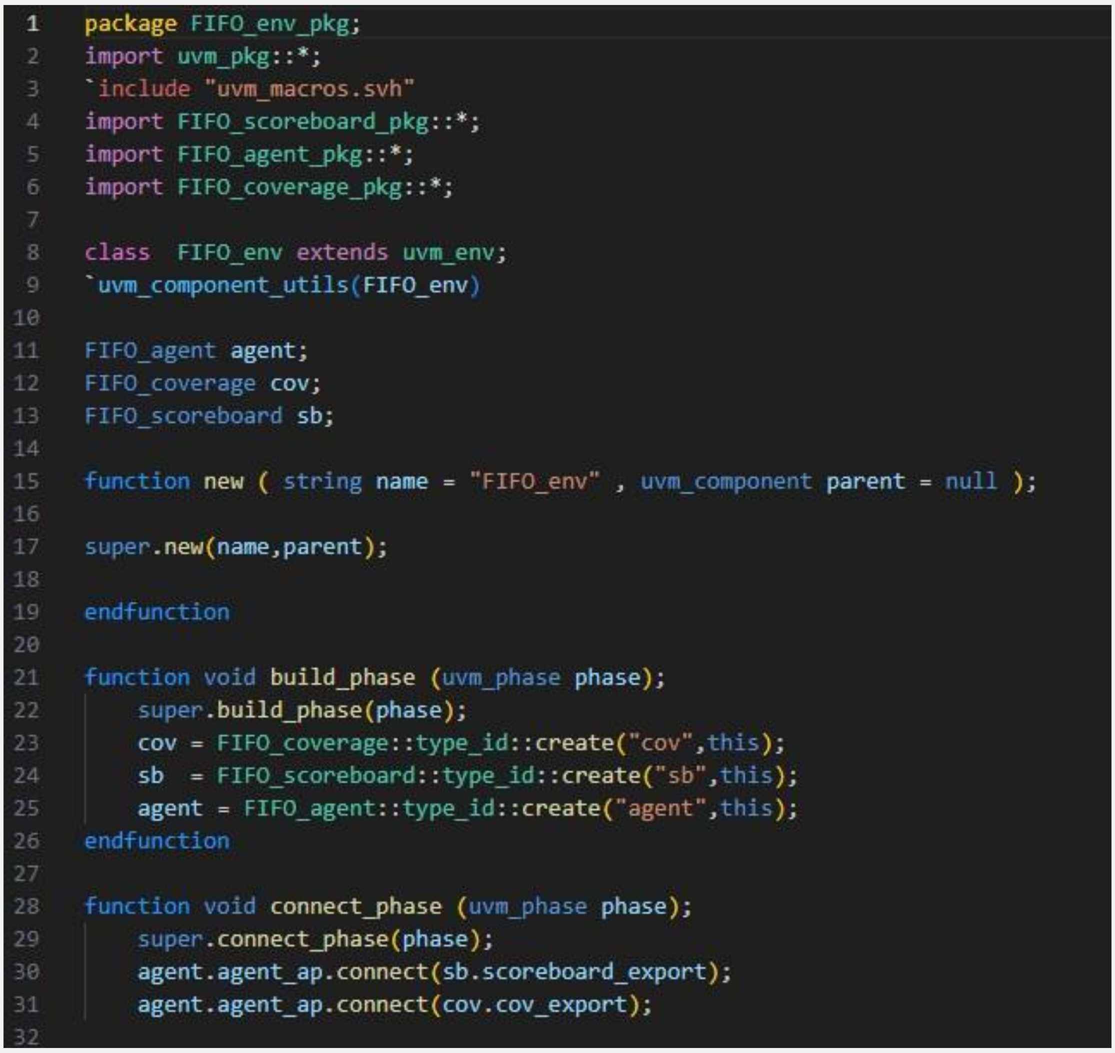
```
if(! uvm_config_db #(virtual FIFO_Interface)::get(this,"","FIFO_IF",FIFO_cfg.FIFO_vif)) begin
    `uvm_fatal("build_phase","Test faild to get the virtual interface from configuration database");
end

uvm_config_db #(FIFO_config_obj)::set(this , "*" , "cfg" ,FIFO_cfg);

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);
rst_seq.start(env.agent.sqr);
`uvm_info("run_phase","Reset Deasserted",UVM_LOW);

// Write Read Sequence  //
`uvm_info("run_phase","Write Read Sequence has Started",UVM_LOW);
wr_rd_seq.start(env.agent.sqr);
`uvm_info("run_phase","Write Read Sequence has Ended",UVM_LOW);

// Write only Sequence //
`uvm_info("run_phase","Write only Sequence has Started",UVM_LOW);
wr_seq.start(env.agent.sqr);
`uvm_info("run_phase","Write only Sequence has Ended",UVM_LOW);

// Read only Sequence //
`uvm_info("run_phase","Read only Sequence has Started",UVM_LOW);
rd_seq.start(env.agent.sqr);
`uvm_info("run_phase","Read only Sequence has Ended",UVM_LOW);

```



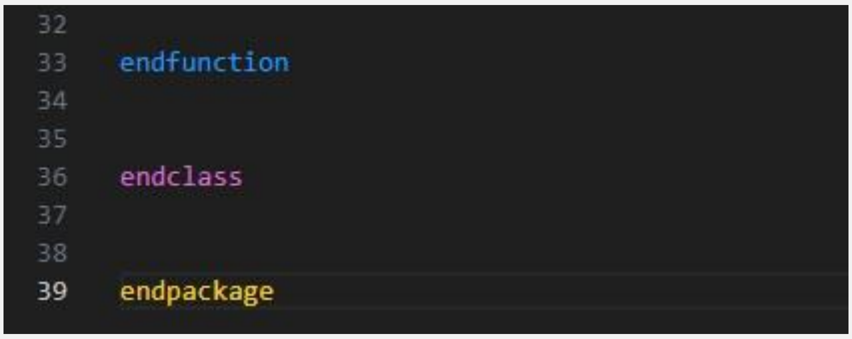
```
// Write only Sequence //
`uvm_info("run_phase","Write only Sequence has Started",UVM_LOW);
wr_seq.start(env.agent.sqr);
`uvm_info("run_phase","Write only Sequence has Ended",UVM_LOW);

// Read only Sequence //
`uvm_info("run_phase","Read only Sequence has Started",UVM_LOW);
rd_seq.start(env.agent.sqr);
`uvm_info("run_phase","Read only Sequence has Ended",UVM_LOW);
```

```

phase.drop_objection(this);


endtask



endclass

endpackage
```

4. Environment:

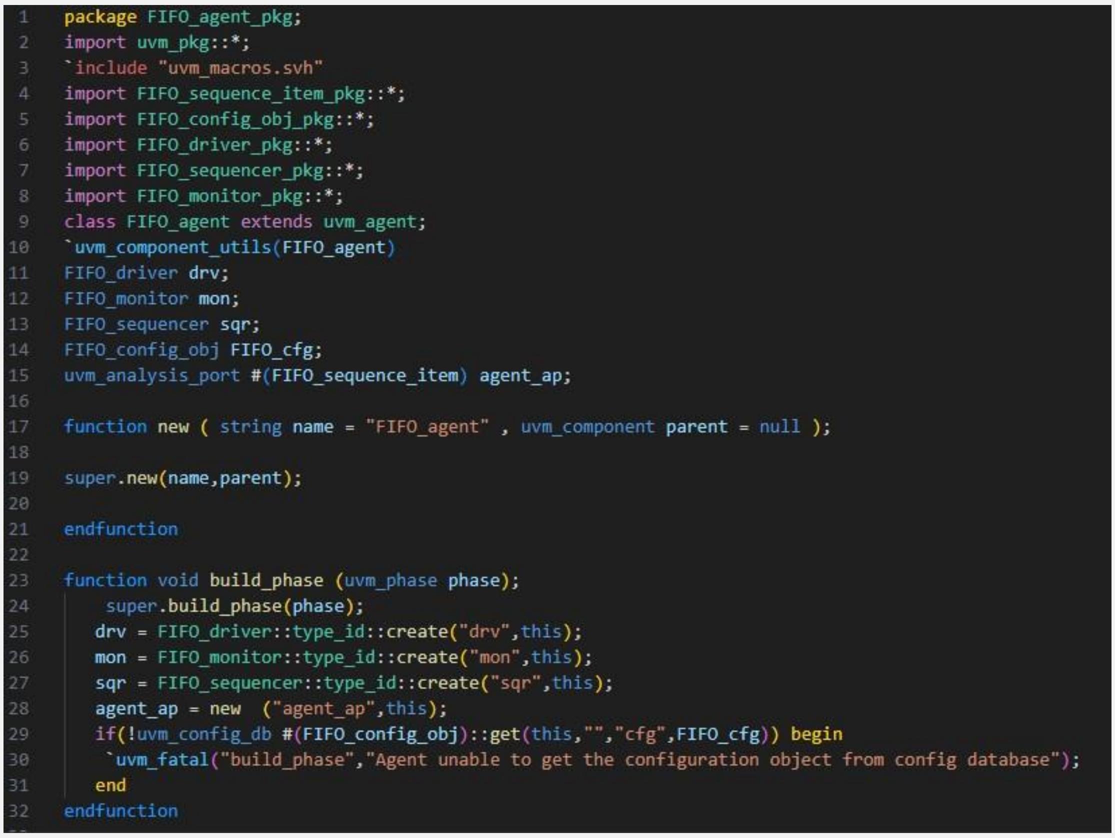


```
package FIFO_env_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_scoreboard_pkg::*;
import FIFO_agent_pkg::*;
import FIFO_coverage_pkg::*;

class  FIFO_env extends uvm_env;
`uvm_component_utils(FIFO_env)

FIFO_agent agent;
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);
    cov = FIFO_coverage::type_id::create("cov",this);
    sb  = FIFO_scoreboard::type_id::create("sb",this);
    agent = FIFO_agent::type_id::create("agent",this);
endfunction

function void connect_phase (uvm_phase phase);
    super.connect_phase(phase);
    agent.agent_ap.connect(sb.scoreboard_export);
    agent.agent_ap.connect(cov.cov_export);

```

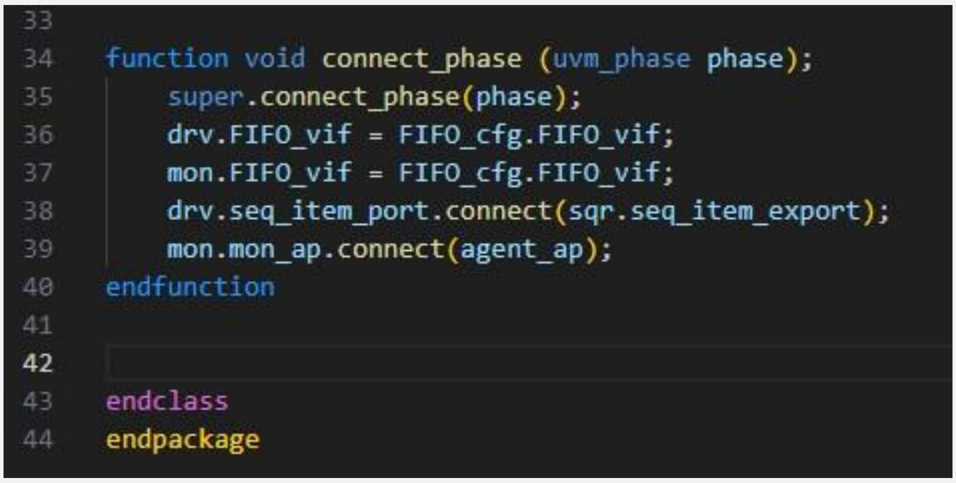


```

endfunction


endclass


endpackage
```

5- Agent:

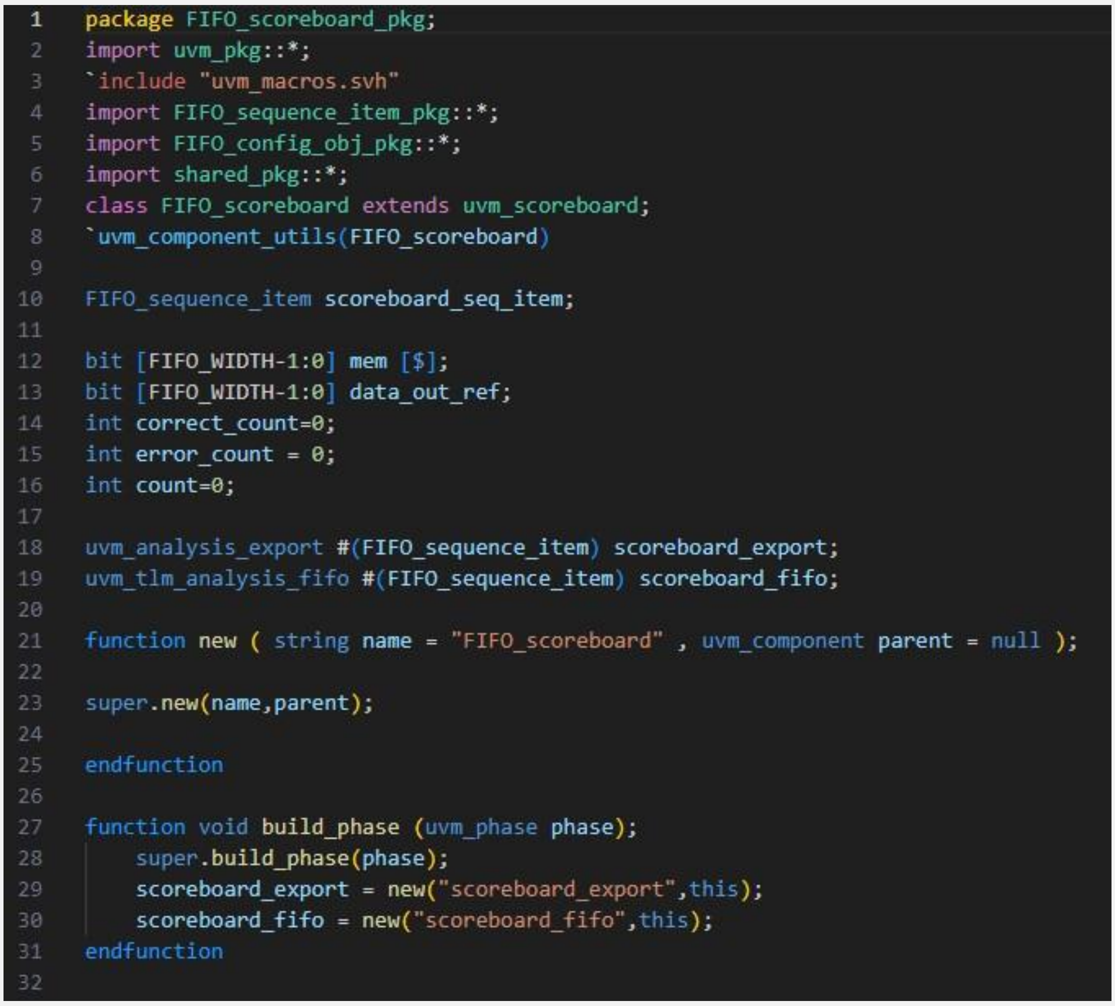


```
package FIFO_agent_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;
import FIFO_config_obj_pkg::*;
import FIFO_driver_pkg::*;
import FIFO_sequencer_pkg::*;
import FIFO_monitor_pkg::*;
class FIFO_agent extends uvm_agent;
`uvm_component_utils(FIFO_agent)
FIFO_driver drv;
FIFO_monitor mon;
FIFO_sequencer sqr;
FIFO_config_obj FIFO_cfg;
uvm_analysis_port #(FIFO_sequence_item) agent_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);
    drv = FIFO_driver::type_id::create("drv",this);
    mon = FIFO_monitor::type_id::create("mon",this);
    sqr = FIFO_sequencer::type_id::create("sqr",this);
    agent_ap = new  ("agent_ap",this);
    if(!uvm_config_db #(FIFO_config_obj)::get(this,"","cfg",FIFO_cfg)) begin
     `uvm_fatal("build_phase","Agent unable to get the configuration object from config database");
    end
endfunction
```

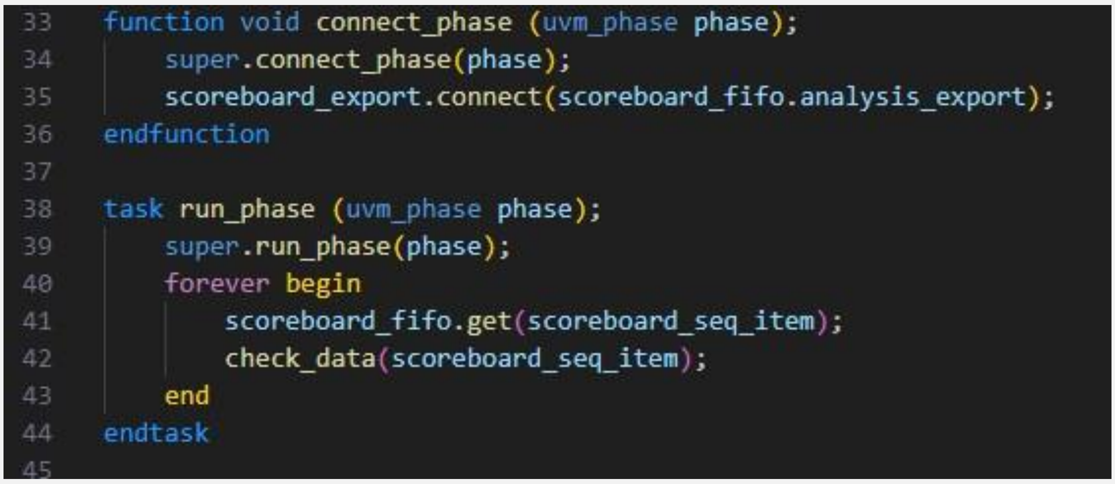


```

function void connect_phase (uvm_phase phase);
    super.connect_phase(phase);
    drv.FIFO_vif = FIFO_cfg.FIFO_vif;
    mon.FIFO_vif = FIFO_cfg.FIFO_vif;
    drv.seq_item_port.connect(sqr.seq_item_export);
    mon.mon_ap.connect(agent_ap);
endfunction


endclass
endpackage
```

