FIFO

[Verification project using UVM]

By: Mina Ehab

[under the superviser of Eng.kareem waseem]

Contents

Introduction:

What is a FIFO and How it works?

Basic Operation:

- A FIFO has two primary operations: **write (enqueue)** and **read (dequeue)**.
- Data is written into the FIFO using a **write pointer**, which points to the next available location in memory.
- Data is read out from the FIFO using a **read pointer**, which points to the next location where data should be read.
- The data is stored in sequential order, meaning the order in which it was written is the order in which it will be read.

First-In, First-Out Principle:

- The first data written to the FIFO is the first to be read, ensuring that no data is skipped or read out of order.
- Think of a FIFO as a pipeline: whatever goes in first comes out first.

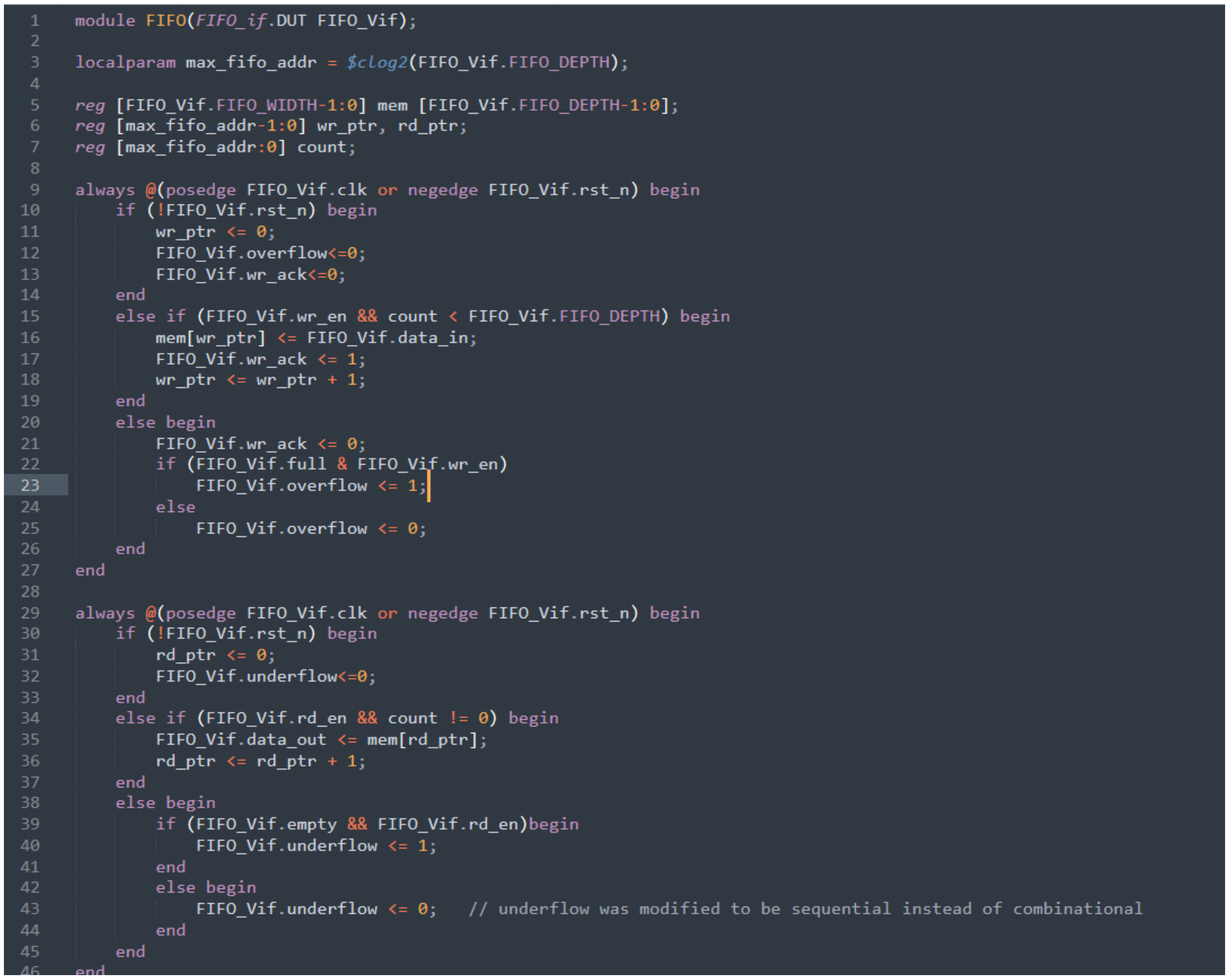
Explanation of our UVM Environment

- **Sequences** in the top module generate transactions.
- **uvm_sequencer** sends transactions to the **uvm_driver**, which drives the stimulus to the DUT using the **virtual interface**.
- The **uvm_monitor** listens to the DUT's outputs through the same virtual interface.
- The captured data is sent to the **uvm_scoreboard** for verification, and coverage is collected by the **coverage collector**.
- All components work together to ensure the DUT behaves as expected and that all test scenarios are covered.

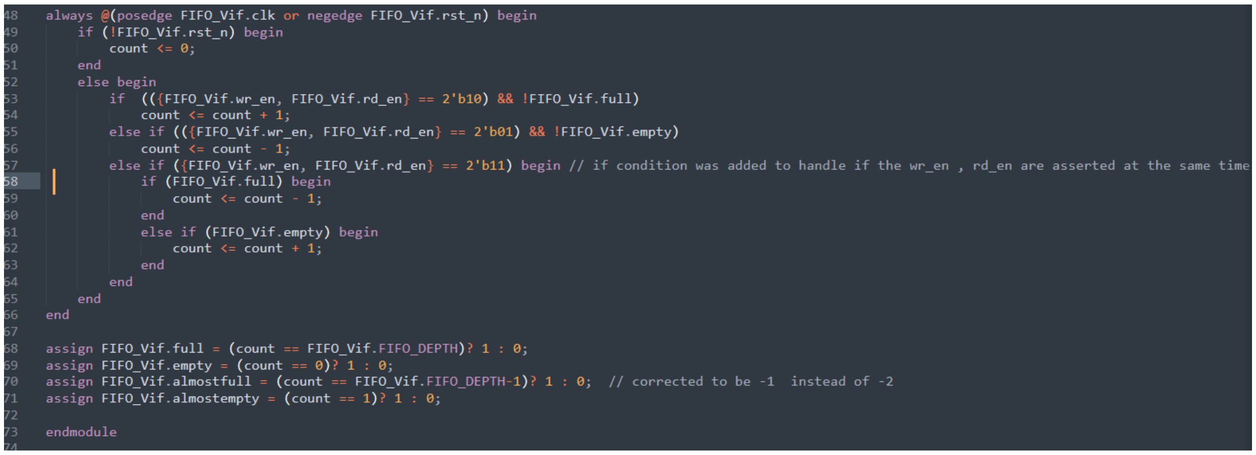
Bugs report:

- Overflow, underflow and wr_ack signals were zeroed at reset asserting.
- Underflow was modified to be sequential instead of combinational as specs.
- If condition was added to handle count behavior in case of write and reading together.
- Almostfull was modified to be high if count = FIFO Depth – 1.

Design code:

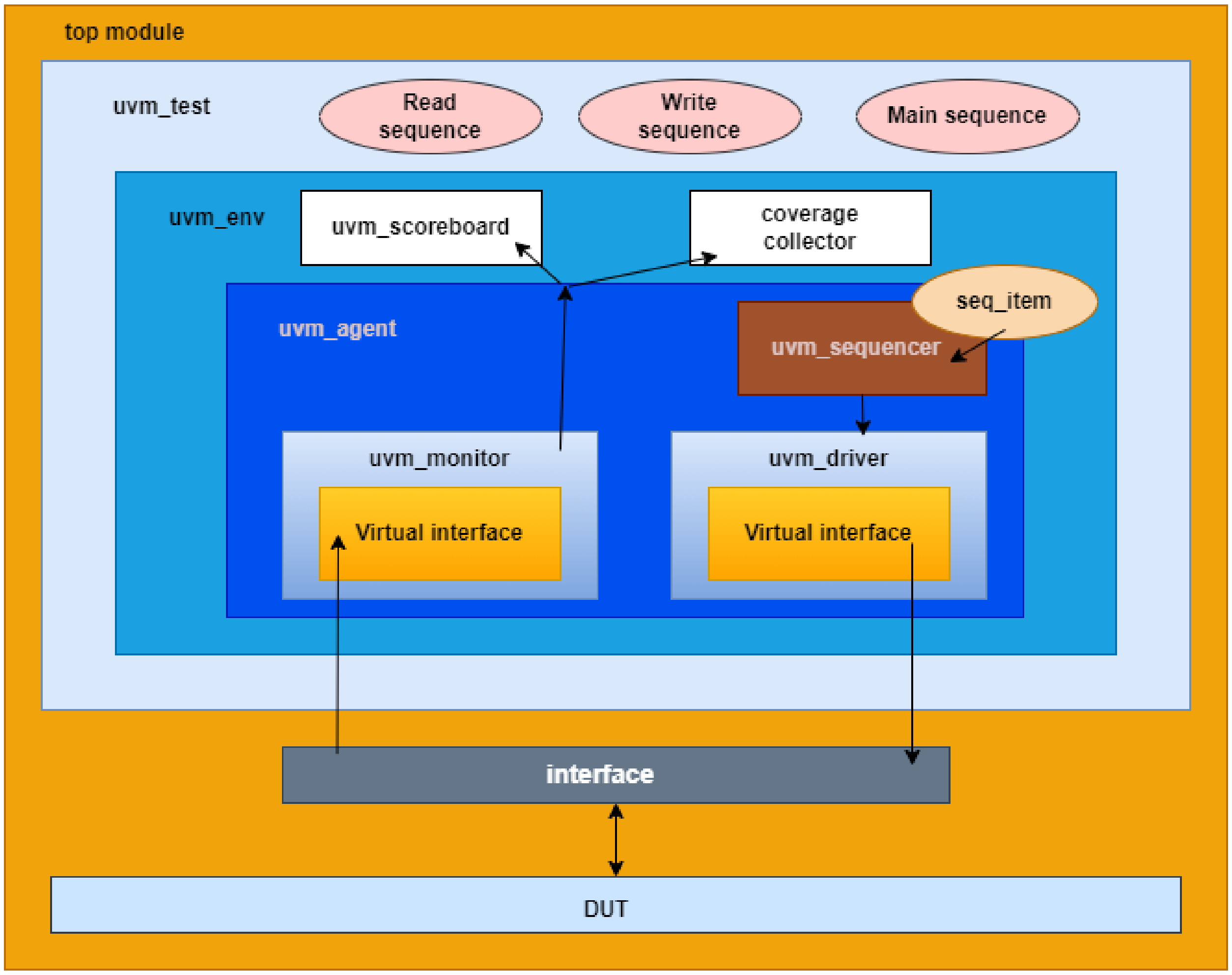


```
module FIFO(FIFO_if.DUT FIFO_Vif);

localparam max_fifo_addr = $clog2(FIFO_Vif.FIFO_DEPTH);

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

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

always @(posedge FIFO_Vif.clk or negedge FIFO_Vif.rst_n) begin
    if (!FIFO_Vif.rst_n) begin
        rd_ptr <= 0;
        FIFO_Vif.underflow<=0;
    end
    else if (FIFO_Vif.rd_en && count != 0) begin
        FIFO_Vif.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin
        if (FIFO_Vif.empty && FIFO_Vif.rd_en)begin
            FIFO_Vif.underflow <= 1;
        end
        else begin
            FIFO_Vif.underflow <= 0;   // underflow was modified to be sequential instead of combinational
        end
    end
end
```