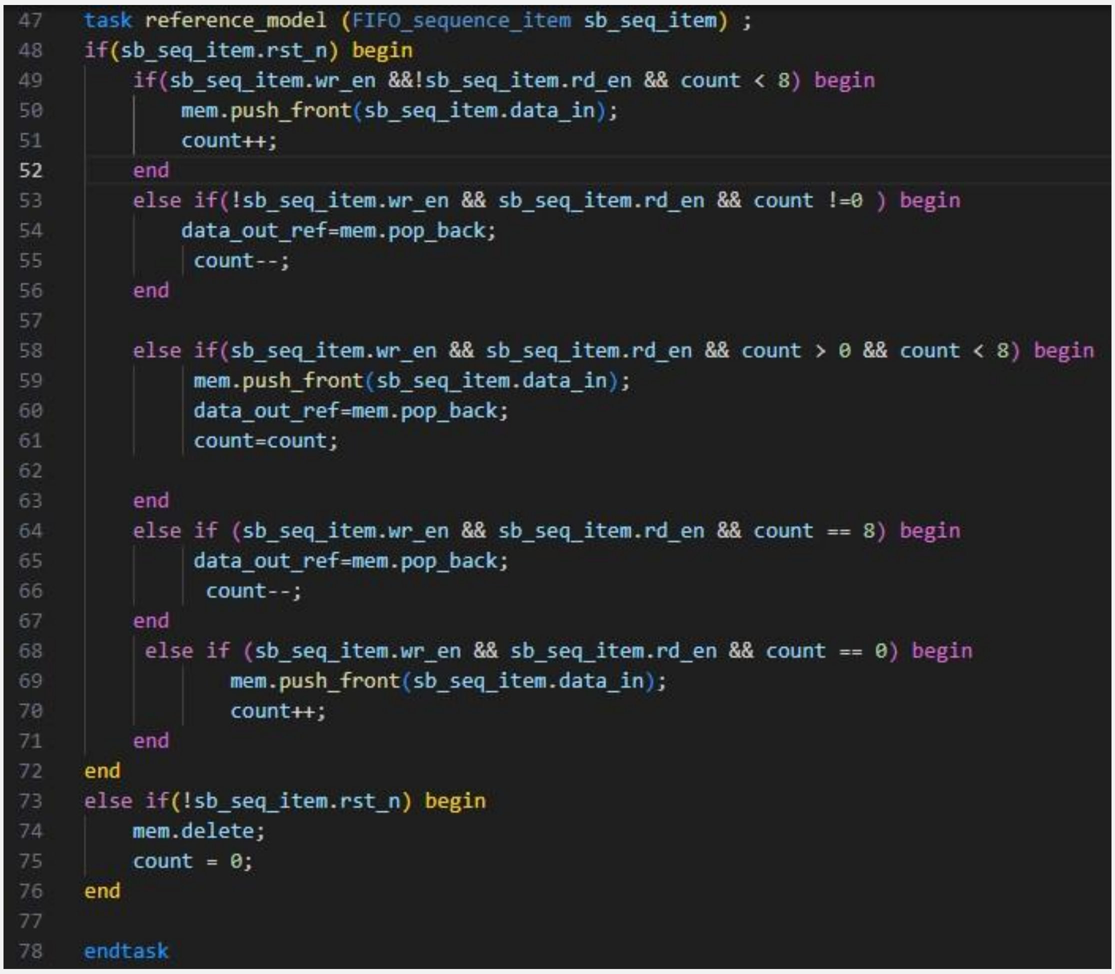
6. Scoreboard:



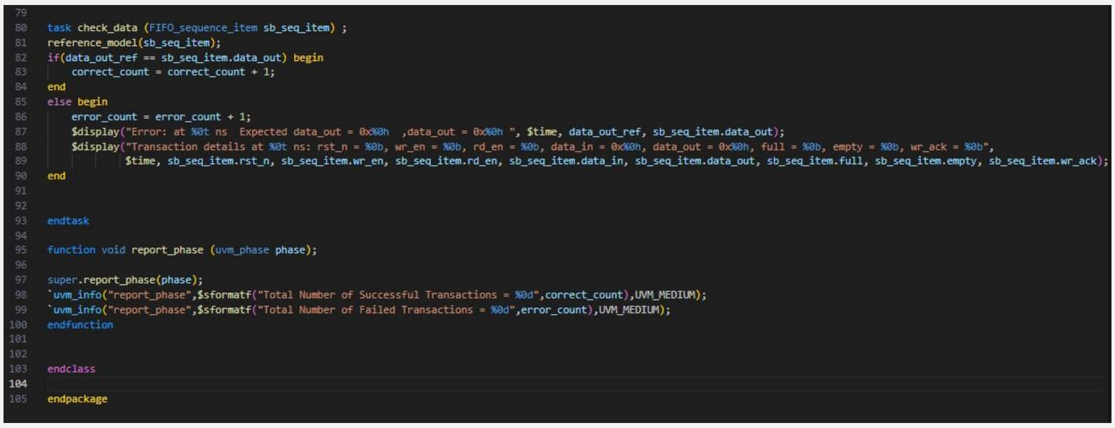
```
package FIFO_scoreboard_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;
import FIFO_config_obj_pkg::*;
import shared_pkg::*;
class FIFO_scoreboard extends uvm_scoreboard;
`uvm_component_utils(FIFO_scoreboard)

FIFO_sequence_item scoreboard_seq_item;

bit [FIFO_WIDTH-1:0] mem [$];
bit [FIFO_WIDTH-1:0] data_out_ref;
int correct_count=0;
int error_count = 0;
int count=0;

uvm_analysis_export #(FIFO_sequence_item) scoreboard_export;
uvm_tlm_analysis_fifo #(FIFO_sequence_item) scoreboard_fifo;

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);
    scoreboard_export = new("scoreboard_export",this);
    scoreboard_fifo = new("scoreboard_fifo",this);
endfunction

```



```
function void connect_phase (uvm_phase phase);
    super.connect_phase(phase);
    scoreboard_export.connect(scoreboard_fifo.analysis_export);
endfunction

task run_phase (uvm_phase phase);
    super.run_phase(phase);
    forever begin
        scoreboard_fifo.get(scoreboard_seq_item);
        check_data(scoreboard_seq_item);
    end
endtask

```

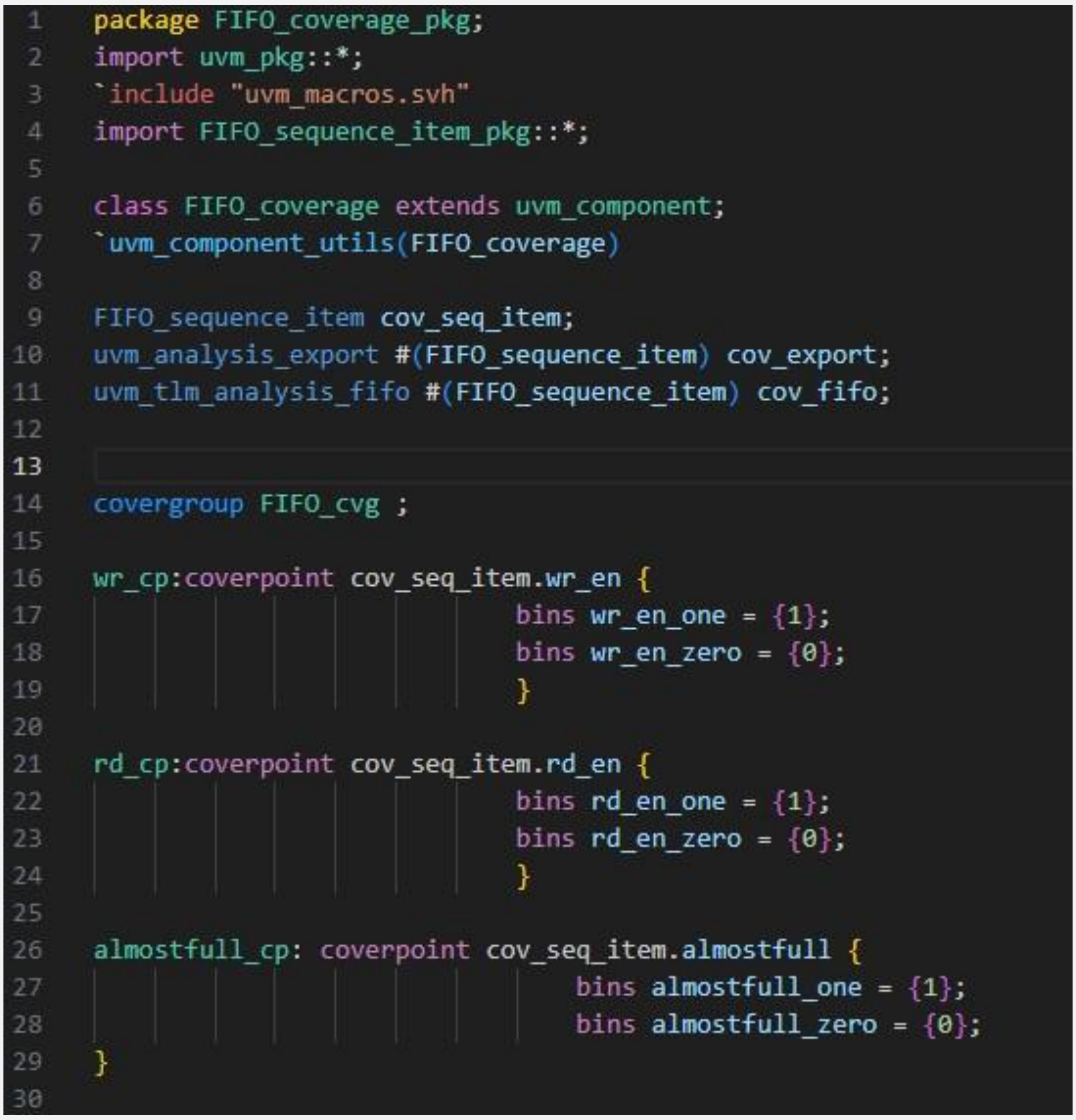


```
task reference_model (FIFO_sequence_item sb_seq_item) ;
if(sb_seq_item.rst_n) begin
    if(sb_seq_item.wr_en &&!sb_seq_item.rd_en && count < 8) begin
        mem.push_front(sb_seq_item.data_in);
        count++;
    end
    else if(!sb_seq_item.wr_en && sb_seq_item.rd_en && count !=0 ) begin
        data_out_ref=mem.pop_back;
         count--;
    end

    else if(sb_seq_item.wr_en && sb_seq_item.rd_en && count > 0 && count < 8) begin
         mem.push_front(sb_seq_item.data_in);
         data_out_ref=mem.pop_back;
         count=count;

    end
    else if (sb_seq_item.wr_en && sb_seq_item.rd_en && count == 8) begin
         data_out_ref=mem.pop_back;
          count--;
    end
     else if (sb_seq_item.wr_en && sb_seq_item.rd_en && count == 0) begin
            mem.push_front(sb_seq_item.data_in);
            count++;
    end
end
else if(!sb_seq_item.rst_n) begin
    mem.delete;
    count = 0;
end

endtask
```

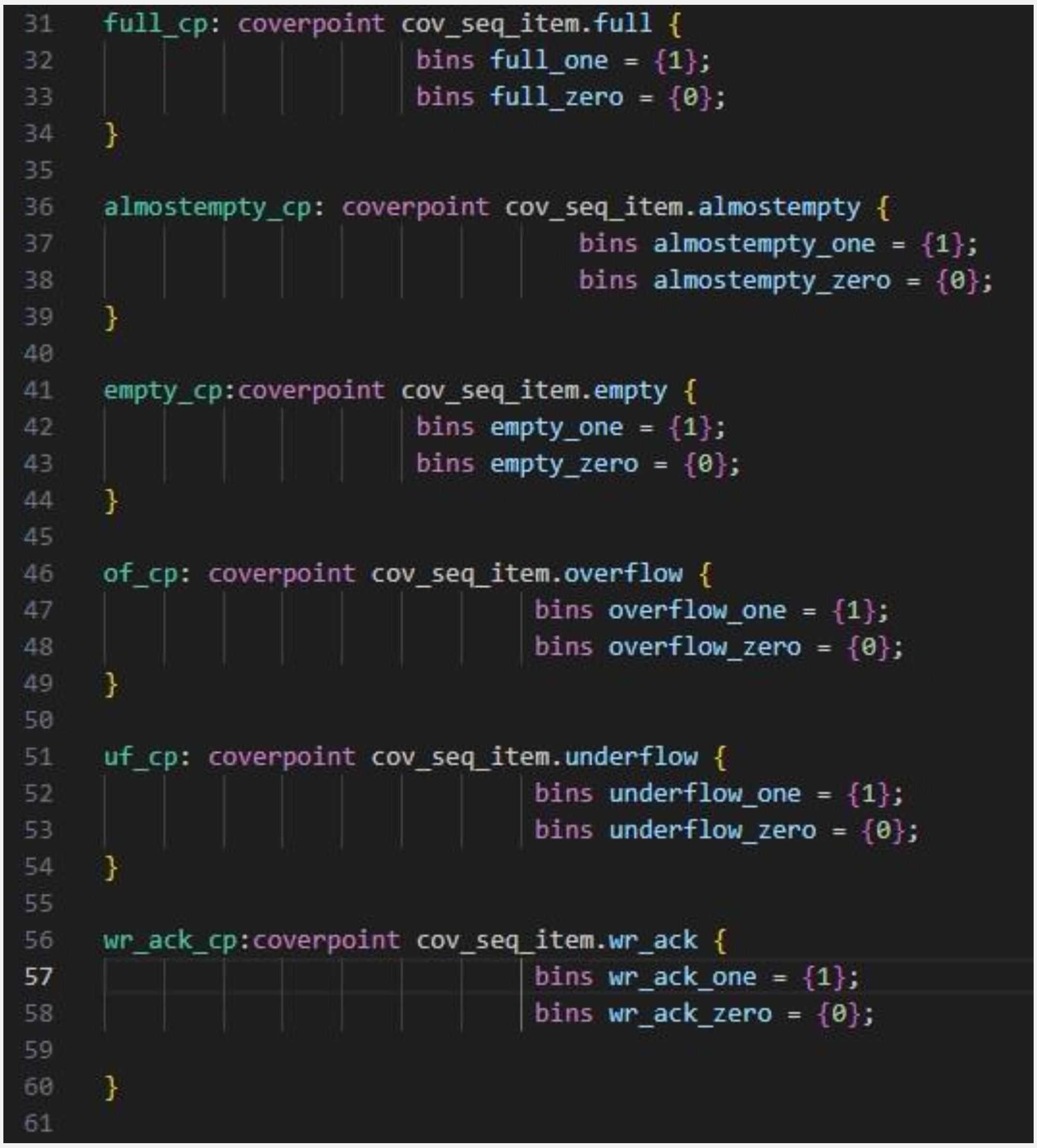


```

task check_data (FIFO_sequence_item sb_seq_item) ;
reference_model(sb_seq_item);
if(data_out_ref == sb_seq_item.data_out) begin
    correct_count = correct_count + 1;
end
else begin
    error_count = error_count + 1;
    $display("Error: at %0t ns  Expected data_out = 0x%0h  ,data_out = 0x%0h ", $time, data_out_ref, sb_seq_item.data_out);
    $display("Transaction details at %0t ns: rst_n = %0b, wr_en = %0b, rd_en = %0b, data_in = 0x%0h, data_out = 0x%0h, full = %0b, empty = %0b, wr_ack = %0b",
            $time, sb_seq_item.rst_n, sb_seq_item.wr_en, sb_seq_item.rd_en, sb_seq_item.data_in, sb_seq_item.data_out, sb_seq_item.full, sb_seq_item.empty, sb_seq_item.wr_ack);
end


endtask

function void report_phase (uvm_phase phase);

super.report_phase(phase);
`uvm_info("report_phase",$sformatf("Total Number of Successful Transactions = %0d",correct_count),UVM_MEDIUM);
`uvm_info("report_phase",$sformatf("Total Number of Failed Transactions = %0d",error_count),UVM_MEDIUM);
endfunction


endclass

endpackage
```

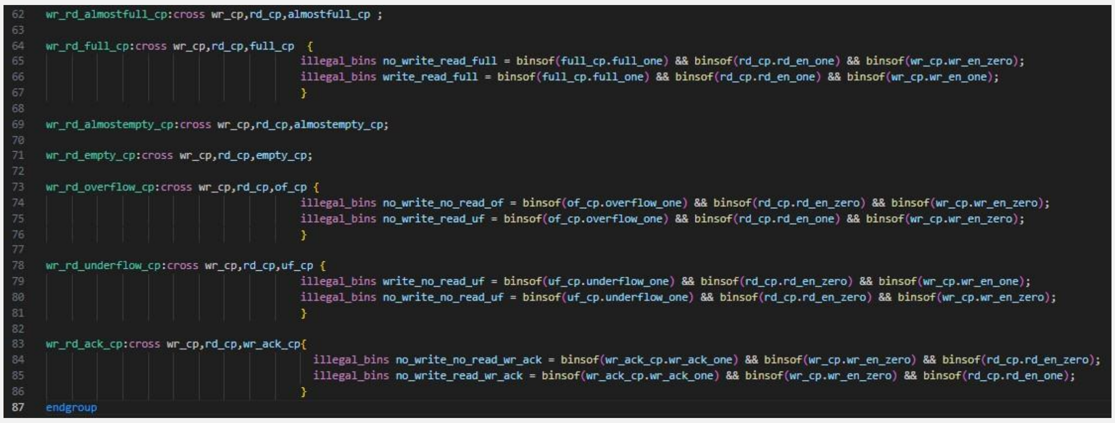
7. Coverage Collector :



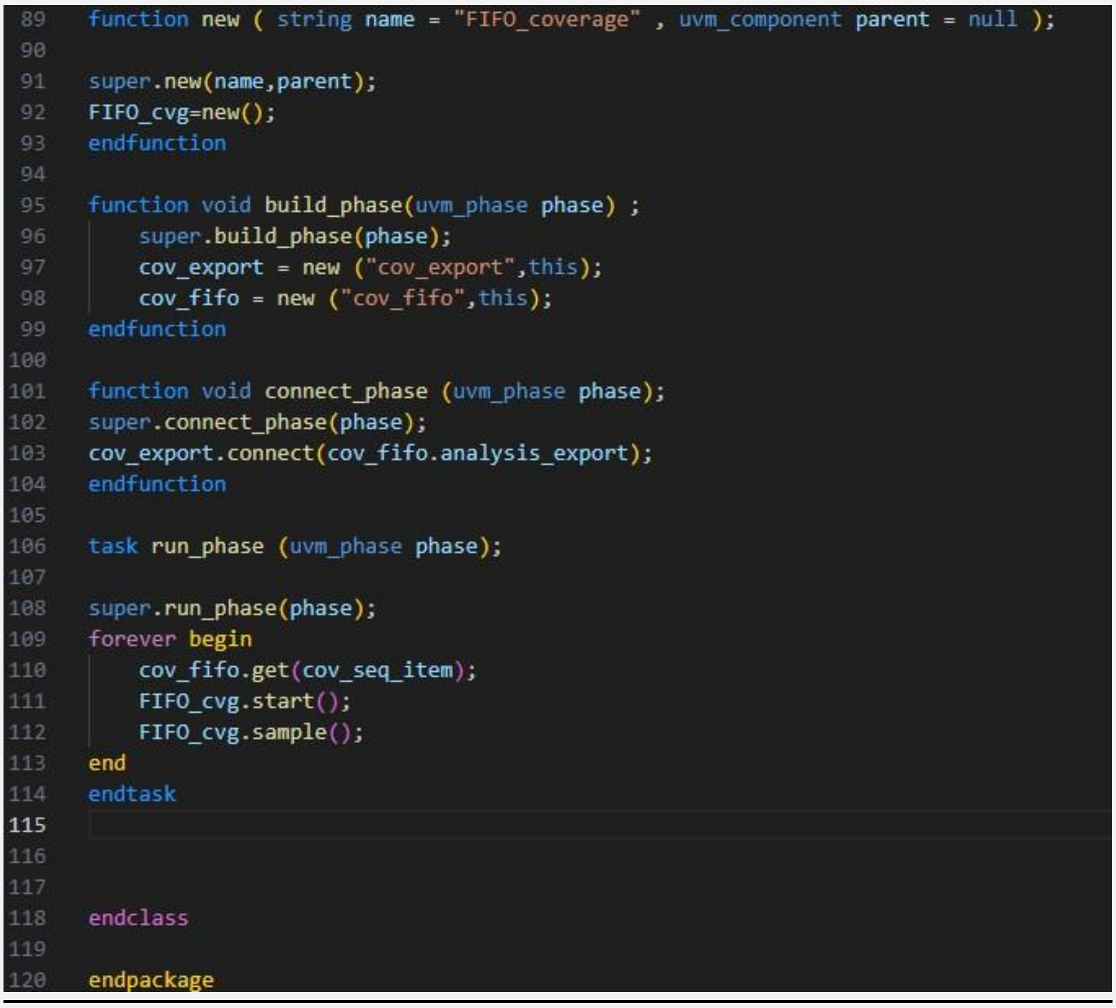
```
package FIFO_coverage_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;

class FIFO_coverage extends uvm_component;
`uvm_component_utils(FIFO_coverage)

FIFO_sequence_item cov_seq_item;
uvm_analysis_export #(FIFO_sequence_item) cov_export;
uvm_tlm_analysis_fifo #(FIFO_sequence_item) cov_fifo;


covergroup FIFO_cvg ;

wr_cp:coverpoint cov_seq_item.wr_en {
                            bins wr_en_one = {1};
                            bins wr_en_zero = {0};
                            }

rd_cp:coverpoint cov_seq_item.rd_en {
                            bins rd_en_one = {1};
                            bins rd_en_zero = {0};
                            }

almostfull_cp: coverpoint cov_seq_item.almostfull {
                                bins almostfull_one = {1};
                                bins almostfull_zero = {0};
}

```



```
full_cp: coverpoint cov_seq_item.full {
                    bins full_one = {1};
                    bins full_zero = {0};
}

almostempty_cp: coverpoint cov_seq_item.almostempty {
                                bins almostempty_one = {1};
                                bins almostempty_zero = {0};
}

empty_cp:coverpoint cov_seq_item.empty {
                    bins empty_one = {1};
                    bins empty_zero = {0};
}

of_cp: coverpoint cov_seq_item.overflow {
                            bins overflow_one = {1};
                            bins overflow_zero = {0};
}

uf_cp: coverpoint cov_seq_item.underflow {
                            bins underflow_one = {1};
                            bins underflow_zero = {0};
}

wr_ack_cp:coverpoint cov_seq_item.wr_ack {
                            bins wr_ack_one = {1};
                            bins wr_ack_zero = {0};

}

```

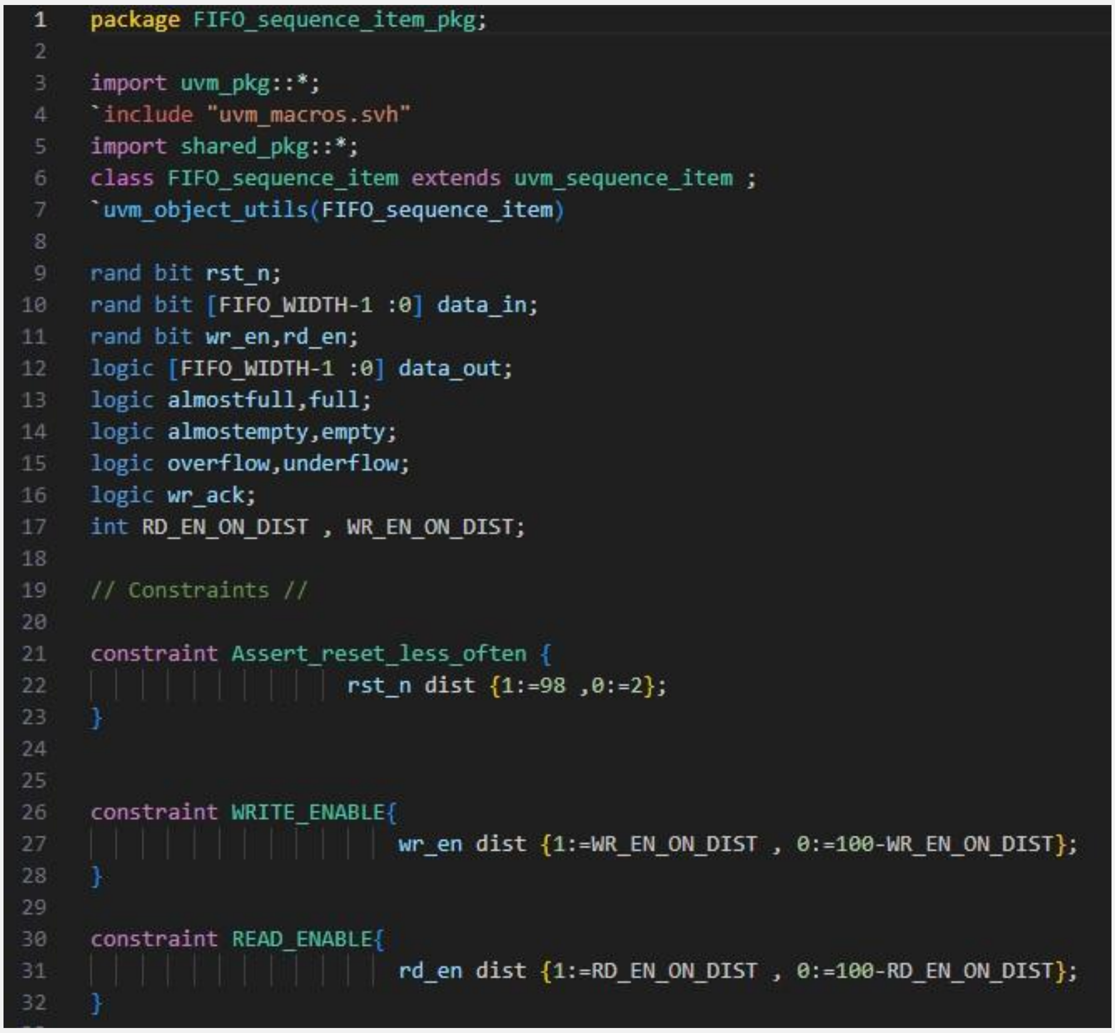


```
wr_rd_almostfull_cp:cross wr_cp,rd_cp,almostfull_cp ;

wr_rd_full_cp:cross wr_cp,rd_cp,full_cp  {
                                        illegal_bins no_write_read_full = binsof(full_cp.full_one) && binsof(rd_cp.rd_en_one) && binsof(wr_cp.wr_en_zero);
                                        illegal_bins write_read_full = binsof(full_cp.full_one) && binsof(rd_cp.rd_en_one) && binsof(wr_cp.wr_en_one);
                                        }

wr_rd_almostempty_cp:cross wr_cp,rd_cp,almostempty_cp;

wr_rd_empty_cp:cross wr_cp,rd_cp,empty_cp;

wr_rd_overflow_cp:cross wr_cp,rd_cp,of_cp {
                                        illegal_bins no_write_no_read_of = binsof(of_cp.overflow_one) && binsof(rd_cp.rd_en_zero) && binsof(wr_cp.wr_en_zero);
                                        illegal_bins no_write_read_uf = binsof(of_cp.overflow_one) && binsof(rd_cp.rd_en_one) && binsof(wr_cp.wr_en_zero);
                                        }

wr_rd_underflow_cp:cross wr_cp,rd_cp,uf_cp {
                                        illegal_bins write_no_read_uf = binsof(uf_cp.underflow_one) && binsof(rd_cp.rd_en_zero) && binsof(wr_cp.wr_en_one);
                                        illegal_bins no_write_no_read_uf = binsof(uf_cp.underflow_one) && binsof(rd_cp.rd_en_zero) && binsof(wr_cp.wr_en_zero);
                                        }

wr_rd_ack_cp:cross wr_cp,rd_cp,wr_ack_cp{
                                          illegal_bins no_write_no_read_wr_ack = binsof(wr_ack_cp.wr_ack_one) && binsof(wr_cp.wr_en_zero) && binsof(rd_cp.rd_en_zero);
                                          illegal_bins no_write_read_wr_ack = binsof(wr_ack_cp.wr_ack_one) && binsof(wr_cp.wr_en_zero) && binsof(rd_cp.rd_en_one);
                                        }
endgroup
```

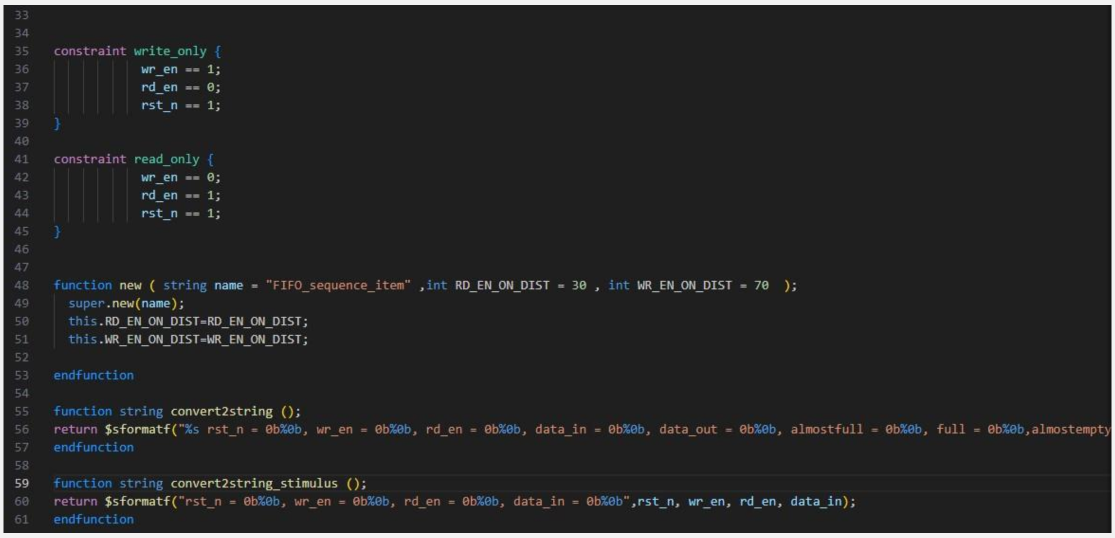


```
function new ( string name = "FIFO_coverage" , uvm_component parent = null );

super.new(name,parent);
FIFO_cvg=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(cov_seq_item);
    FIFO_cvg.start();
    FIFO_cvg.sample();
end
endtask



endclass

endpackage
```

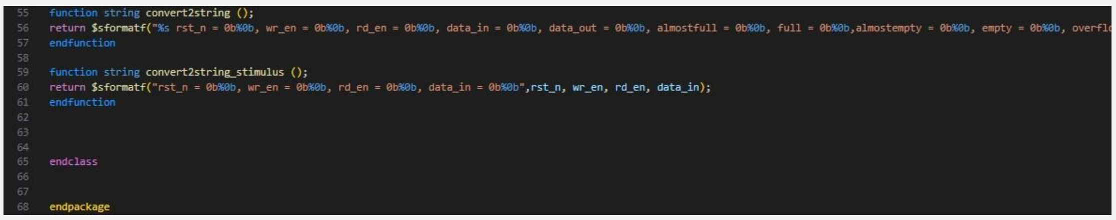
8. Sequence Item :



```systemverilog
package FIFO_sequence_item_pkg;

import uvm_pkg::*;
`include "uvm_macros.svh"
import shared_pkg::*;
class FIFO_sequence_item extends uvm_sequence_item ;
`uvm_object_utils(FIFO_sequence_item)

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

// Constraints //

constraint Assert_reset_less_often {
                        rst_n dist {1:=98 ,0:=2};
}


constraint WRITE_ENABLE{
                            wr_en dist {1:=WR_EN_ON_DIST , 0:=100-WR_EN_ON_DIST};
}

constraint READ_ENABLE{
                            rd_en dist {1:=RD_EN_ON_DIST , 0:=100-RD_EN_ON_DIST};
}
```

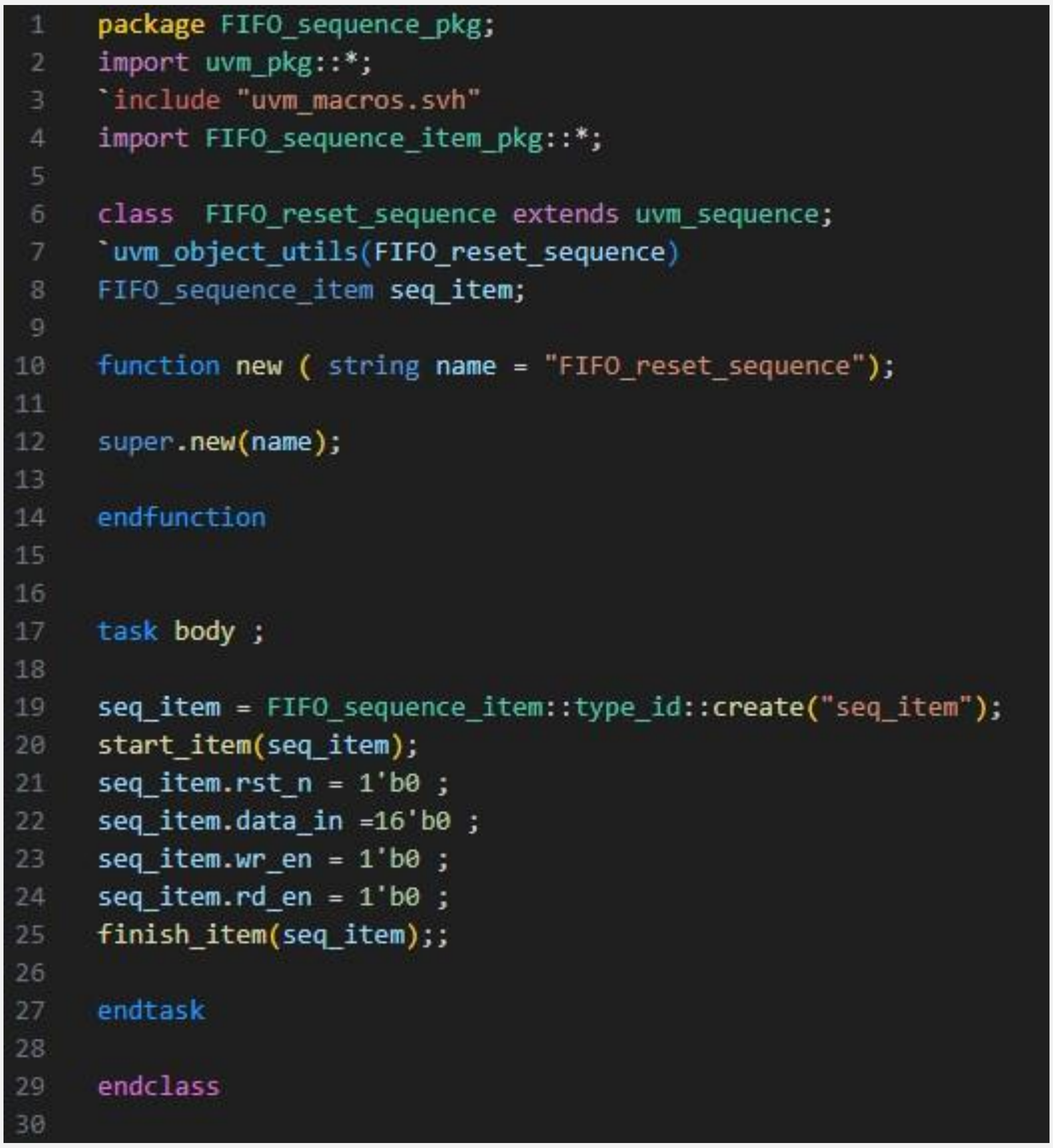


```systemverilog


constraint write_only {
            wr_en == 1;
            rd_en == 0;
            rst_n == 1;
}

constraint read_only {
            wr_en == 0;
            rd_en == 1;
            rst_n == 1;
}


function new ( string name = "FIFO_sequence_item" ,int RD_EN_ON_DIST = 30 , int WR_EN_ON_DIST = 70  );
  super.new(name);
  this.RD_EN_ON_DIST=RD_EN_ON_DIST;
  this.WR_EN_ON_DIST=WR_EN_ON_DIST;

endfunction

function string convert2string ();
return $sformatf("%s rst_n = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b, data_in = 0b%0b, data_out = 0b%0b, almostfull = 0b%0b, full = 0b%0b,almostempty
endfunction

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

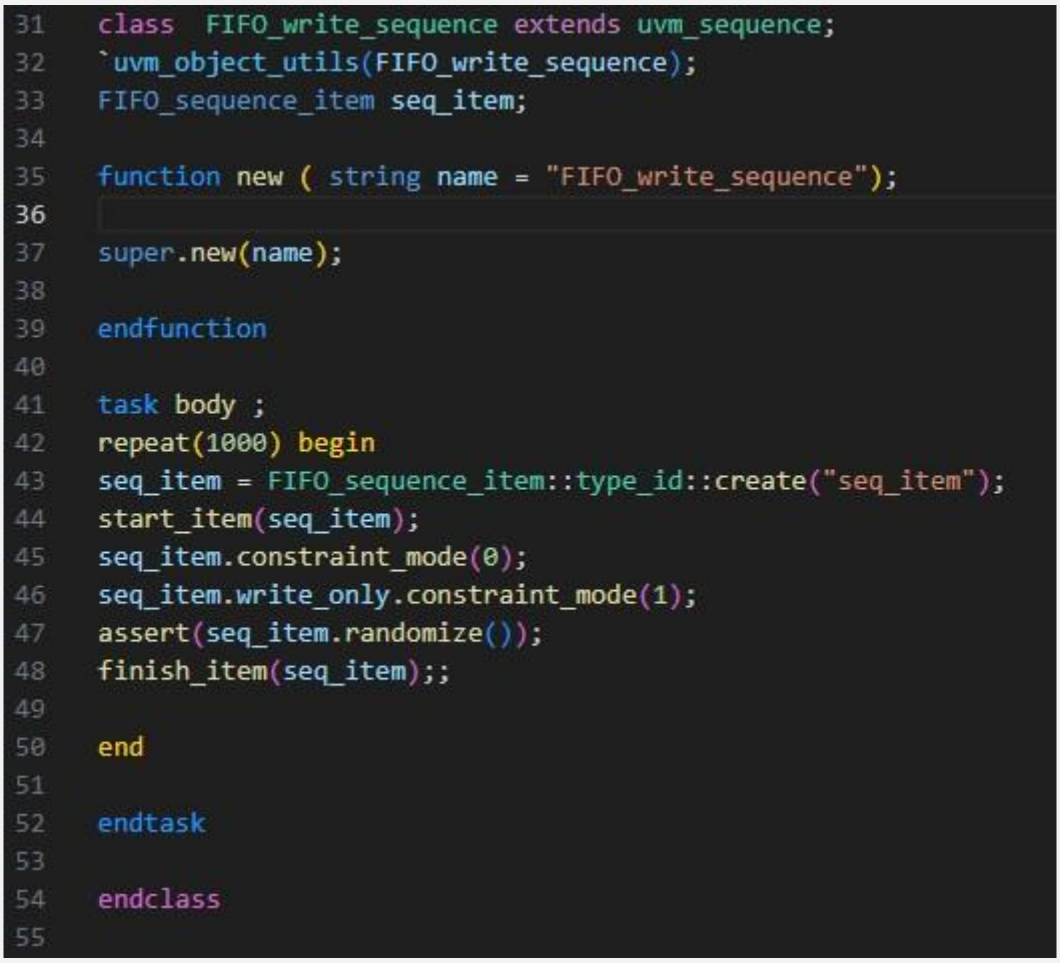


```
function string convert2string ();
return $sformatf("%s rst_n = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b, data_in = 0b%0b, data_out = 0b%0b, almostfull = 0b%0b, full = 0b%0b,almostempty = 0b%0b, empty = 0b%0b, overfl
endfunction