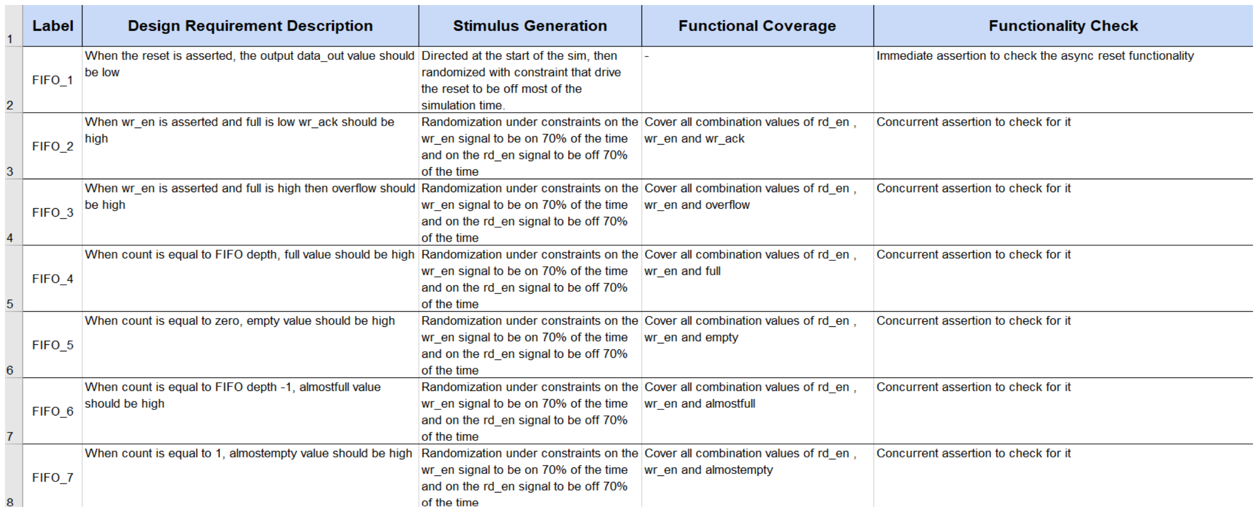


```
always @(posedge FIFO_Vif.clk or negedge FIFO_Vif.rst_n) begin
    if (!FIFO_Vif.rst_n) begin
        count <= 0;
    end
    else begin
        if  (({FIFO_Vif.wr_en, FIFO_Vif.rd_en} == 2'b10) && !FIFO_Vif.full)
            count <= count + 1;
        else if (({FIFO_Vif.wr_en, FIFO_Vif.rd_en} == 2'b01) && !FIFO_Vif.empty)
            count <= count - 1;
        else if ({FIFO_Vif.wr_en, FIFO_Vif.rd_en} == 2'b11) begin // if condition was added to handle if the wr_en , rd_en are asserted at the same time
            if (FIFO_Vif.full) begin
                count <= count - 1;
            end
            else if (FIFO_Vif.empty) begin
                count <= count + 1;
            end
        end
    end
end

assign FIFO_Vif.full = (count == FIFO_Vif.FIFO_DEPTH)? 1 : 0;
assign FIFO_Vif.empty = (count == 0)? 1 : 0;
assign FIFO_Vif.almostfull = (count == FIFO_Vif.FIFO_DEPTH-1)? 1 : 0;  // corrected to be -1  instead of -2
assign FIFO_Vif.almostempty = (count == 1)? 1 : 0;

endmodule
```

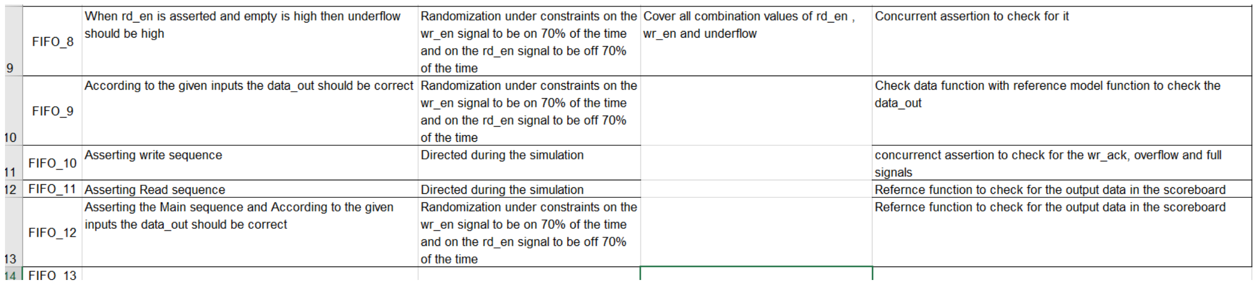
UVM structure:



Verification plan:



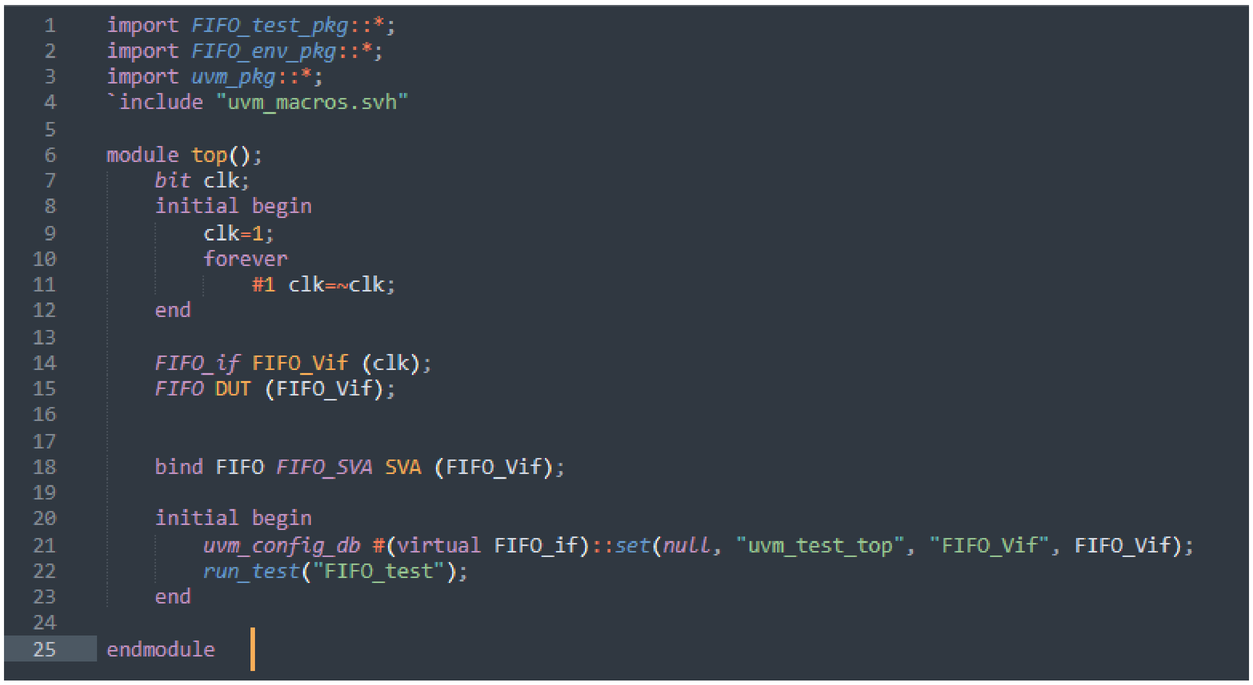
Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted, the output data_out value should be low	Directed at the start of the sim, then randomized with constraint that drive the reset to be off most of the simulation time.	-	Immediate assertion to check the async reset functionality
FIFO_2	When wr_en is asserted and full is low wr_ack should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and wr_ack	Concurrent assertion to check for it
FIFO_3	When wr_en is asserted and full is high then overflow should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and overflow	Concurrent assertion to check for it
FIFO_4	When count is equal to FIFO depth, full value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and full	Concurrent assertion to check for it
FIFO_5	When count is equal to zero, empty value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and empty	Concurrent assertion to check for it
FIFO_6	When count is equal to FIFO depth -1, almostfull value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and almostfull	Concurrent assertion to check for it
FIFO_7	When count is equal to 1, almostempty value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and almostempty	Concurrent assertion to check for it