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



endclass


endpackage
```

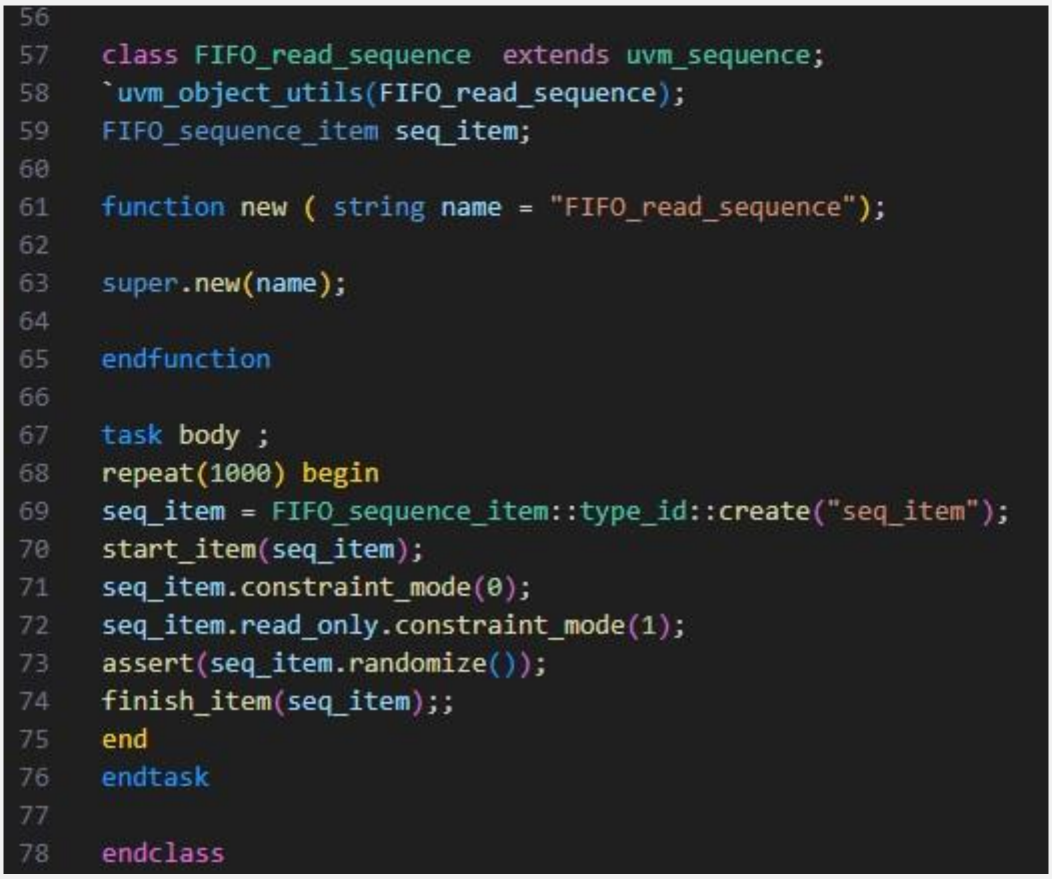
9. Sequence:



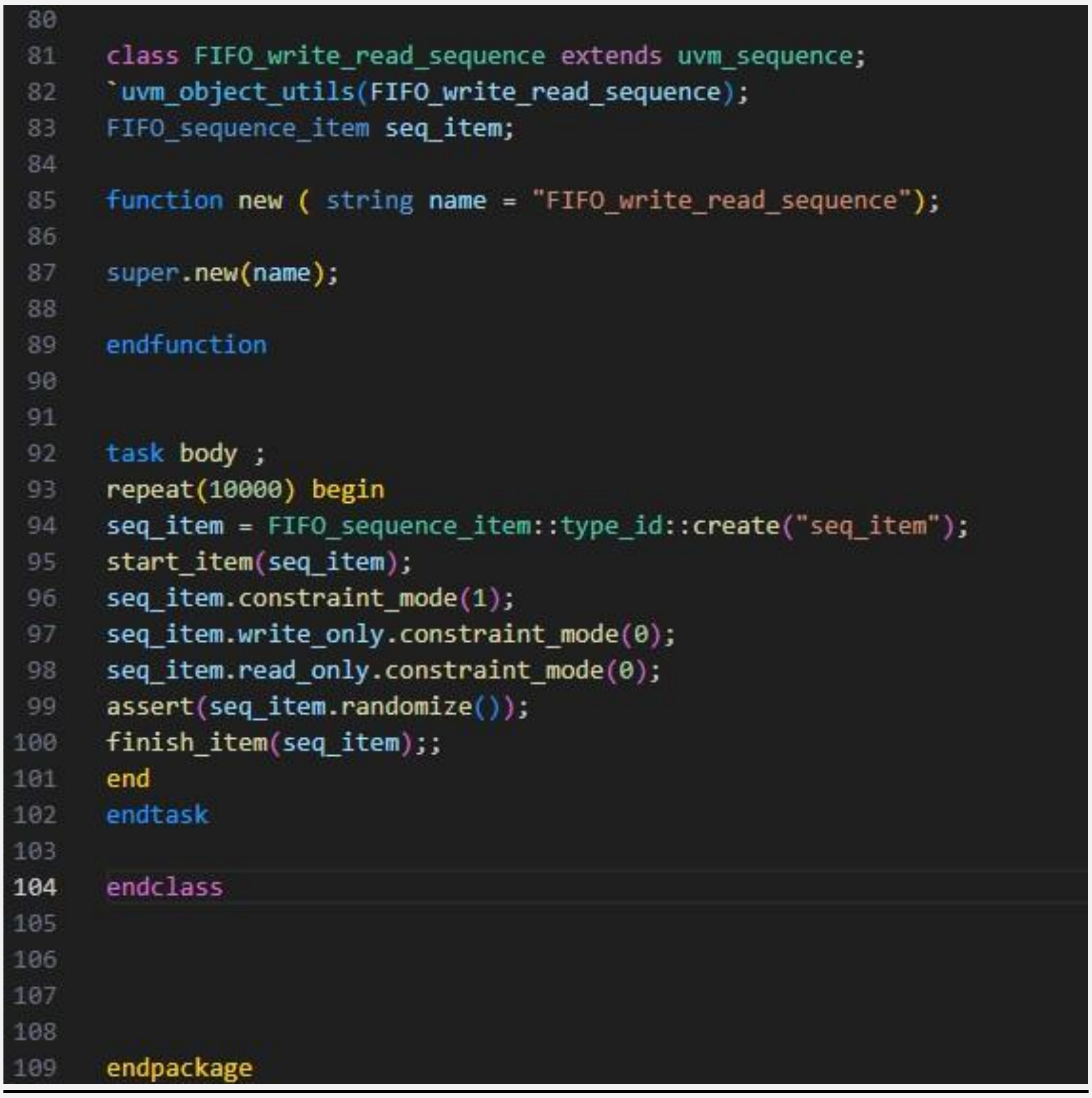
```
package FIFO_sequence_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;

class  FIFO_reset_sequence extends uvm_sequence;
`uvm_object_utils(FIFO_reset_sequence)
FIFO_sequence_item seq_item;

function new ( string name = "FIFO_reset_sequence");

super.new(name);

endfunction


task body ;

seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.rst_n = 1'b0 ;
seq_item.data_in =16'b0 ;
seq_item.wr_en = 1'b0 ;
seq_item.rd_en = 1'b0 ;
finish_item(seq_item);;

endtask

endclass

```



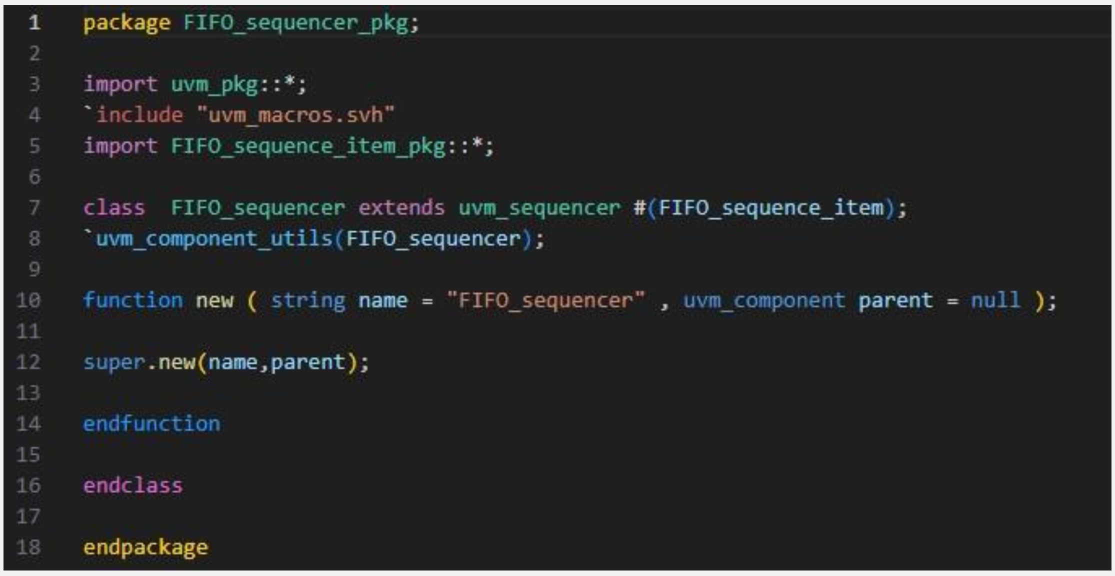
```
class  FIFO_write_sequence extends uvm_sequence;
`uvm_object_utils(FIFO_write_sequence);
FIFO_sequence_item seq_item;

function new ( string name = "FIFO_write_sequence");

super.new(name);

endfunction

task body ;
repeat(1000) begin
seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.constraint_mode(0);
seq_item.write_only.constraint_mode(1);
assert(seq_item.randomize());
finish_item(seq_item);;

end

endtask

endclass

```



```

class FIFO_read_sequence  extends uvm_sequence;
`uvm_object_utils(FIFO_read_sequence);
FIFO_sequence_item seq_item;

function new ( string name = "FIFO_read_sequence");

super.new(name);

endfunction

task body ;
repeat(1000) begin
seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.constraint_mode(0);
seq_item.read_only.constraint_mode(1);
assert(seq_item.randomize());
finish_item(seq_item);;
end
endtask

endclass
```

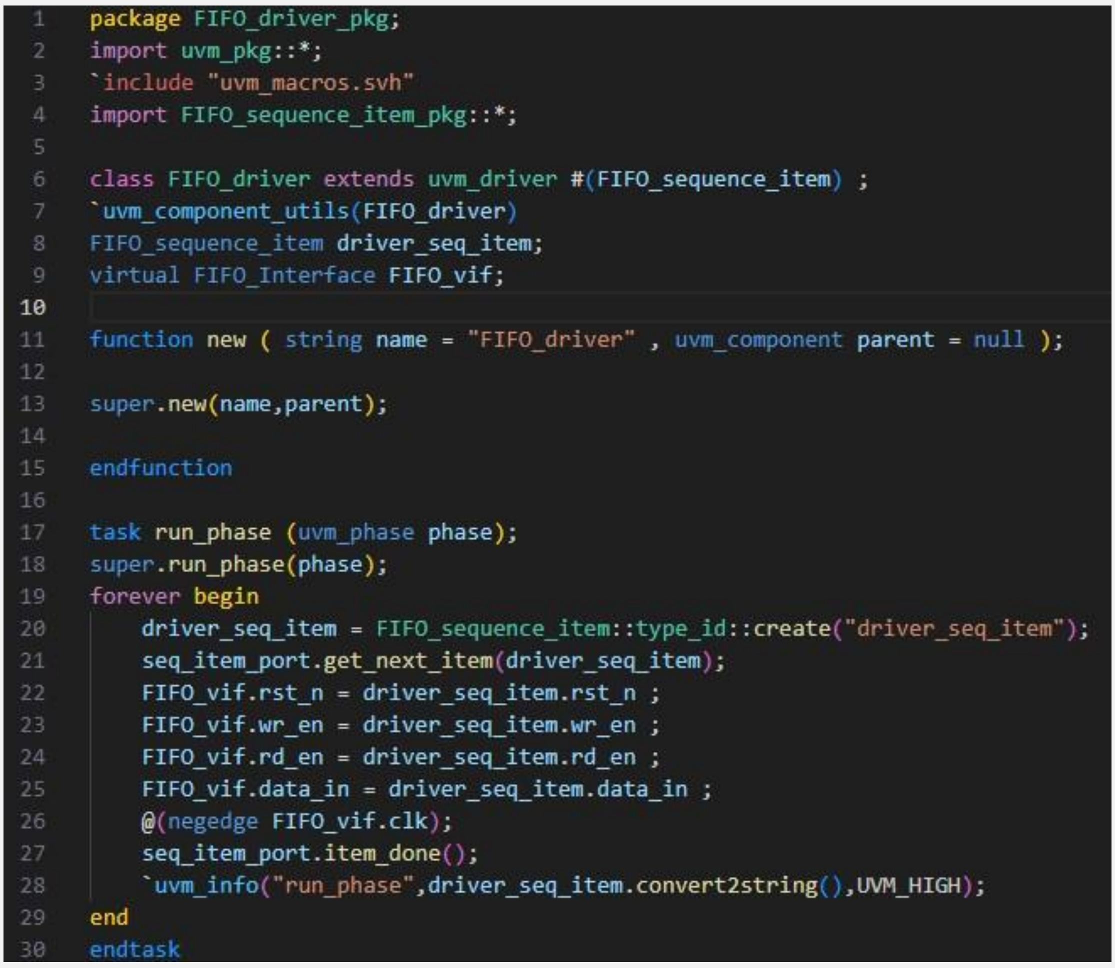

```

class FIFO_write_read_sequence extends uvm_sequence;
`uvm_object_utils(FIFO_write_read_sequence);
FIFO_sequence_item seq_item;

function new ( string name = "FIFO_write_read_sequence");

super.new(name);

endfunction


task body ;
repeat(10000) begin
seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.constraint_mode(1);
seq_item.write_only.constraint_mode(0);
seq_item.read_only.constraint_mode(0);
assert(seq_item.randomize());
finish_item(seq_item);;
end
endtask

endclass




endpackage
```

10.Sequencer:

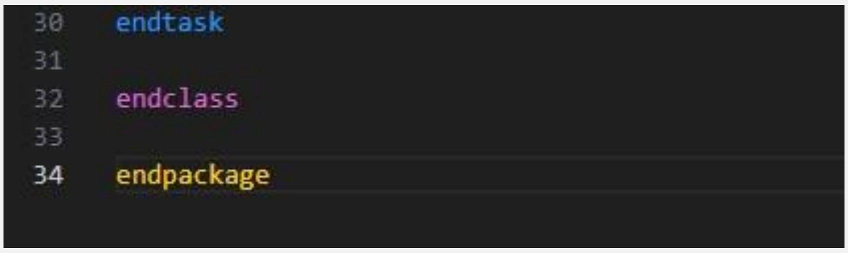


```
package FIFO_sequencer_pkg;

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

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

function new ( string name = "FIFO_sequencer" , uvm_component parent = null );

super.new(name,parent);

endfunction

endclass

endpackage
```

11.Driver:

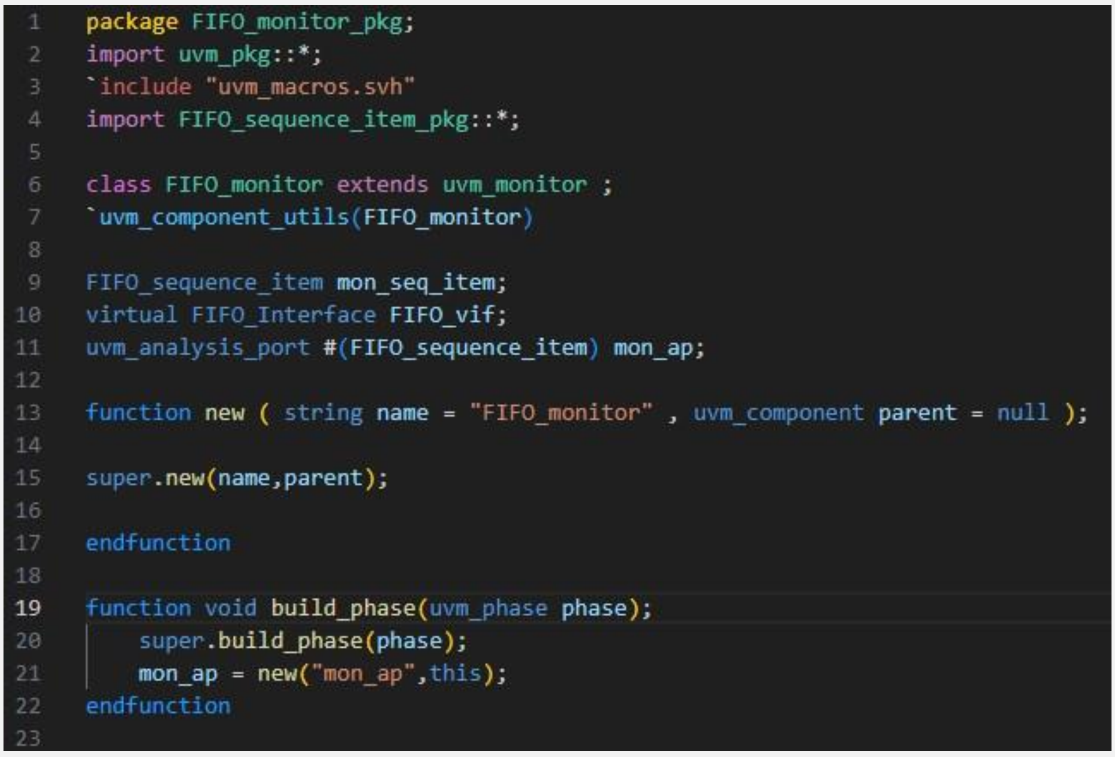


```
package FIFO_driver_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;

class FIFO_driver extends uvm_driver #(FIFO_sequence_item) ;
`uvm_component_utils(FIFO_driver)
FIFO_sequence_item driver_seq_item;
virtual FIFO_Interface FIFO_vif;

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
    driver_seq_item = FIFO_sequence_item::type_id::create("driver_seq_item");
    seq_item_port.get_next_item(driver_seq_item);
    FIFO_vif.rst_n = driver_seq_item.rst_n ;
    FIFO_vif.wr_en = driver_seq_item.wr_en ;
    FIFO_vif.rd_en = driver_seq_item.rd_en ;
    FIFO_vif.data_in = driver_seq_item.data_in ;
    @(negedge FIFO_vif.clk);
    seq_item_port.item_done();
    `uvm_info("run_phase",driver_seq_item.convert2string(),UVM_HIGH);
end
endtask
```

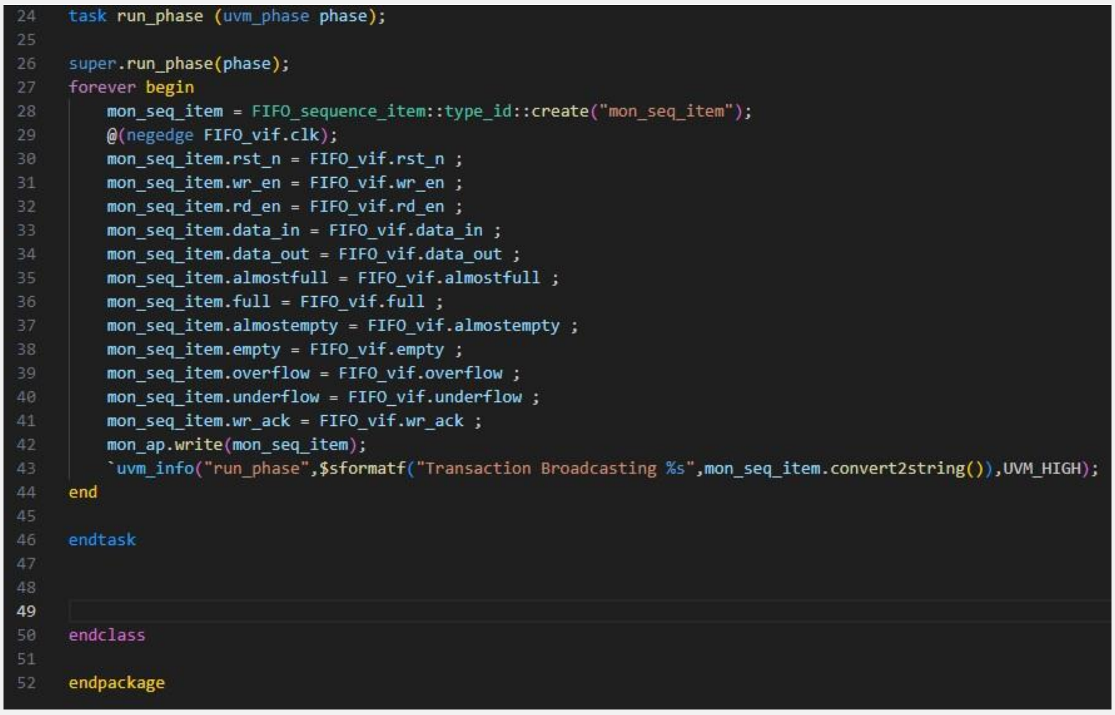