9	FIFO_8	When rd_en is asserted and empty is high then underflow should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and underflow	Concurrent assertion to check for it
10	FIFO_9	According to the given inputs the data_out should be correct	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time		Check data function with reference model function to check the data_out
11	FIFO_10	Asserting write sequence	Directed during the simulation		concurrenct assertion to check for the wr_ack, overflow and full signals
12	FIFO_11	Asserting Read sequence	Directed during the simulation		Refernce function to check for the output data in the scoreboard
13	FIFO_12	Asserting the Main sequence and According to the given inputs the data_out should be correct	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time		Refernce function to check for the output data in the scoreboard
14	FIFO_13				

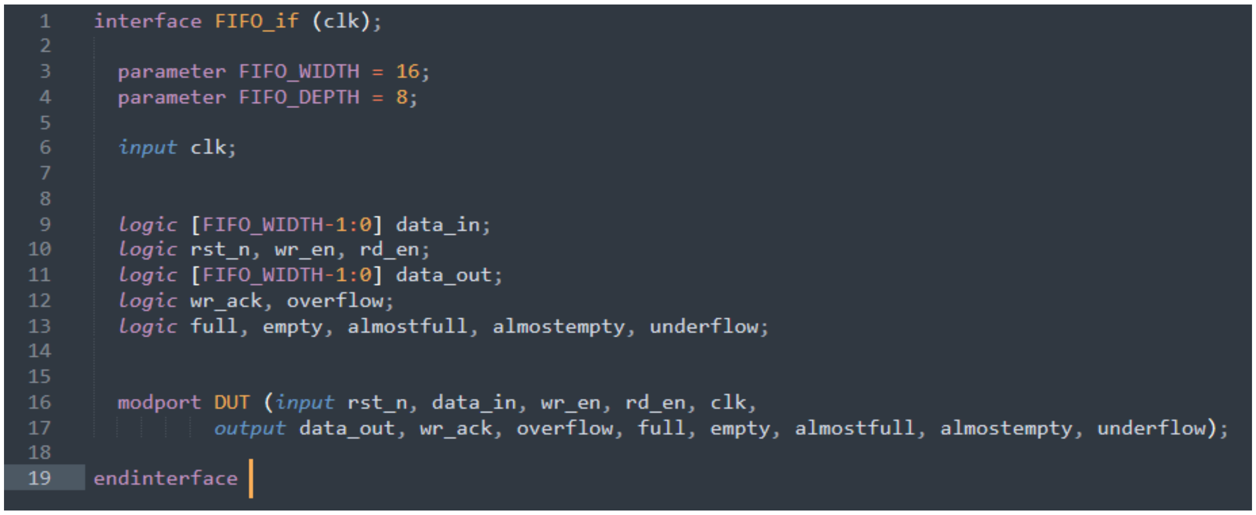
UVM Codes

Top:



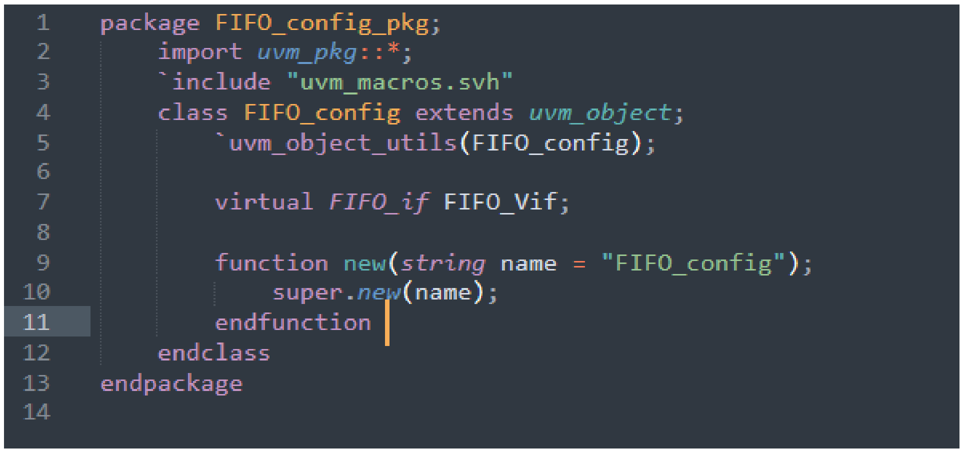
```
import FIFO_test_pkg::*;
import FIFO_env_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

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

    FIFO_if FIFO_Vif (clk);
    FIFO DUT (FIFO_Vif);


    bind FIFO FIFO_SVA SVA (FIFO_Vif);

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

endmodule
```

Interface:



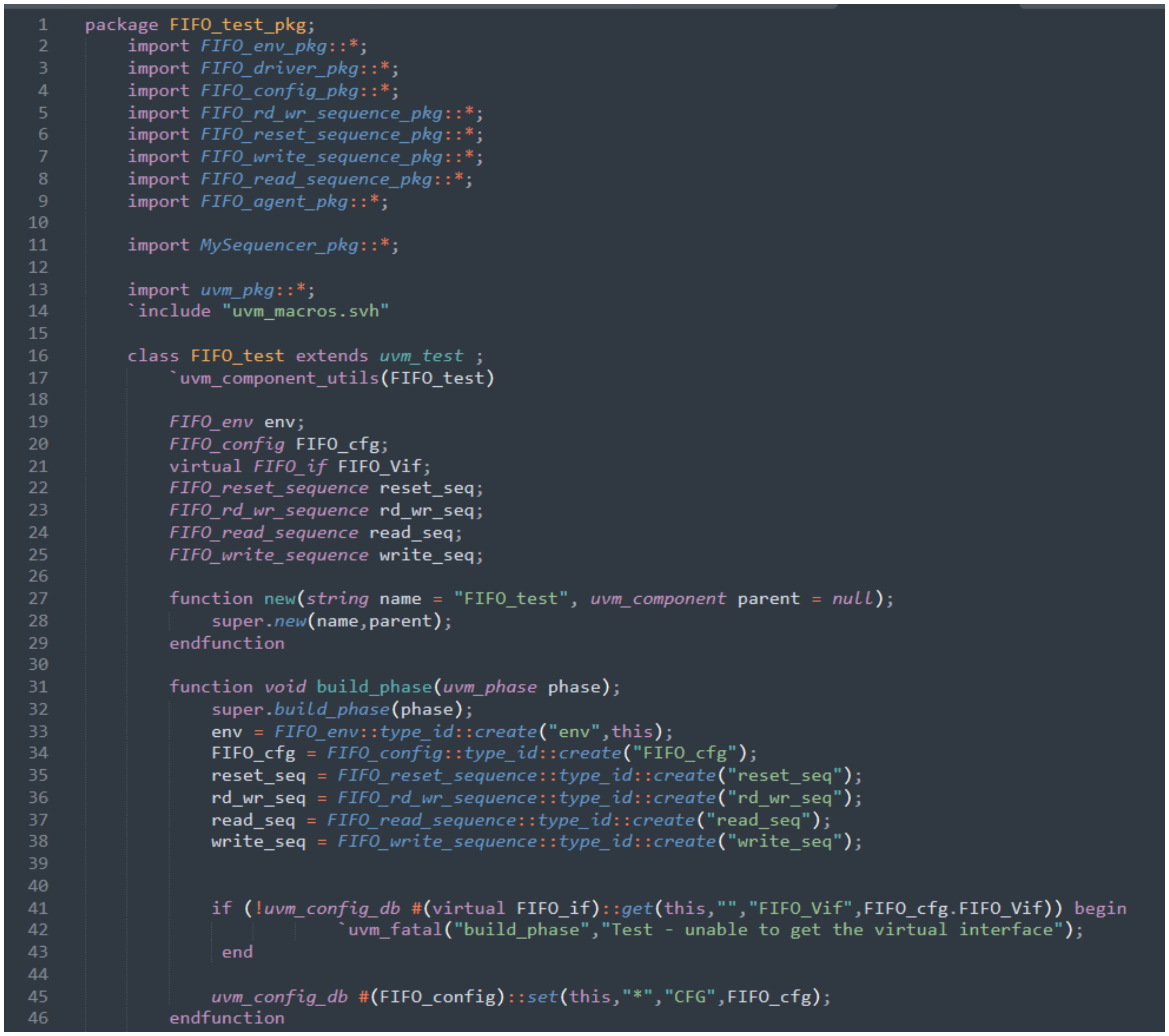
```
interface FIFO_if (clk);

  parameter FIFO_WIDTH = 16;
  parameter FIFO_DEPTH = 8;

  input clk;


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


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

endinterface
```

Configuration:

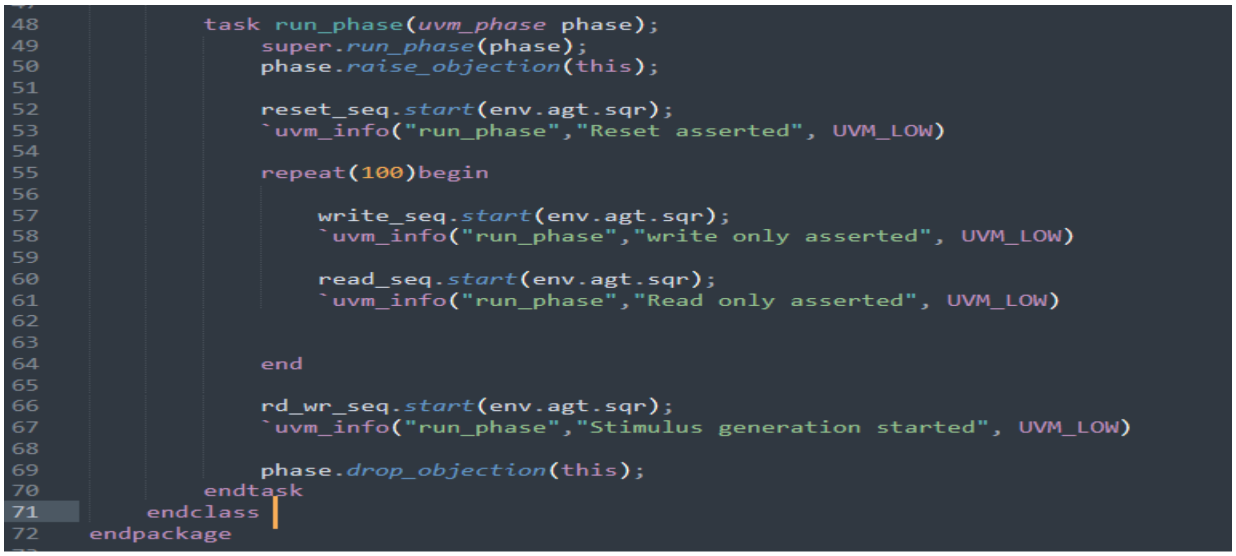


```
package FIFO_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_config extends uvm_object;
        `uvm_object_utils(FIFO_config);

        virtual FIFO_if FIFO_Vif;

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

Test:

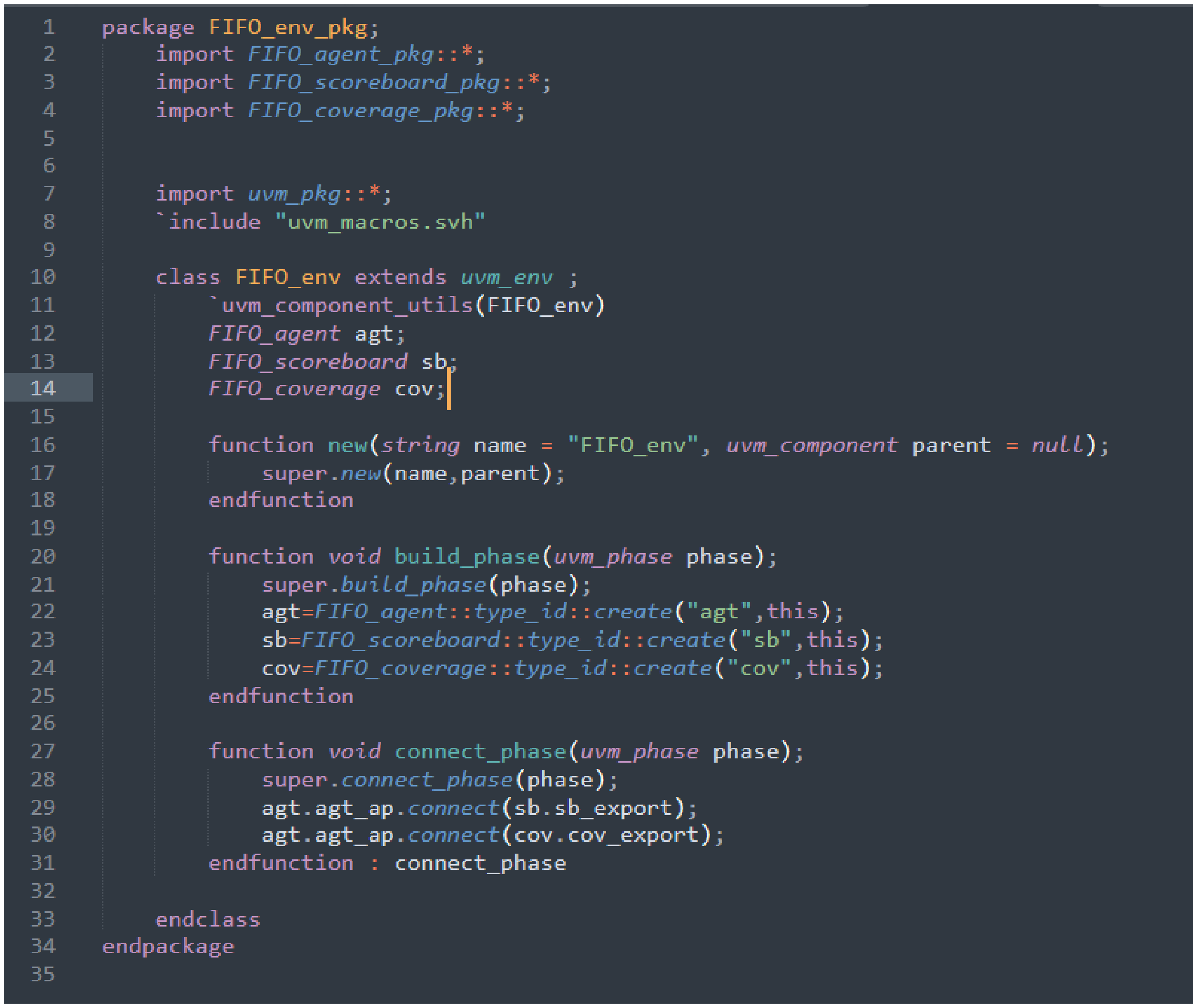


```
package FIFO_test_pkg;
    import FIFO_env_pkg::*;
    import FIFO_driver_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_rd_wr_sequence_pkg::*;
    import FIFO_reset_sequence_pkg::*;
    import FIFO_write_sequence_pkg::*;
    import FIFO_read_sequence_pkg::*;
    import FIFO_agent_pkg::*;

    import MySequencer_pkg::*;

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

    class FIFO_test extends uvm_test ;
        `uvm_component_utils(FIFO_test)

        FIFO_env env;
        FIFO_config FIFO_cfg;
        virtual FIFO_if FIFO_Vif;
        FIFO_reset_sequence reset_seq;
        FIFO_rd_wr_sequence rd_wr_seq;
        FIFO_read_sequence read_seq;
        FIFO_write_sequence write_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::type_id::create("FIFO_cfg");
            reset_seq = FIFO_reset_sequence::type_id::create("reset_seq");
            rd_wr_seq = FIFO_rd_wr_sequence::type_id::create("rd_wr_seq");
            read_seq = FIFO_read_sequence::type_id::create("read_seq");
            write_seq = FIFO_write_sequence::type_id::create("write_seq");