```
endtask

endclass

endpackage
```

12.Monitor:

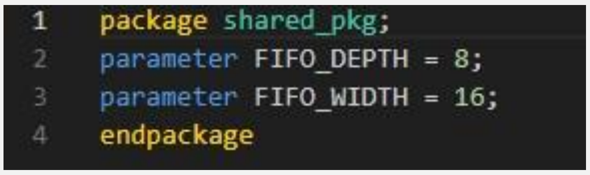


```
package FIFO_monitor_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_sequence_item_pkg::*;

class FIFO_monitor extends uvm_monitor ;
`uvm_component_utils(FIFO_monitor)

FIFO_sequence_item mon_seq_item;
virtual FIFO_Interface FIFO_vif;
uvm_analysis_port #(FIFO_sequence_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

```



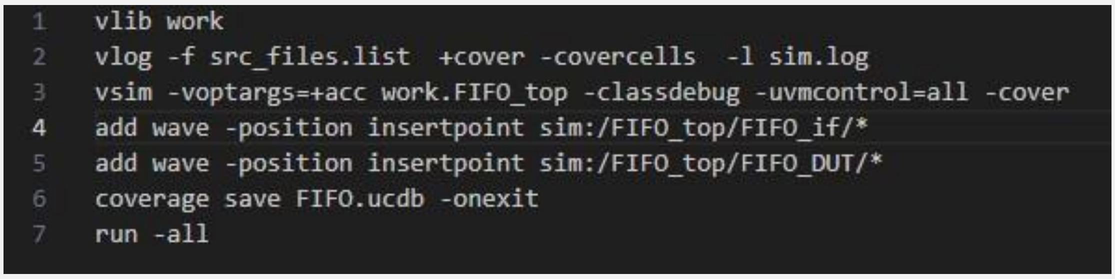
```
task run_phase (uvm_phase phase);

super.run_phase(phase);
forever begin
    mon_seq_item = FIFO_sequence_item::type_id::create("mon_seq_item");
    @(negedge FIFO_vif.clk);
    mon_seq_item.rst_n = FIFO_vif.rst_n ;
    mon_seq_item.wr_en = FIFO_vif.wr_en ;
    mon_seq_item.rd_en = FIFO_vif.rd_en ;
    mon_seq_item.data_in = FIFO_vif.data_in ;
    mon_seq_item.data_out = FIFO_vif.data_out ;
    mon_seq_item.almostfull = FIFO_vif.almostfull ;
    mon_seq_item.full = FIFO_vif.full ;
    mon_seq_item.almostempty = FIFO_vif.almostempty ;
    mon_seq_item.empty = FIFO_vif.empty ;
    mon_seq_item.overflow = FIFO_vif.overflow ;
    mon_seq_item.underflow = FIFO_vif.underflow ;
    mon_seq_item.wr_ack = FIFO_vif.wr_ack ;
    mon_ap.write(mon_seq_item);
    `uvm_info("run_phase",$sformatf("Transaction Broadcasting %s",mon_seq_item.convert2string()),UVM_HIGH);
end

endtask



endclass

endpackage
```

13.Shared package:



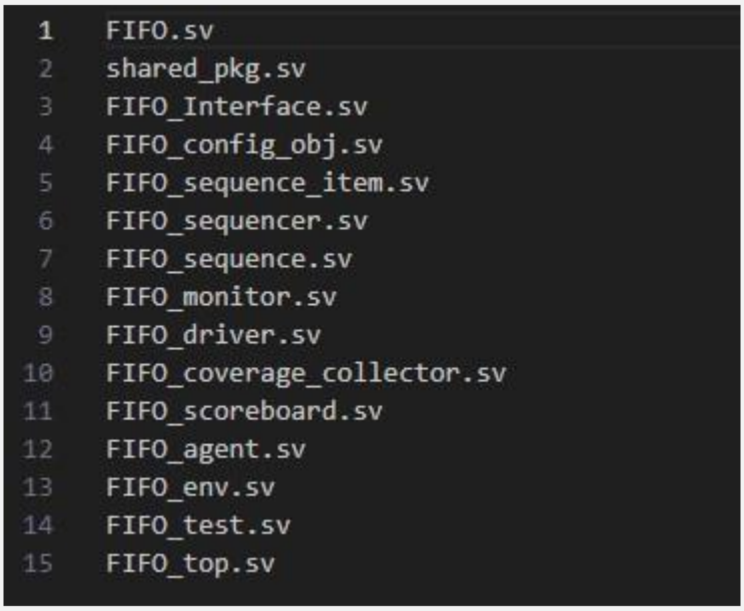
```
package shared_pkg;
parameter FIFO_DEPTH = 8;
parameter FIFO_WIDTH = 16;
endpackage
```

7- Do File:



```
vlib work
vlog -f src_files.list  +cover -covercells  -l sim.log
vsim -voptargs=+acc work.FIFO_top -classdebug -uvmcontrol=all -cover
add wave -position insertpoint sim:/FIFO_top/FIFO_if/*
add wave -position insertpoint sim:/FIFO_top/FIFO_DUT/*
coverage save FIFO.ucdb -onexit
run -all
```

src_files:

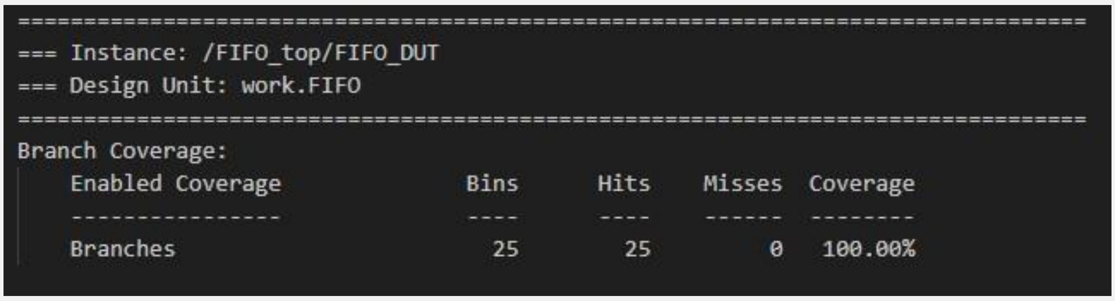


```
FIFO.sv
shared_pkg.sv
FIFO_Interface.sv
FIFO_config_obj.sv
FIFO_sequence_item.sv
FIFO_sequencer.sv
FIFO_sequence.sv
FIFO_monitor.sv
FIFO_driver.sv
FIFO_coverage_collector.sv
FIFO_scoreboard.sv
FIFO_agent.sv
FIFO_env.sv
FIFO_test.sv
FIFO_top.sv
```

8- Coverage Report:

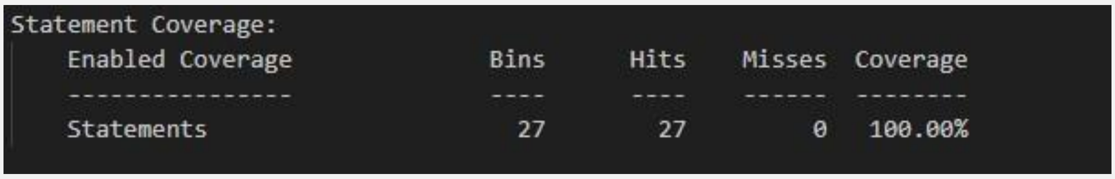
1. Code Coverage:

1- Branch Coverage:



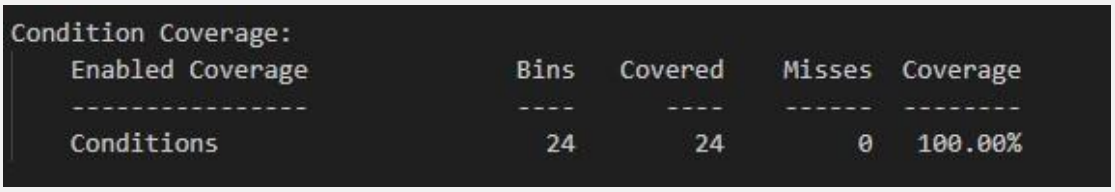
```
==================================================================================
=== Instance: /FIFO_top/FIFO_DUT
=== Design Unit: work.FIFO
==================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        25        25         0   100.00%
```

2- Statement Coverage:



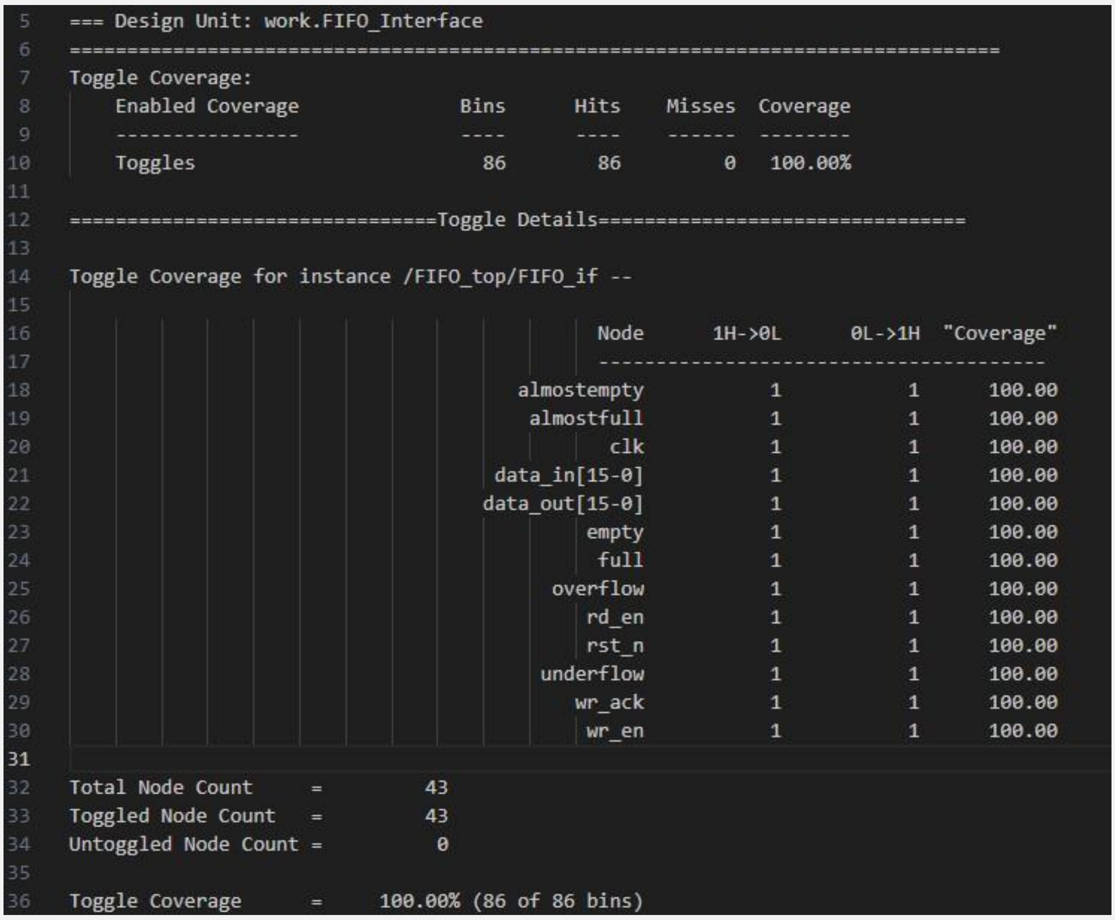
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      27        27         0   100.00%
```

3- Condition Coverage:

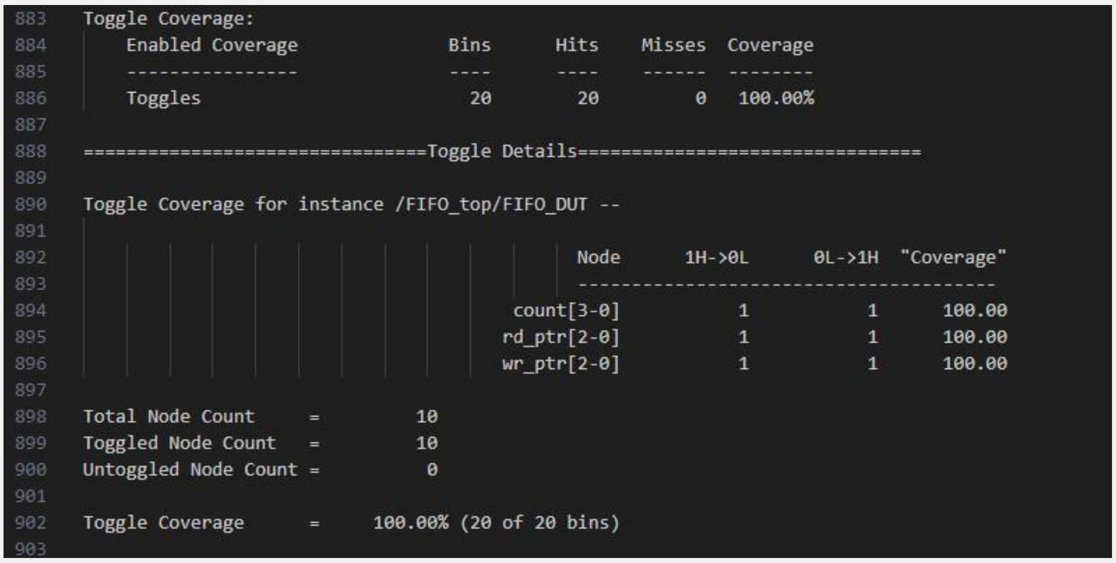


```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      24        24         0   100.00%
```

4- Toggle Coverage:

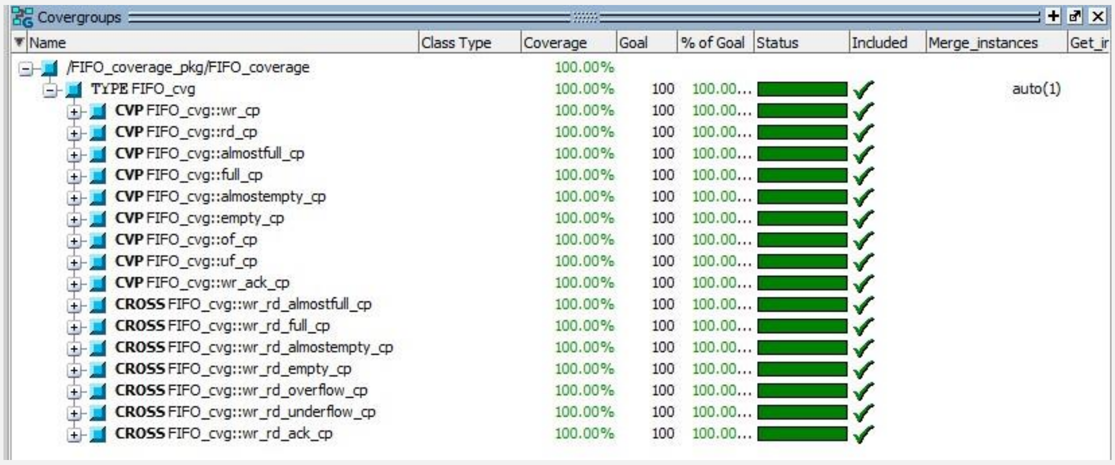


```
=== Design Unit: work.FIFO_Interface
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         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)
```