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

            uvm_config_db #(FIFO_config)::set(this,"*","CFG",FIFO_cfg);
        endfunction
```



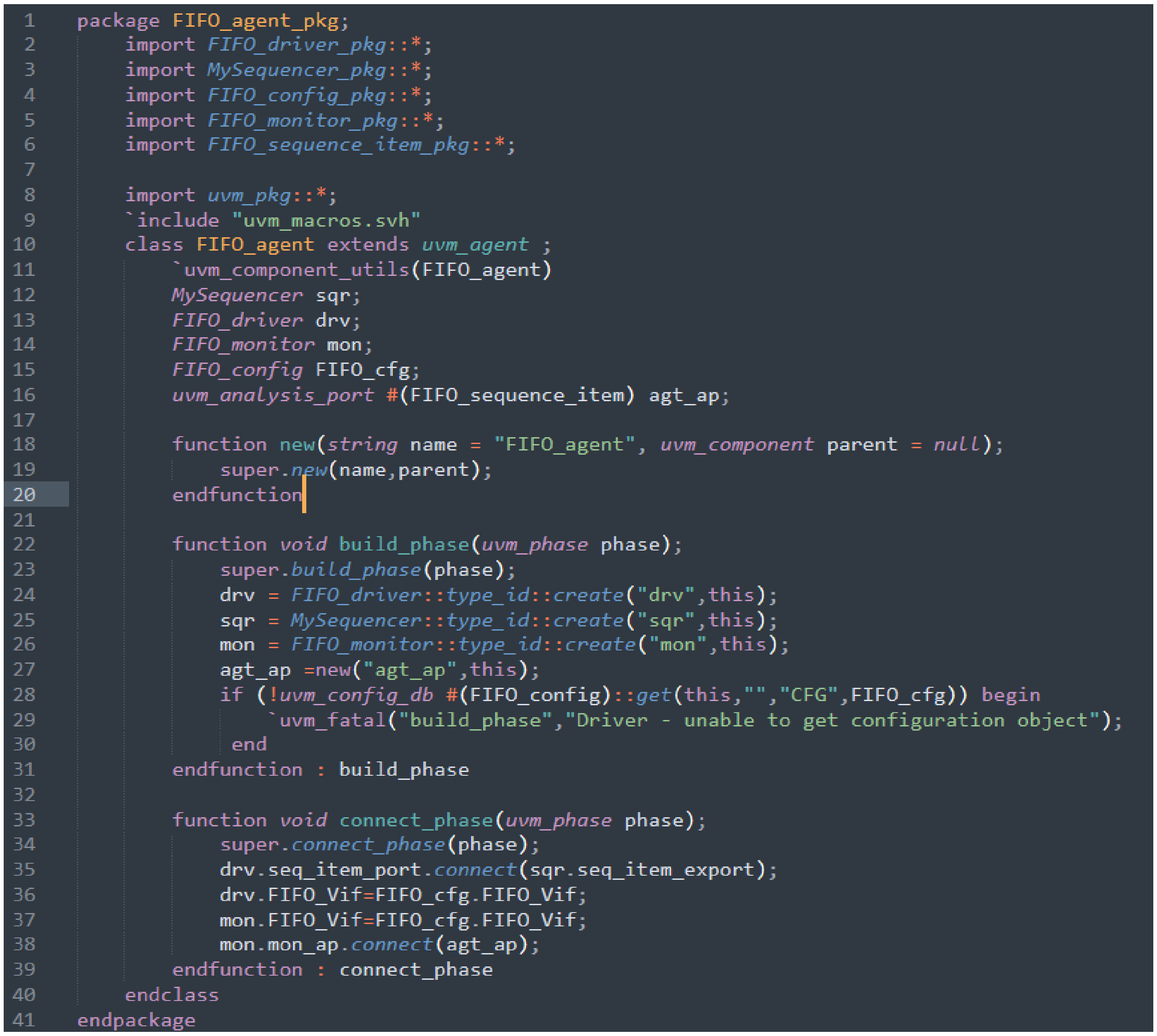
```
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);

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

            repeat(100)begin

                write_seq.start(env.agt.sqr);
                `uvm_info("run_phase","write only asserted", UVM_LOW)

                read_seq.start(env.agt.sqr);
                `uvm_info("run_phase","Read only asserted", UVM_LOW)


            end

            rd_wr_seq.start(env.agt.sqr);
            `uvm_info("run_phase","Stimulus generation started", UVM_LOW)

            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

Environment:



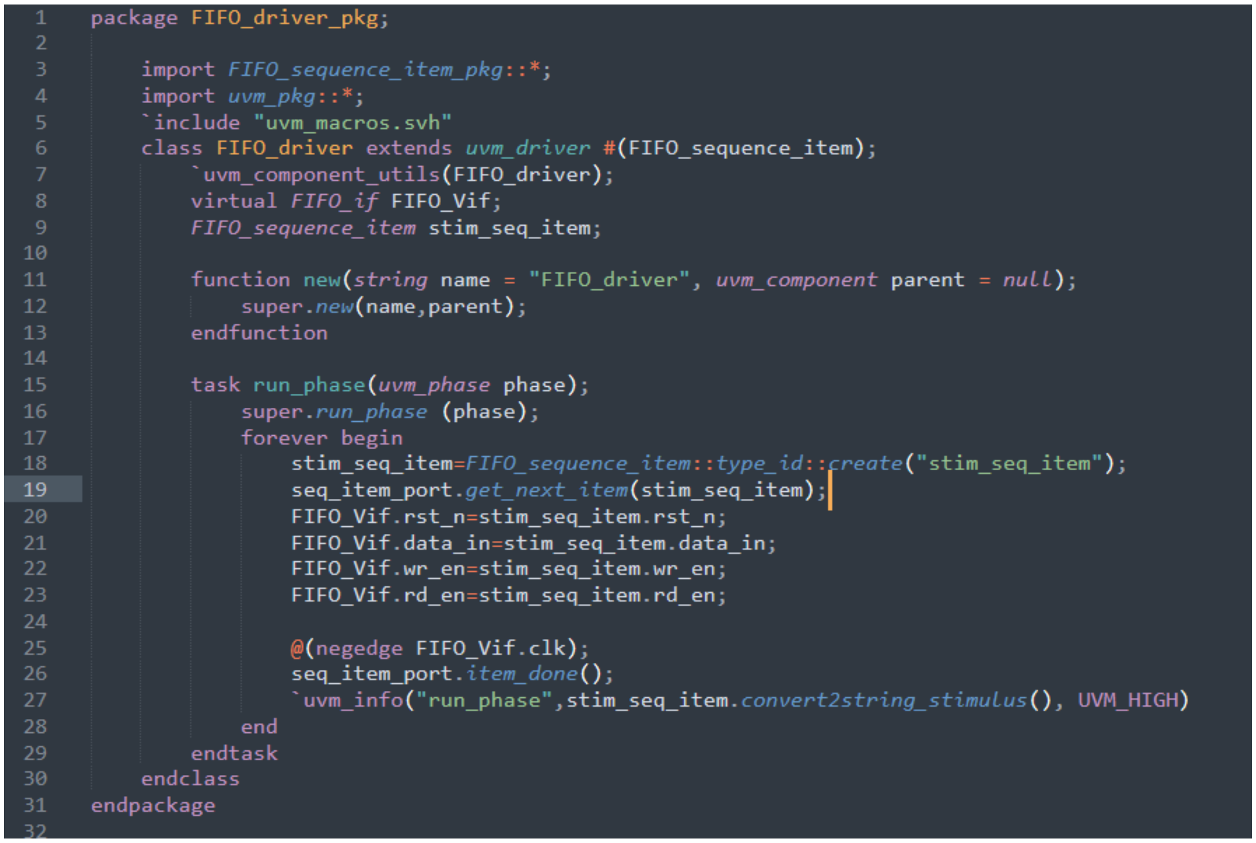
```
package FIFO_env_pkg;
    import FIFO_agent_pkg::*;
    import FIFO_scoreboard_pkg::*;
    import FIFO_coverage_pkg::*;


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

    class FIFO_env extends uvm_env ;
        `uvm_component_utils(FIFO_env)
        FIFO_agent agt;
        FIFO_scoreboard sb;
        FIFO_coverage cov;

        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);
            sb=FIFO_scoreboard::type_id::create("sb",this);
            cov=FIFO_coverage::type_id::create("cov",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 : connect_phase

    endclass
endpackage