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

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

Toggle Coverage for instance /FIFO_top/FIFO_DUT --

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

Total Node Count     =         10
Toggled Node Count   =         10
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (20 of 20 bins)
```

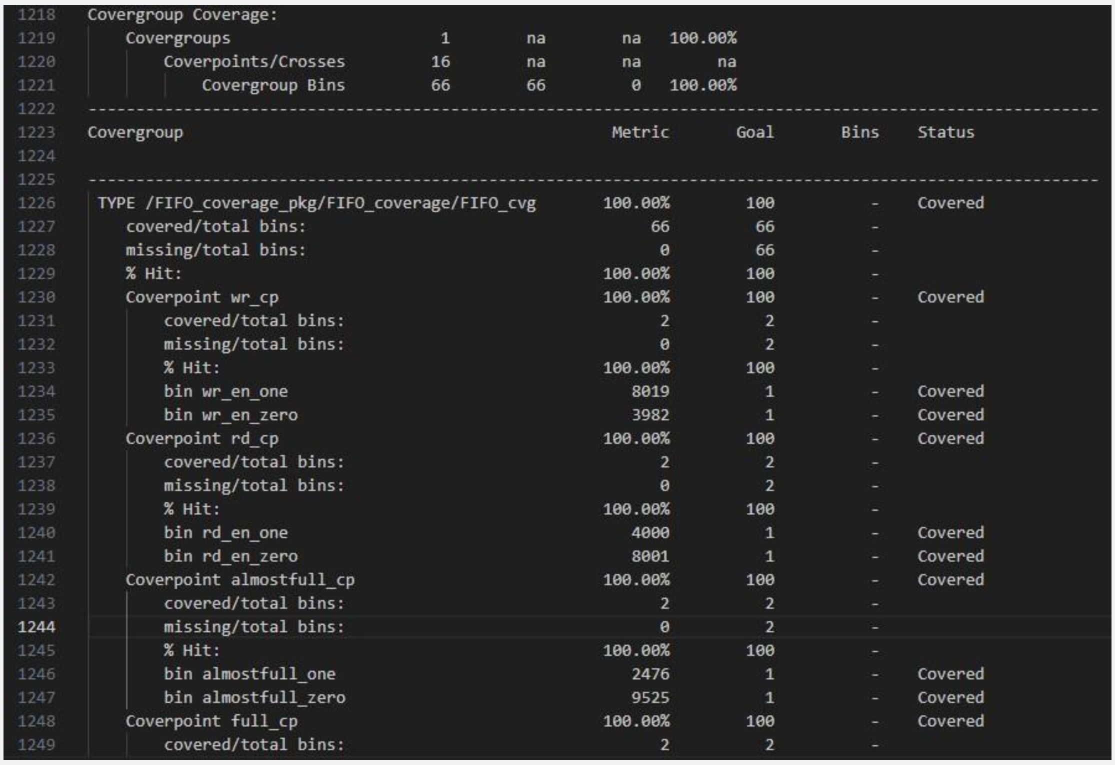
2. Functional Coverage:



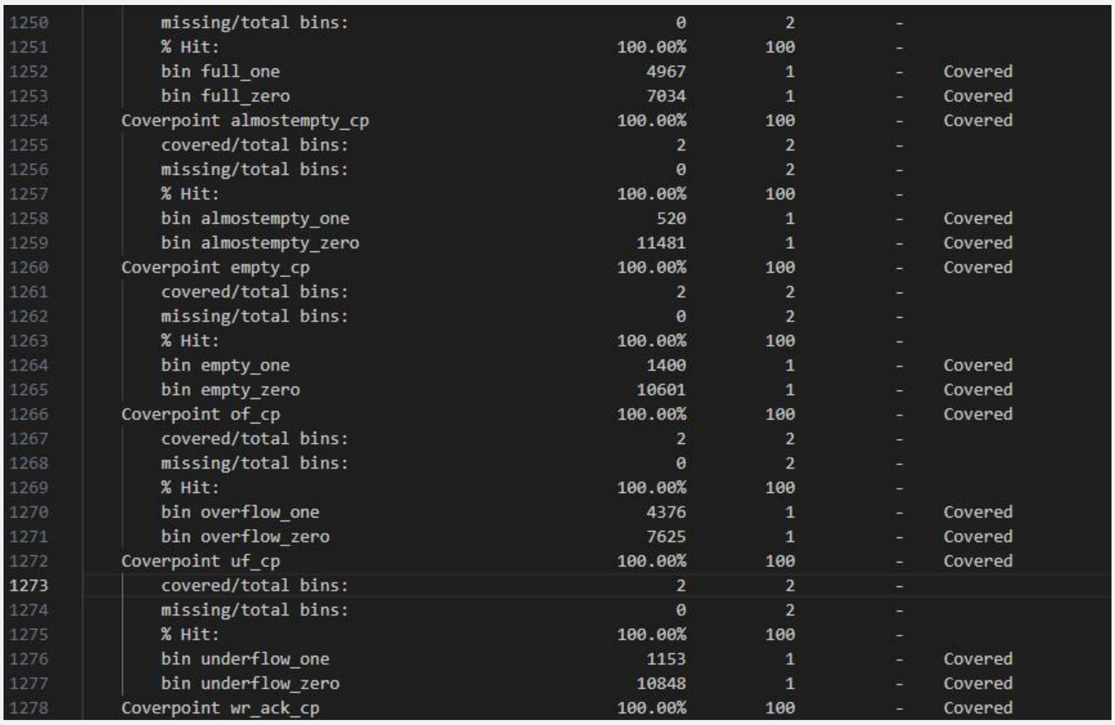
Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_ir
/FIFO_coverage_pkg/FIFO_coverage		100.00%						
TYPE FIFO_cvg		100.00%	100	100.00...	🟩	✓	auto(1)	
CVP FIFO_cvg::wr_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::rd_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::almostfull_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::full_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::almostempty_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::empty_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::of_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::uf_cp		100.00%	100	100.00...	🟩	✓		
CVP FIFO_cvg::wr_ack_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_almostfull_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_full_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_almostempty_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_empty_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_overflow_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_underflow_cp		100.00%	100	100.00...	🟩	✓		
CROSS FIFO_cvg::wr_rd_ack_cp		100.00%	100	100.00...	🟩	✓		

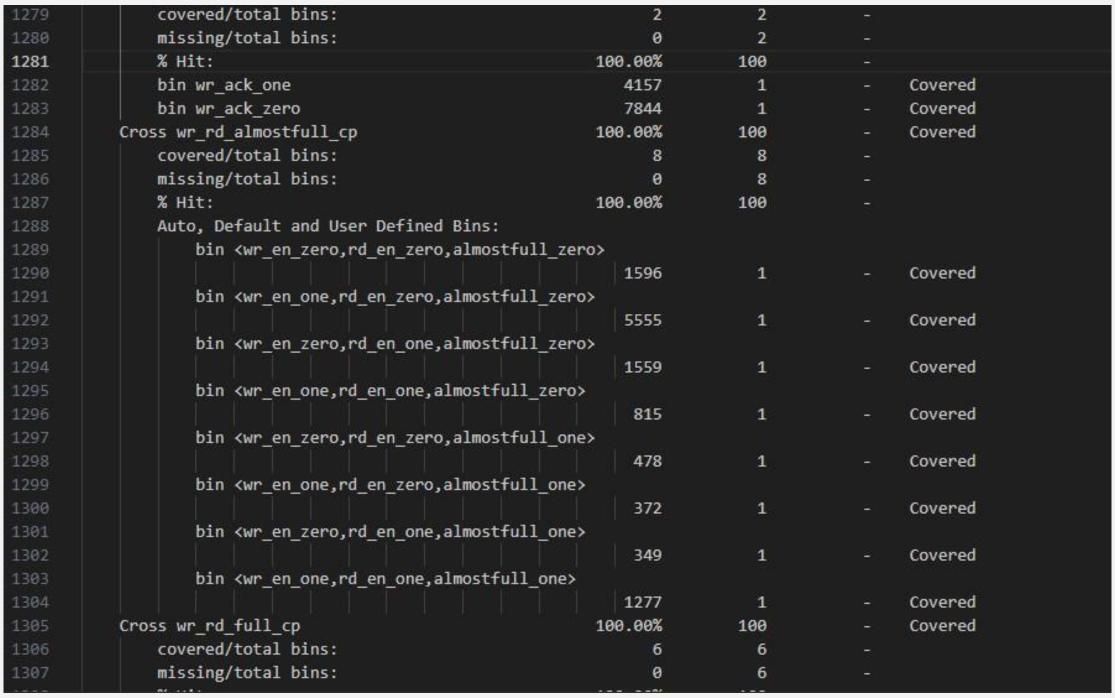
Details:



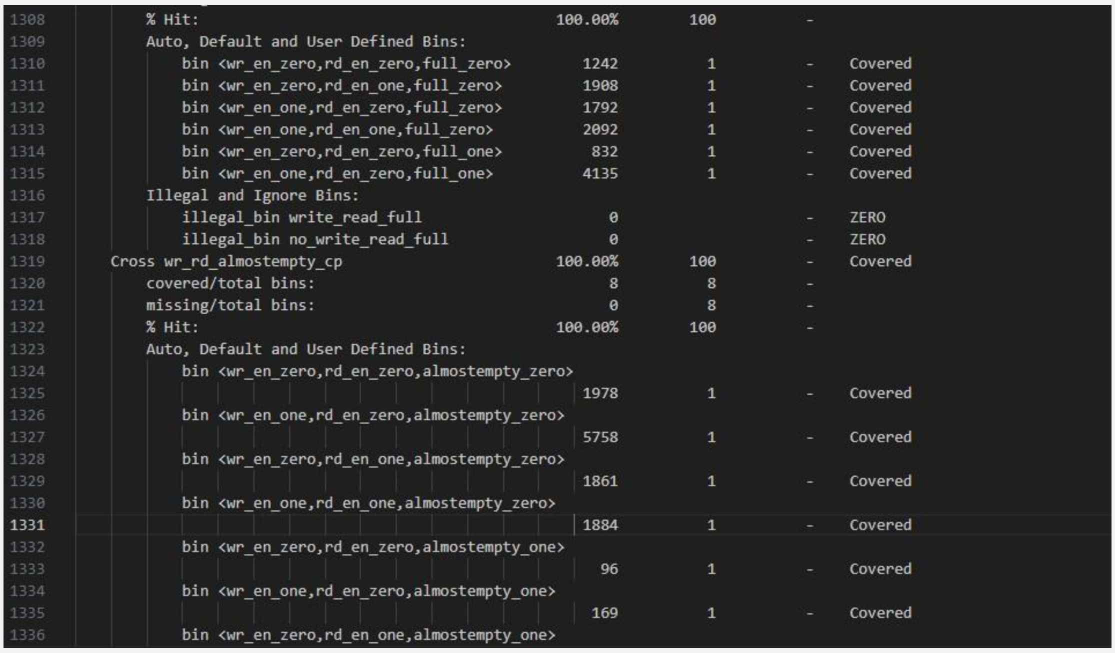
```
Covergroup Coverage:
    Covergroups                      1         na         na   100.00%
        Coverpoints/Crosses         16         na         na        na
            Covergroup Bins         66         66          0   100.00%
----------------------------------------------------------------------------------------------------------------
Covergroup                                             Metric        Goal       Bins    Status

----------------------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pkg/FIFO_coverage/FIFO_cvg       100.00%         100          -    Covered
    covered/total bins:                                    66          66          -
    missing/total bins:                                     0          66          -
    % Hit:                                            100.00%         100          -
    Coverpoint wr_cp                                  100.00%         100          -    Covered
        covered/total bins:                                 2           2          -
        missing/total bins:                                 0           2          -
        % Hit:                                        100.00%         100          -
        bin wr_en_one                                    8019           1          -    Covered
        bin wr_en_zero                                   3982           1          -    Covered
    Coverpoint rd_cp                                  100.00%         100          -    Covered
        covered/total bins:                                 2           2          -
        missing/total bins:                                 0           2          -
        % Hit:                                        100.00%         100          -
        bin rd_en_one                                    4000           1          -    Covered
        bin rd_en_zero                                   8001           1          -    Covered
    Coverpoint almostfull_cp                          100.00%         100          -    Covered
        covered/total bins:                                 2           2          -
        missing/total bins:                                 0           2          -
        % Hit:                                        100.00%         100          -
        bin almostfull_one                               2476           1          -    Covered
        bin almostfull_zero                              9525           1          -    Covered
    Coverpoint full_cp                                100.00%         100          -    Covered
        covered/total bins:                                 2           2          -
```



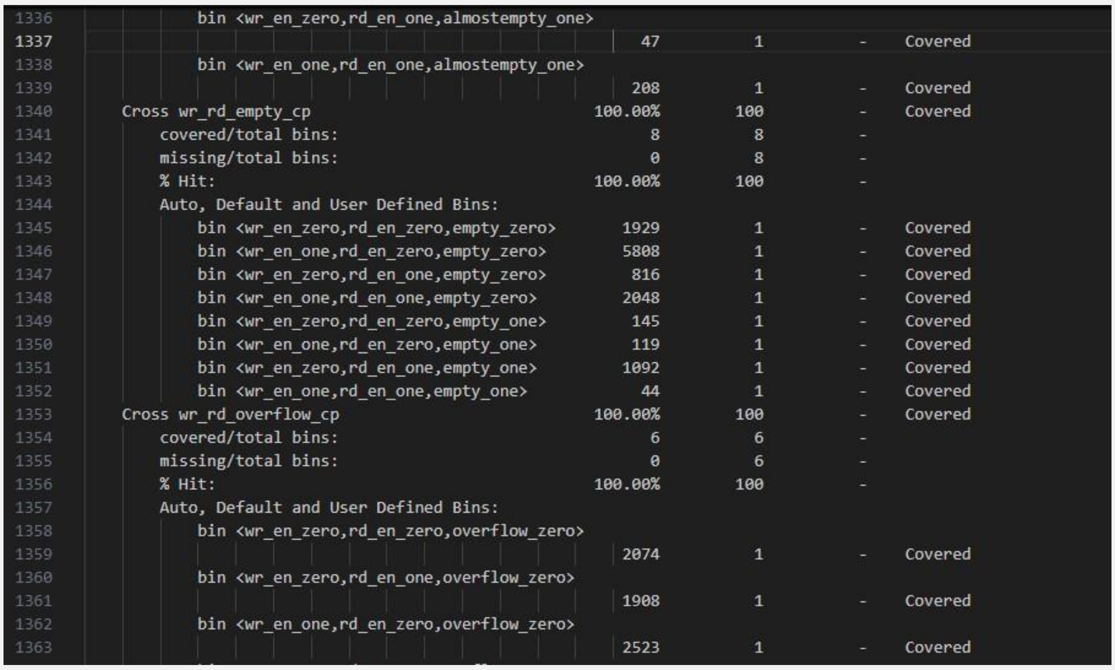
```
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin full_one                             4967          1          -    Covered
            bin full_zero                            7034          1          -    Covered
        Coverpoint almostempty_cp                 100.00%        100          -    Covered
            covered/total bins:                         2          2          -
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin almostempty_one                       520          1          -    Covered
            bin almostempty_zero                    11481          1          -    Covered
        Coverpoint empty_cp                       100.00%        100          -    Covered
            covered/total bins:                         2          2          -
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin empty_one                            1400          1          -    Covered
            bin empty_zero                          10601          1          -    Covered
        Coverpoint of_cp                          100.00%        100          -    Covered
            covered/total bins:                         2          2          -
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin overflow_one                         4376          1          -    Covered
            bin overflow_zero                        7625          1          -    Covered
        Coverpoint uf_cp                          100.00%        100          -    Covered
            covered/total bins:                         2          2          -
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin underflow_one                        1153          1          -    Covered
            bin underflow_zero                      10848          1          -    Covered
        Coverpoint wr_ack_cp                      100.00%        100          -    Covered
```



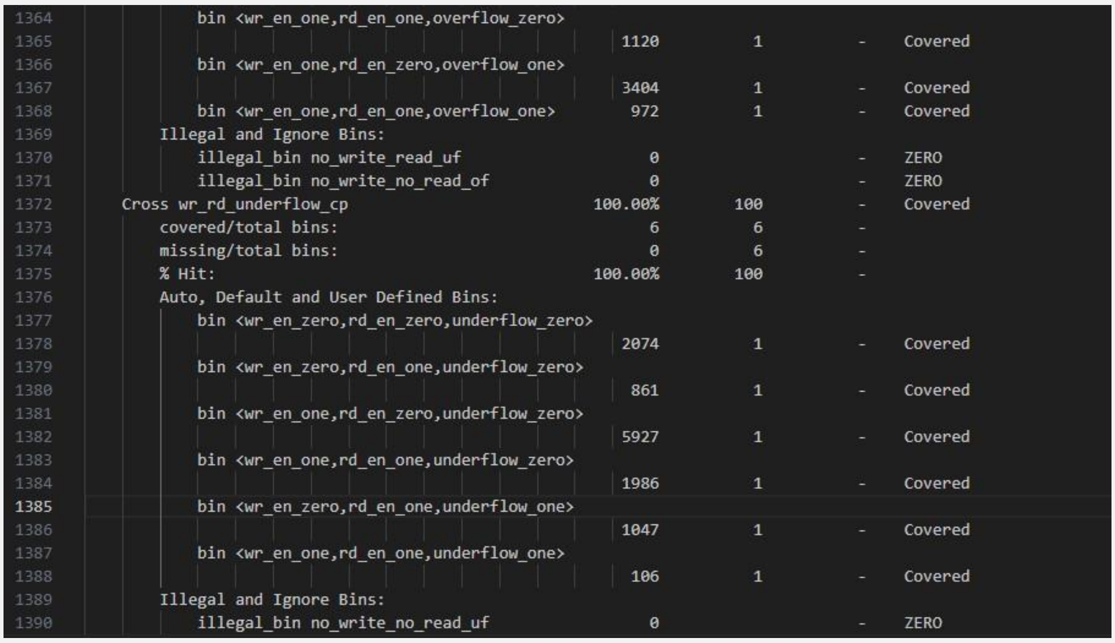
```
            covered/total bins:                         2          2          -
            missing/total bins:                         0          2          -
            % Hit:                                100.00%        100          -
            bin wr_ack_one                           4157          1          -    Covered
            bin wr_ack_zero                          7844          1          -    Covered
        Cross wr_rd_almostfull_cp                 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 <wr_en_zero,rd_en_zero,almostfull_zero>
                                                     1596          1          -    Covered
                bin <wr_en_one,rd_en_zero,almostfull_zero>
                                                     5555          1          -    Covered
                bin <wr_en_zero,rd_en_one,almostfull_zero>
                                                     1559          1          -    Covered
                bin <wr_en_one,rd_en_one,almostfull_zero>
                                                      815          1          -    Covered
                bin <wr_en_zero,rd_en_zero,almostfull_one>
                                                      478          1          -    Covered
                bin <wr_en_one,rd_en_zero,almostfull_one>
                                                      372          1          -    Covered
                bin <wr_en_zero,rd_en_one,almostfull_one>
                                                      349          1          -    Covered
                bin <wr_en_one,rd_en_one,almostfull_one>
                                                     1277          1          -    Covered
        Cross wr_rd_full_cp                       100.00%        100          -    Covered
            covered/total bins:                         6          6          -
            missing/total bins:                         0          6          -
```



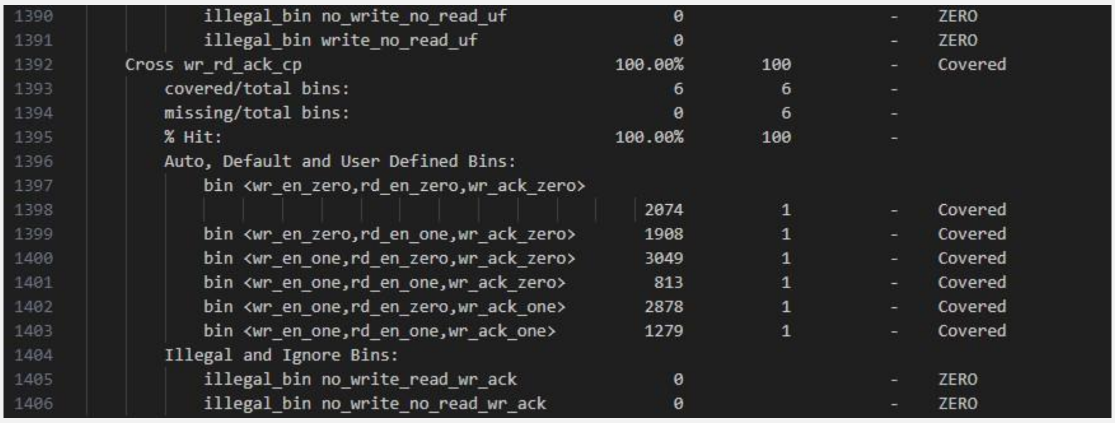
```
        % Hit:                                   100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <wr_en_zero,rd_en_zero,full_zero>        1242          1          -    Covered
            bin <wr_en_zero,rd_en_one,full_zero>         1908          1          -    Covered
            bin <wr_en_one,rd_en_zero,full_zero>         1792          1          -    Covered
            bin <wr_en_one,rd_en_one,full_zero>          2092          1          -    Covered
            bin <wr_en_zero,rd_en_zero,full_one>          832          1          -    Covered
            bin <wr_en_one,rd_en_zero,full_one>          4135          1          -    Covered
        Illegal and Ignore Bins:
            illegal_bin write_read_full                     0                     -    ZERO
            illegal_bin no_write_read_full                  0                     -    ZERO
    Cross wr_rd_almostempty_cp                    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 <wr_en_zero,rd_en_zero,almostempty_zero>
                                                     1978          1          -    Covered
            bin <wr_en_one,rd_en_zero,almostempty_zero>
                                                     5758          1          -    Covered
            bin <wr_en_zero,rd_en_one,almostempty_zero>
                                                     1861          1          -    Covered
            bin <wr_en_one,rd_en_one,almostempty_zero>
                                                     1884          1          -    Covered
            bin <wr_en_zero,rd_en_zero,almostempty_one>
                                                       96          1          -    Covered
            bin <wr_en_one,rd_en_zero,almostempty_one>
                                                      169          1          -    Covered
            bin <wr_en_zero,rd_en_one,almostempty_one>
```



```
            bin <wr_en_zero,rd_en_one,almostempty_one>
                                                       47          1          -    Covered
            bin <wr_en_one,rd_en_one,almostempty_one>
                                                      208          1          -    Covered
    Cross wr_rd_empty_cp                          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 <wr_en_zero,rd_en_zero,empty_zero>       1929          1          -    Covered
            bin <wr_en_one,rd_en_zero,empty_zero>        5808          1          -    Covered
            bin <wr_en_zero,rd_en_one,empty_zero>         816          1          -    Covered
            bin <wr_en_one,rd_en_one,empty_zero>         2048          1          -    Covered
            bin <wr_en_zero,rd_en_zero,empty_one>         145          1          -    Covered
            bin <wr_en_one,rd_en_zero,empty_one>          119          1          -    Covered
            bin <wr_en_zero,rd_en_one,empty_one>         1092          1          -    Covered
            bin <wr_en_one,rd_en_one,empty_one>            44          1          -    Covered
    Cross wr_rd_overflow_cp                       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 <wr_en_zero,rd_en_zero,overflow_zero>
                                                     2074          1          -    Covered
            bin <wr_en_zero,rd_en_one,overflow_zero>
                                                     1908          1          -    Covered
            bin <wr_en_one,rd_en_zero,overflow_zero>
                                                     2523          1          -    Covered
```

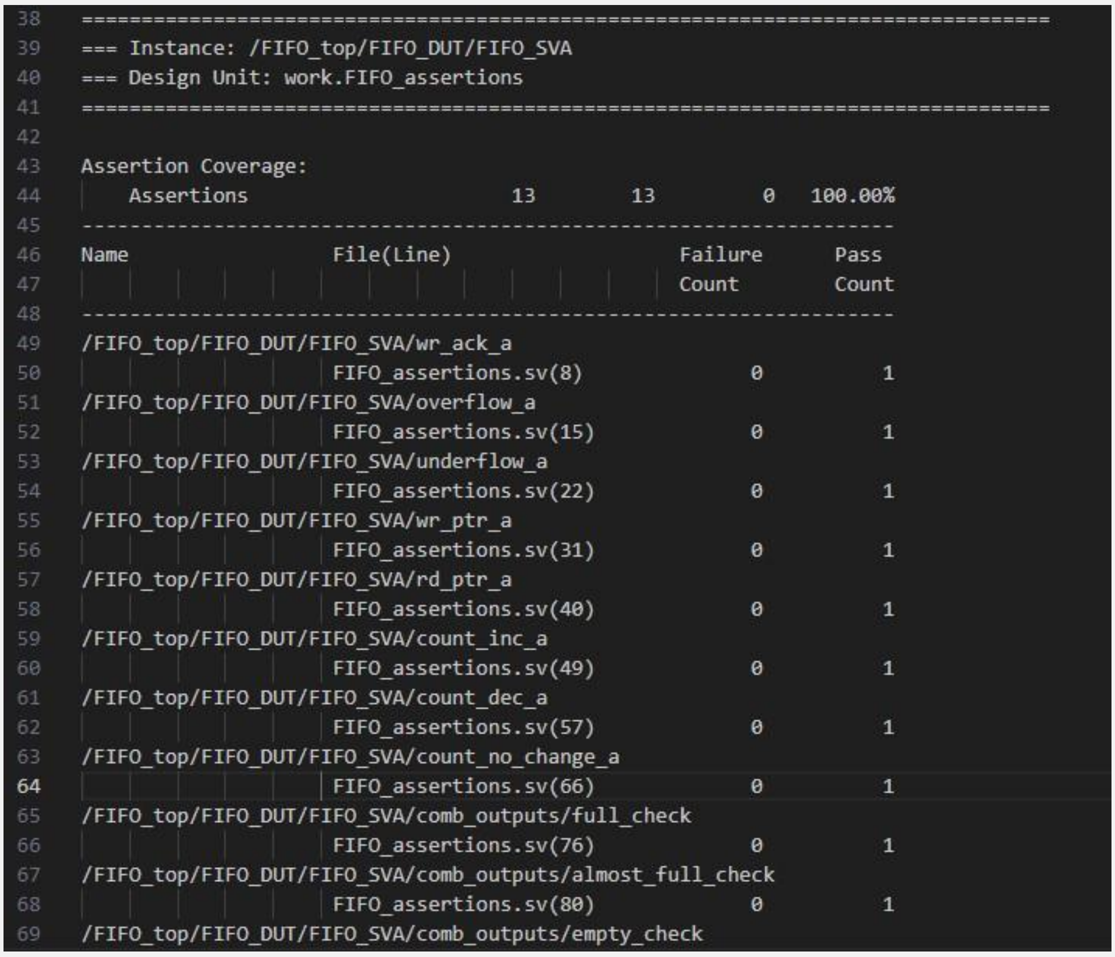


```
            bin <wr_en_one,rd_en_one,overflow_zero>
                                                1120          1          -     Covered
            bin <wr_en_one,rd_en_zero,overflow_one>
                                                3404          1          -     Covered
            bin <wr_en_one,rd_en_one,overflow_one>         972          1          -     Covered
        Illegal and Ignore Bins:
            illegal_bin no_write_read_uf                 0                     -     ZERO
            illegal_bin no_write_no_read_of              0                     -     ZERO
    Cross wr_rd_underflow_cp                         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 <wr_en_zero,rd_en_zero,underflow_zero>
                                                2074          1          -     Covered
            bin <wr_en_zero,rd_en_one,underflow_zero>
                                                 861          1          -     Covered
            bin <wr_en_one,rd_en_zero,underflow_zero>
                                                5927          1          -     Covered
            bin <wr_en_one,rd_en_one,underflow_zero>
                                                1986          1          -     Covered
            bin <wr_en_zero,rd_en_one,underflow_one>
                                                1047          1          -     Covered
            bin <wr_en_one,rd_en_one,underflow_one>
                                                 106          1          -     Covered
        Illegal and Ignore Bins:
            illegal_bin no_write_no_read_uf              0                     -     ZERO
```

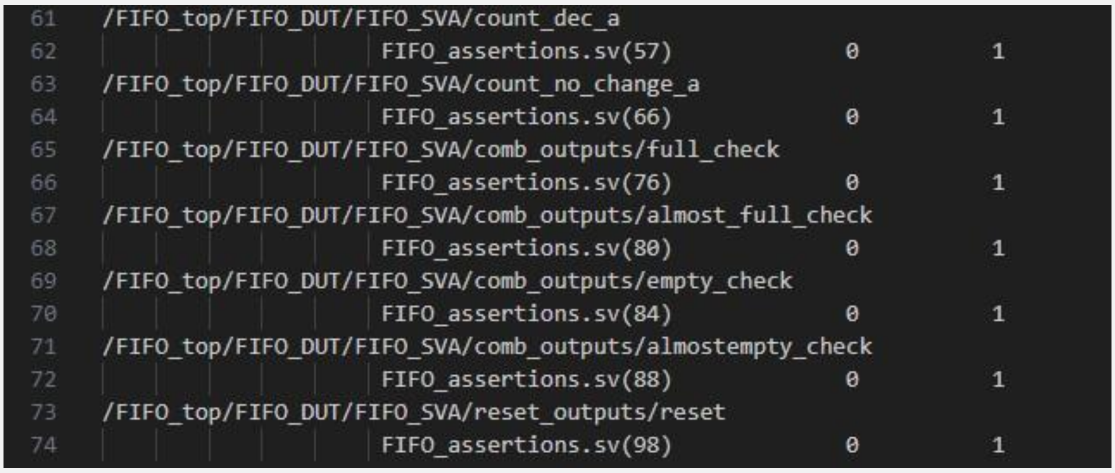


```
            illegal_bin no_write_no_read_uf              0                     -     ZERO
            illegal_bin write_no_read_uf                 0                     -     ZERO
    Cross wr_rd_ack_cp                               100.00%        100          -     Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                       100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <wr_en_zero,rd_en_zero,wr_ack_zero>
                                                2074          1          -     Covered
            bin <wr_en_zero,rd_en_one,wr_ack_zero>        1908          1          -     Covered
            bin <wr_en_one,rd_en_zero,wr_ack_zero>        3049          1          -     Covered
            bin <wr_en_one,rd_en_one,wr_ack_zero>          813          1          -     Covered
            bin <wr_en_one,rd_en_zero,wr_ack_one>         2878          1          -     Covered
            bin <wr_en_one,rd_en_one,wr_ack_one>          1279          1          -     Covered
        Illegal and Ignore Bins:
            illegal_bin no_write_read_wr_ack             0                     -     ZERO
            illegal_bin no_write_no_read_wr_ack          0                     -     ZERO
```

3. Sequential Domain Coverage(Assertions):

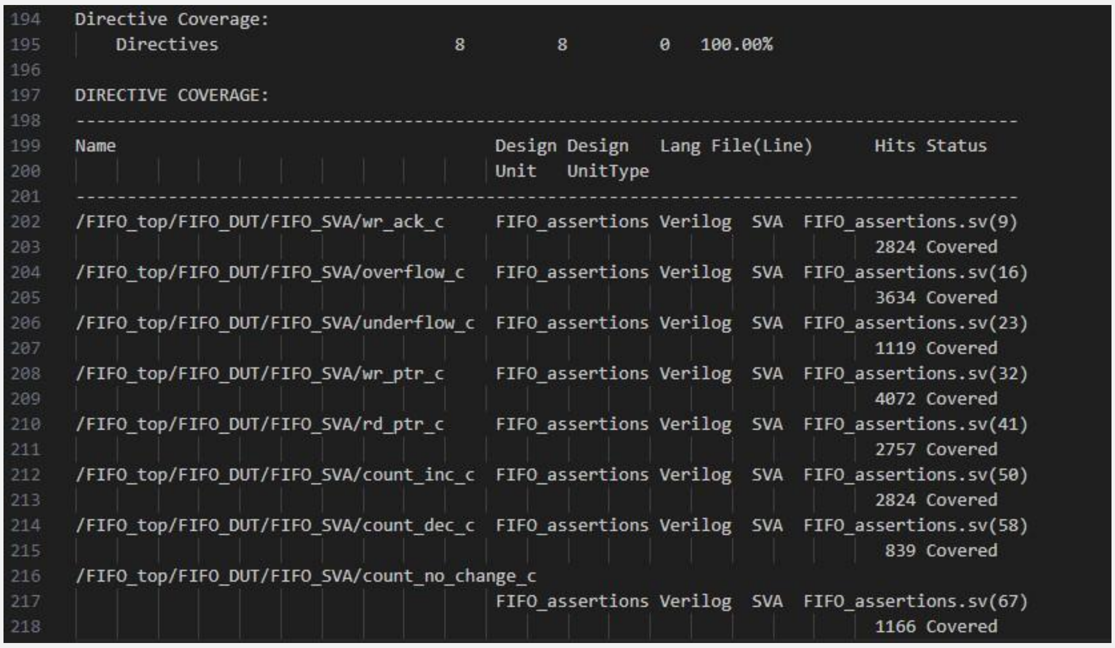


```
=============================================================================
=== Instance: /FIFO_top/FIFO_DUT/FIFO_SVA
=== Design Unit: work.FIFO_assertions
=============================================================================

Assertion Coverage:
    Assertions                      13        13         0   100.00%
-----------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
-----------------------------------------------------------------
/FIFO_top/FIFO_DUT/FIFO_SVA/wr_ack_a
                     FIFO_assertions.sv(8)              0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/overflow_a
                     FIFO_assertions.sv(15)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/underflow_a
                     FIFO_assertions.sv(22)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/wr_ptr_a
                     FIFO_assertions.sv(31)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/rd_ptr_a
                     FIFO_assertions.sv(40)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/count_inc_a
                     FIFO_assertions.sv(49)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/count_dec_a
                     FIFO_assertions.sv(57)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/count_no_change_a
                     FIFO_assertions.sv(66)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/full_check
                     FIFO_assertions.sv(76)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/almost_full_check
                     FIFO_assertions.sv(80)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/empty_check
```



```
/FIFO_top/FIFO_DUT/FIFO_SVA/count_dec_a
                     FIFO_assertions.sv(57)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/count_no_change_a
                     FIFO_assertions.sv(66)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/full_check
                     FIFO_assertions.sv(76)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/almost_full_check
                     FIFO_assertions.sv(80)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/empty_check
```

```
                     FIFO_assertions.sv(84)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/comb_outputs/almostempty_check
                     FIFO_assertions.sv(88)             0          1
/FIFO_top/FIFO_DUT/FIFO_SVA/reset_outputs/reset
                     FIFO_assertions.sv(98)             0          1
```

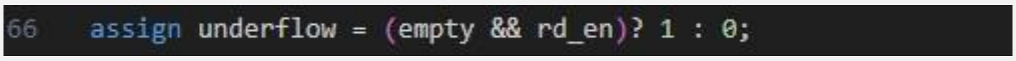


```
Directive Coverage:
    Directives                      8         8         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/FIFO_top/FIFO_DUT/FIFO_SVA/wr_ack_c     FIFO_assertions Verilog  SVA  FIFO_assertions.sv(9)
                                                                             2824 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/overflow_c   FIFO_assertions Verilog  SVA  FIFO_assertions.sv(16)
                                                                             3634 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/underflow_c  FIFO_assertions Verilog  SVA  FIFO_assertions.sv(23)
                                                                             1119 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/wr_ptr_c     FIFO_assertions Verilog  SVA  FIFO_assertions.sv(32)
                                                                             4072 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/rd_ptr_c     FIFO_assertions Verilog  SVA  FIFO_assertions.sv(41)
                                                                             2757 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/count_inc_c  FIFO_assertions Verilog  SVA  FIFO_assertions.sv(50)
                                                                             2824 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/count_dec_c  FIFO_assertions Verilog  SVA  FIFO_assertions.sv(58)
                                                                              839 Covered
/FIFO_top/FIFO_DUT/FIFO_SVA/count_no_change_c
                                        FIFO_assertions Verilog  SVA  FIFO_assertions.sv(67)
                                                                             1166 Covered
```

9- Bugs Detected:

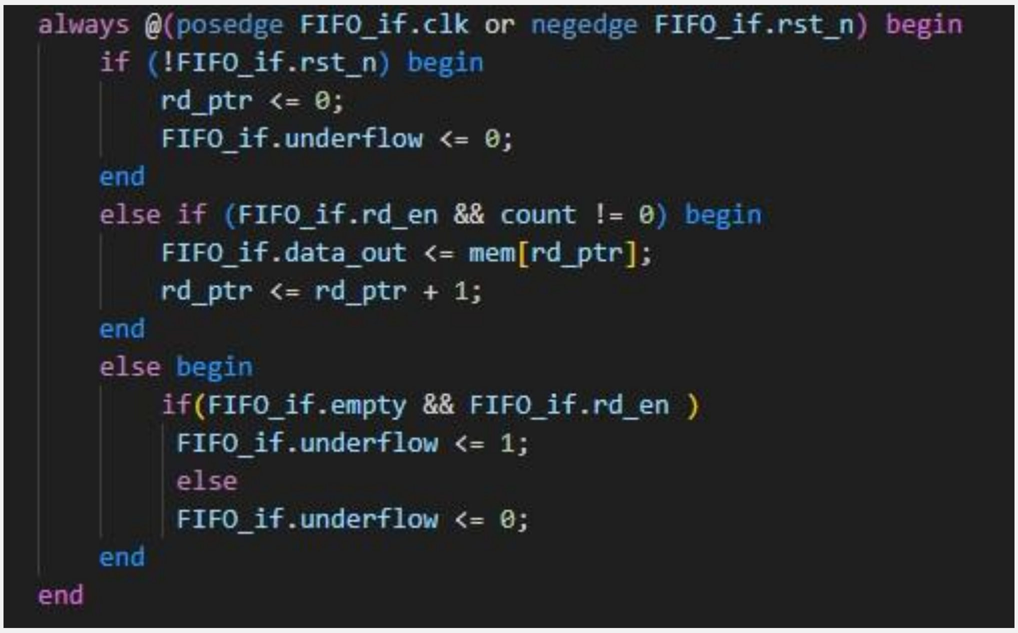
1- Output underflow is supposed to be sequential output while it was assigned as combinational output

Bug:



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

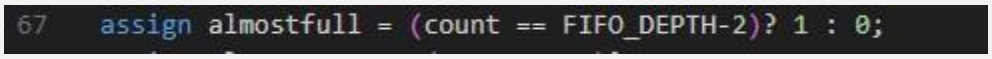
Fixed Code :



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

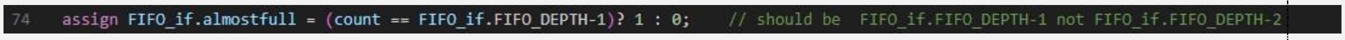
2-Output almostfull is asserted to be high when count = FIFO_DEPTH -1 not FIFO_DEPTH -2 (when reaching FIFO_DEPTH -1 means we 've written FIFO_DEPTH -1 times so there's only one write operation needed to be full)

Bug:



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

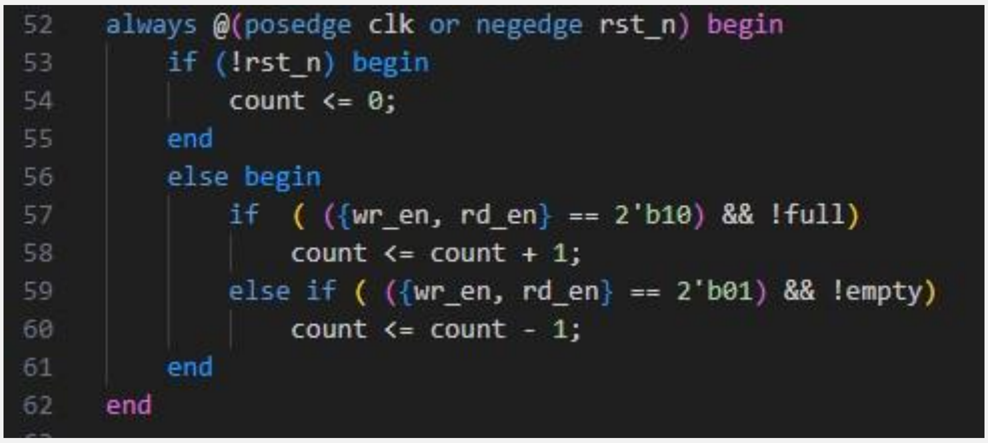
Fixed Design:



```
assign FIFO_if.almostfull = (count == FIFO_if.FIFO_DEPTH-1)? 1 : 0;    // should be  FIFO_if.FIFO_DEPTH-1 not FIFO_if.FIFO_DEPTH-2
```

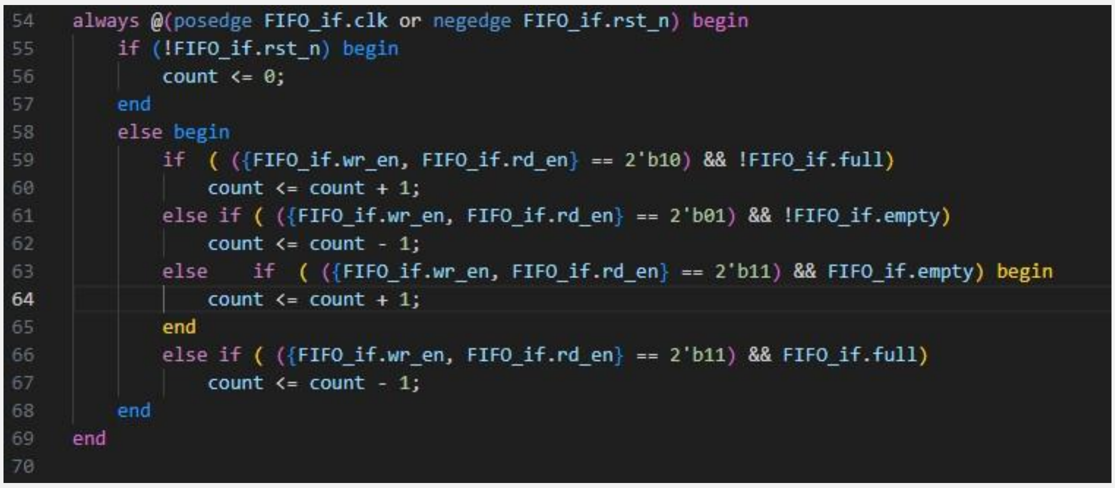
3- The RTL of Design doesn't cover cases when write && read enables asserted while the fifo is empty or full.

Bug:



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

Fixed Design:

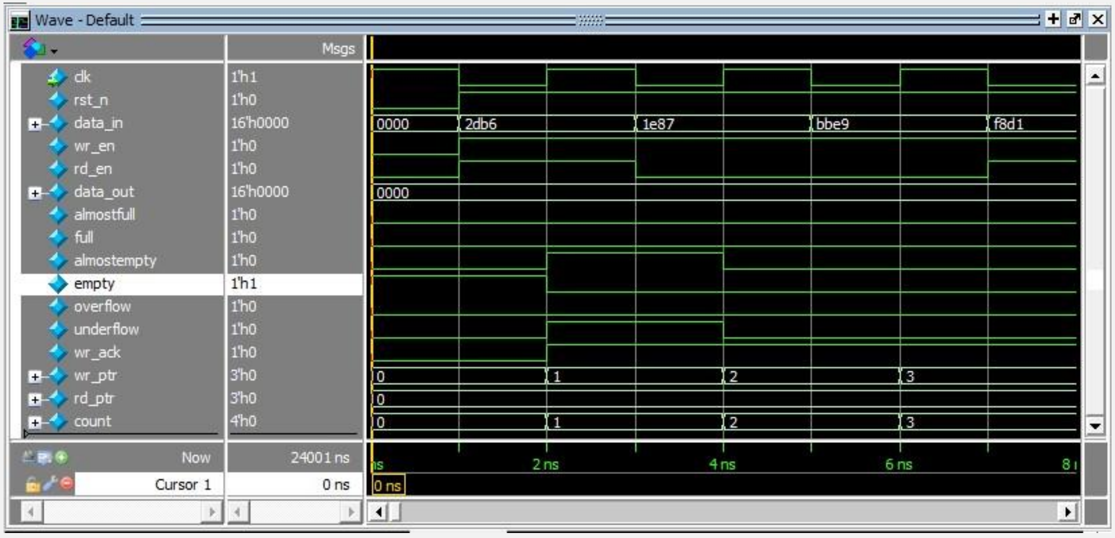


```
always @(posedge FIFO_if.clk or negedge FIFO_if.rst_n) begin
    if (!FIFO_if.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({FIFO_if.wr_en, FIFO_if.rd_en} == 2'b10) && !FIFO_if.full)
            count <= count + 1;
        else if ( ({FIFO_if.wr_en, FIFO_if.rd_en} == 2'b01)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

4- bug: when reset asserted overflow ,underflow, and wr_Ack must be zero

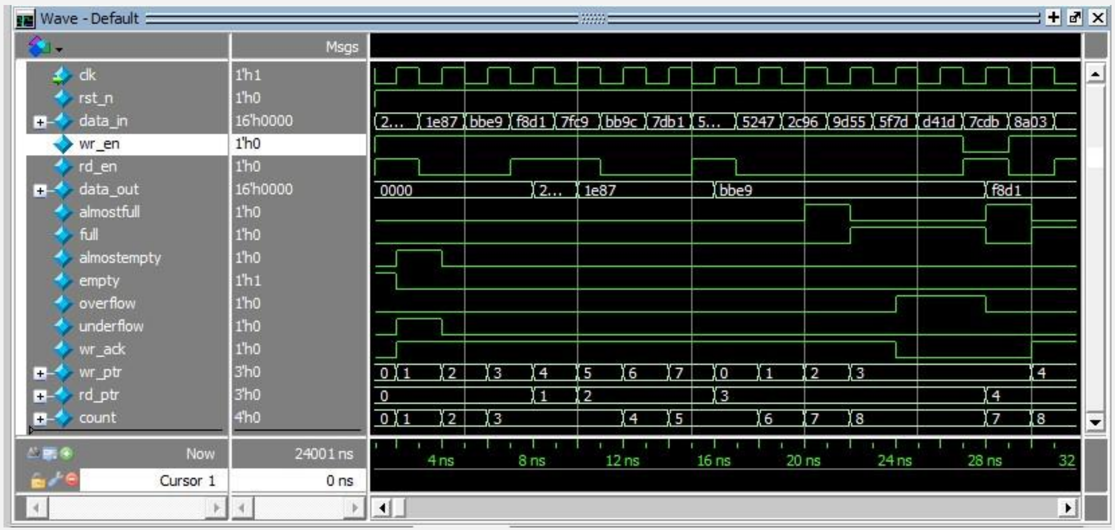
10- Waveform Snippets:

1.Reset Sequence: 0 – 1 ns

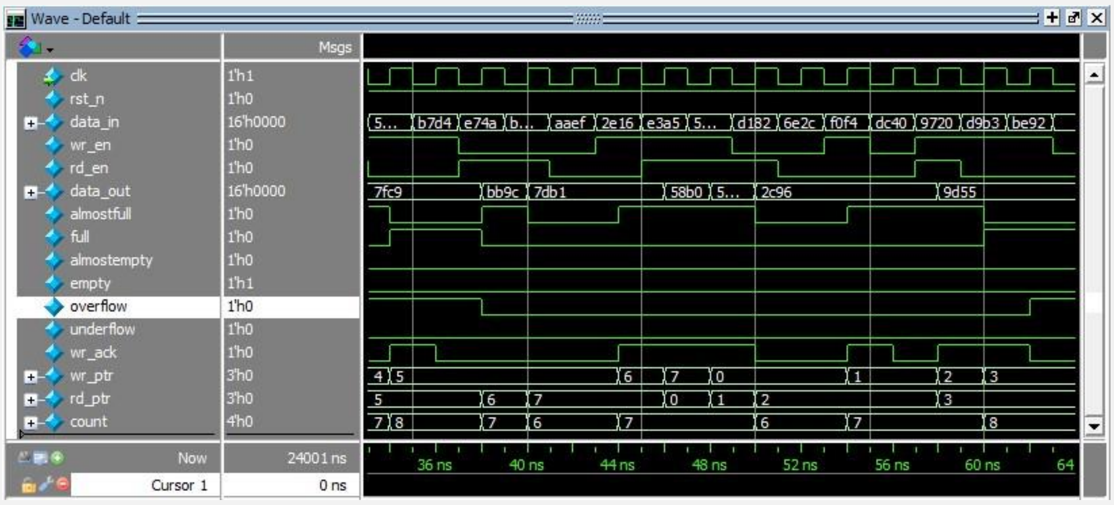


```
Wave - Default
                Msgs
clk             1'h1
rst_n           1'h0
data_in         16'h0000
wr_en           1'h0
rd_en           1'h0
data_out        16'h0000
almostfull      1'h0
full            1'h0
almostempty     1'h0
empty           1'h1
overflow        1'h0
underflow       1'h0
wr_ack          1'h0
wr_ptr          3'h0
rd_ptr          3'h0
count           4'h0
Now             24001 ns
Cursor 1        0 ns
```

2.Write Read Sequence: 1- 20001 ns

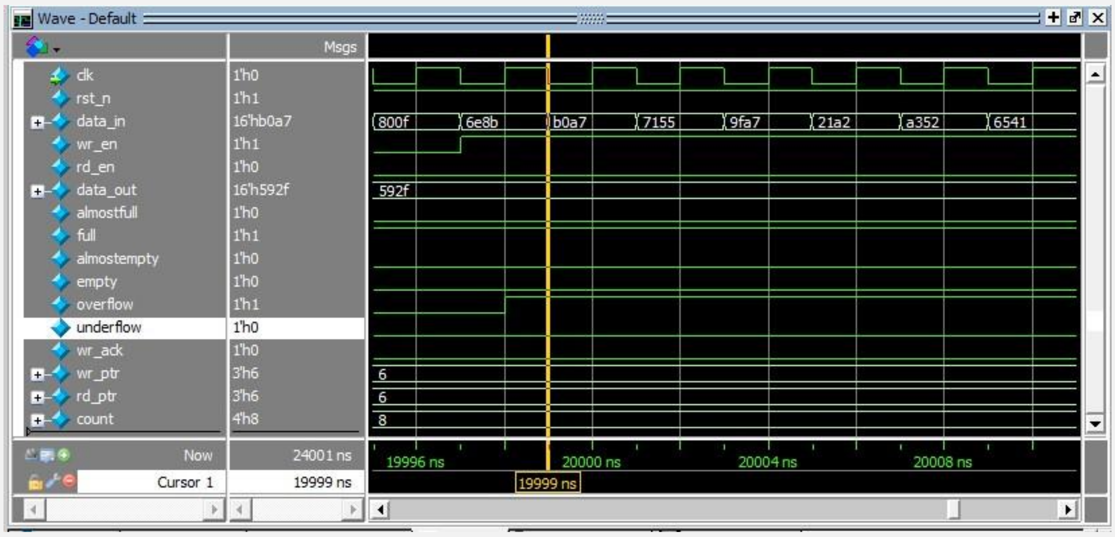


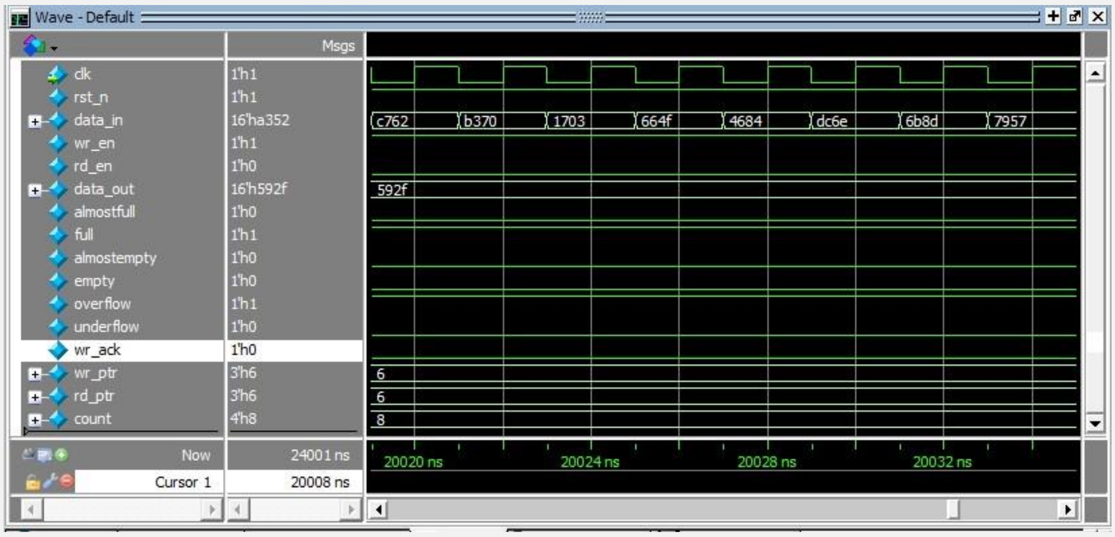
```
Wave - Default
                Msgs
clk             1'h1
rst_n           1'h0
data_in         16'h0000
wr_en           1'h0
rd_en           1'h0
data_out        16'h0000
almostfull      1'h0
full            1'h0
almostempty     1'h0
empty           1'h1
overflow        1'h0
underflow       1'h0
wr_ack          1'h0
wr_ptr          3'h0
rd_ptr          3'h0
count           4'h0
Now             24001 ns
Cursor 1        0 ns
```



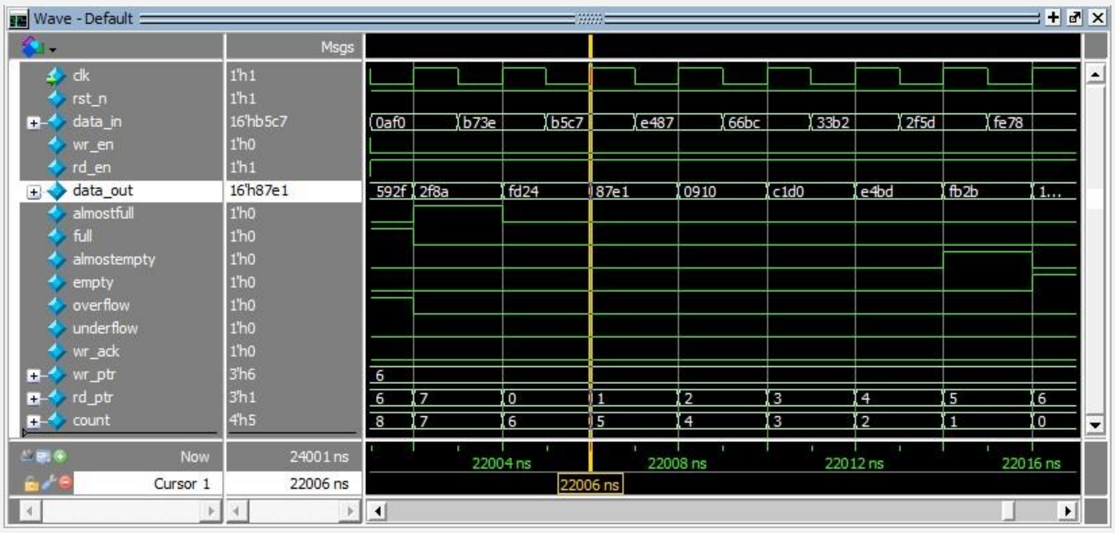
```
Wave - Default
                Msgs
clk             1'h1
rst_n           1'h0
data_in         16'h0000
wr_en           1'h0
rd_en           1'h0
data_out        16'h0000
almostfull      1'h0
full            1'h0
almostempty     1'h0
empty           1'h1
overflow        1'h0
underflow       1'h0
wr_ack          1'h0
wr_ptr          3'h0
rd_ptr          3'h0
count           4'h0
Now             24001 ns
Cursor 1        0 ns
```

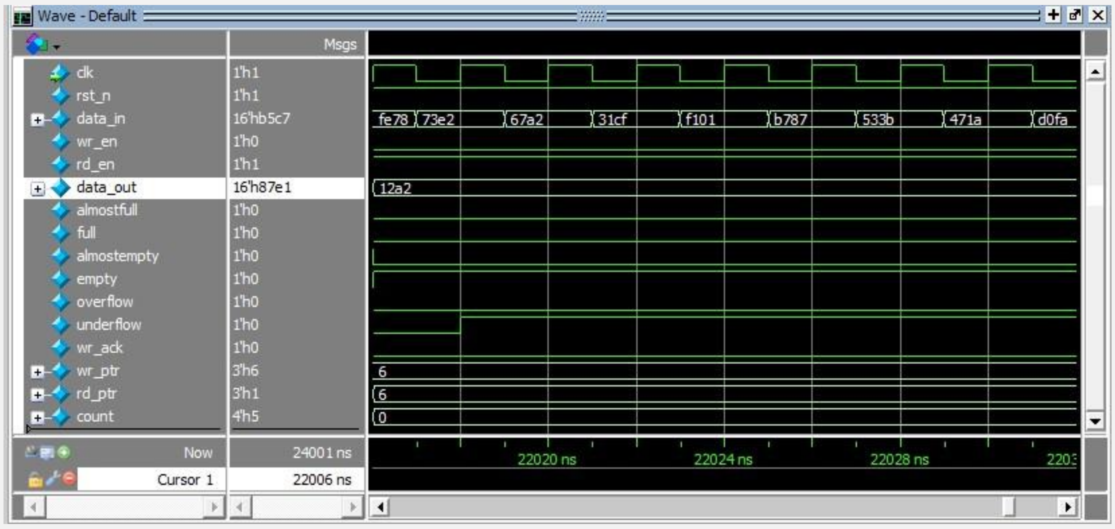
3.Write Sequence : 20001 - 22001 ns



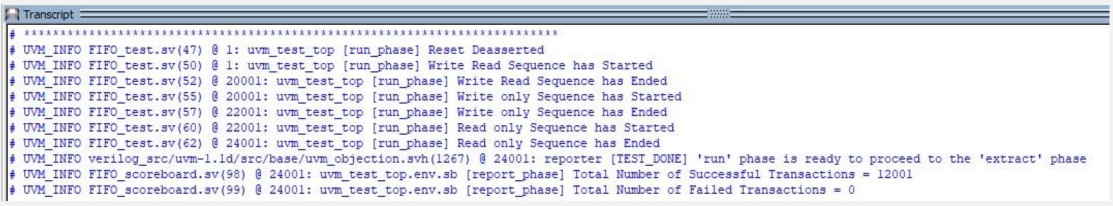


4. Read Sequence : 22001 – 24001 ns

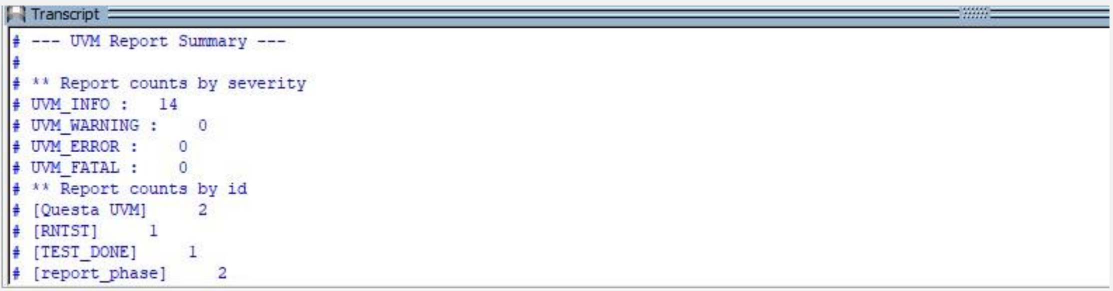




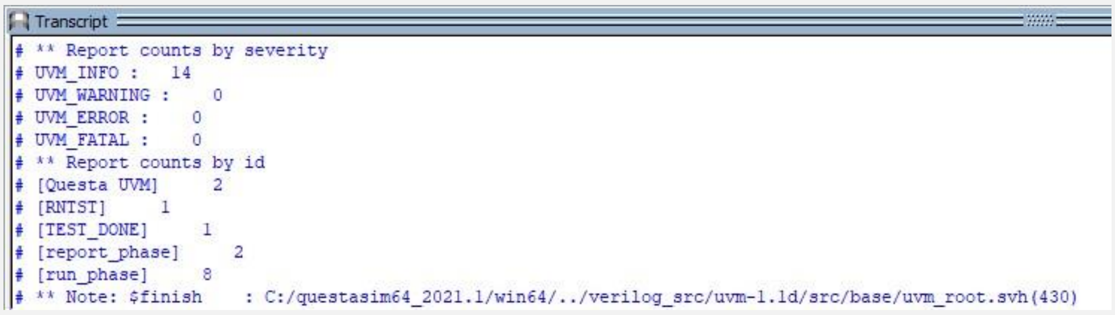
Transcript & UVM Testbench Summary :



```
# ************************************************************************
# UVM_INFO FIFO_test.sv(47) @ 1: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO FIFO_test.sv(50) @ 1: uvm_test_top [run_phase] Write Read Sequence has Started
# UVM_INFO FIFO_test.sv(52) @ 20001: uvm_test_top [run_phase] Write Read Sequence has Ended
# UVM_INFO FIFO_test.sv(55) @ 20001: uvm_test_top [run_phase] Write only Sequence has Started
# UVM_INFO FIFO_test.sv(57) @ 22001: uvm_test_top [run_phase] Write only Sequence has Ended
# UVM_INFO FIFO_test.sv(60) @ 22001: uvm_test_top [run_phase] Read only Sequence has Started
# UVM_INFO FIFO_test.sv(62) @ 24001: uvm_test_top [run_phase] Read only Sequence has Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 24001: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_scoreboard.sv(98) @ 24001: uvm_test_top.env.sb [report_phase] Total Number of Successful Transactions = 12001
# UVM_INFO FIFO_scoreboard.sv(99) @ 24001: uvm_test_top.env.sb [report_phase] Total Number of Failed Transactions = 0
```



```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    14
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
```


Transcript

```
# ** Report counts by severity
# UVM_INFO :   14
# 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]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
```