```

Agent:



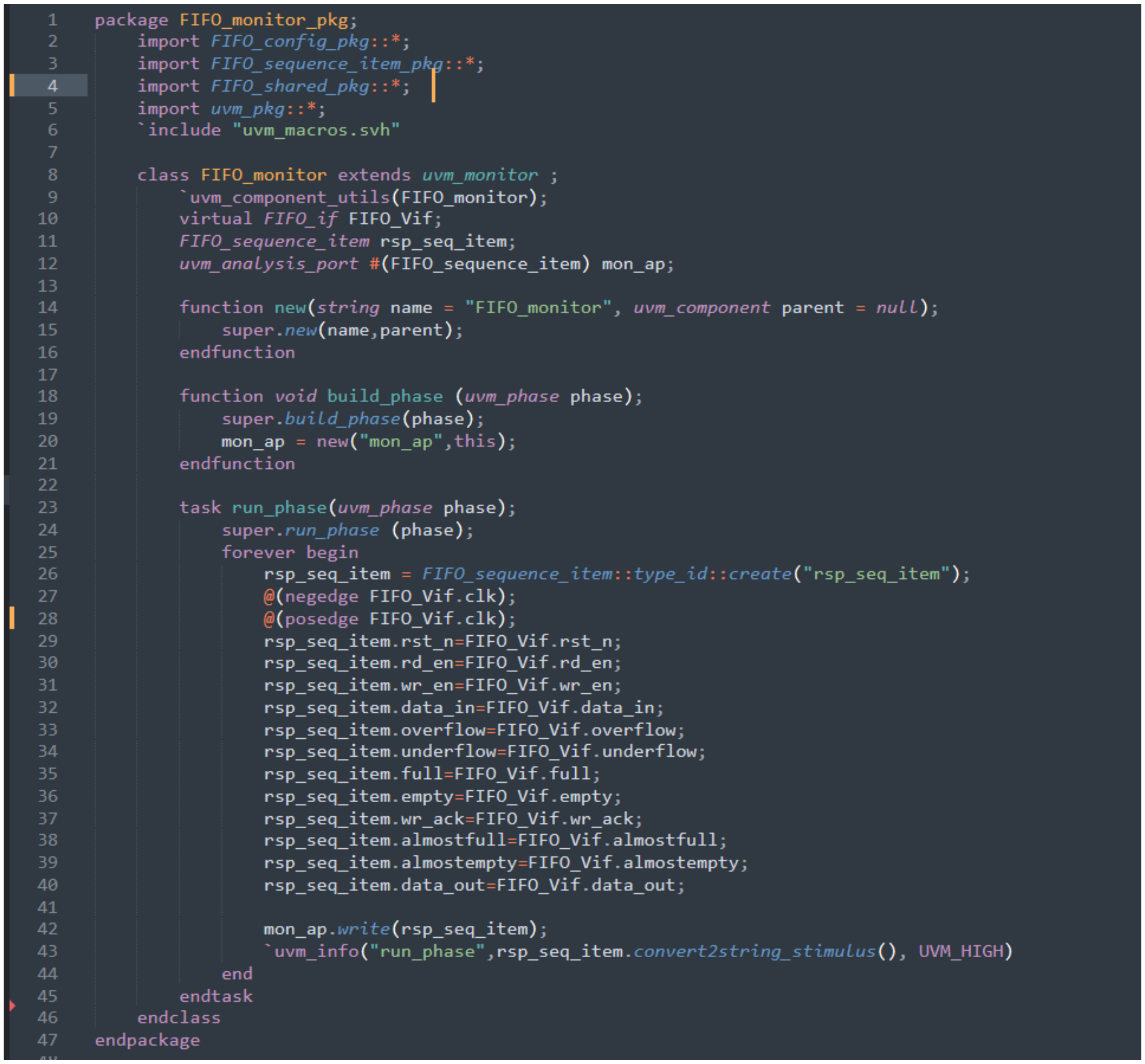
```
package FIFO_agent_pkg;
    import FIFO_driver_pkg::*;
    import MySequencer_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_monitor_pkg::*;
    import FIFO_sequence_item_pkg::*;

    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_agent extends uvm_agent ;
        `uvm_component_utils(FIFO_agent)
        MySequencer sqr;
        FIFO_driver drv;
        FIFO_monitor mon;
        FIFO_config FIFO_cfg;
        uvm_analysis_port #(FIFO_sequence_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);
            drv = FIFO_driver::type_id::create("drv",this);
            sqr = MySequencer::type_id::create("sqr",this);
            mon = FIFO_monitor::type_id::create("mon",this);
            agt_ap =new("agt_ap",this);
            if (!uvm_config_db #(FIFO_config)::get(this,"","CFG",FIFO_cfg)) begin
                `uvm_fatal("build_phase","Driver - unable to get configuration object");
             end
        endfunction : build_phase

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            drv.seq_item_port.connect(sqr.seq_item_export);
            drv.FIFO_Vif=FIFO_cfg.FIFO_Vif;
            mon.FIFO_Vif=FIFO_cfg.FIFO_Vif;
            mon.mon_ap.connect(agt_ap);
        endfunction : connect_phase
    endclass
endpackage
```

Driver:



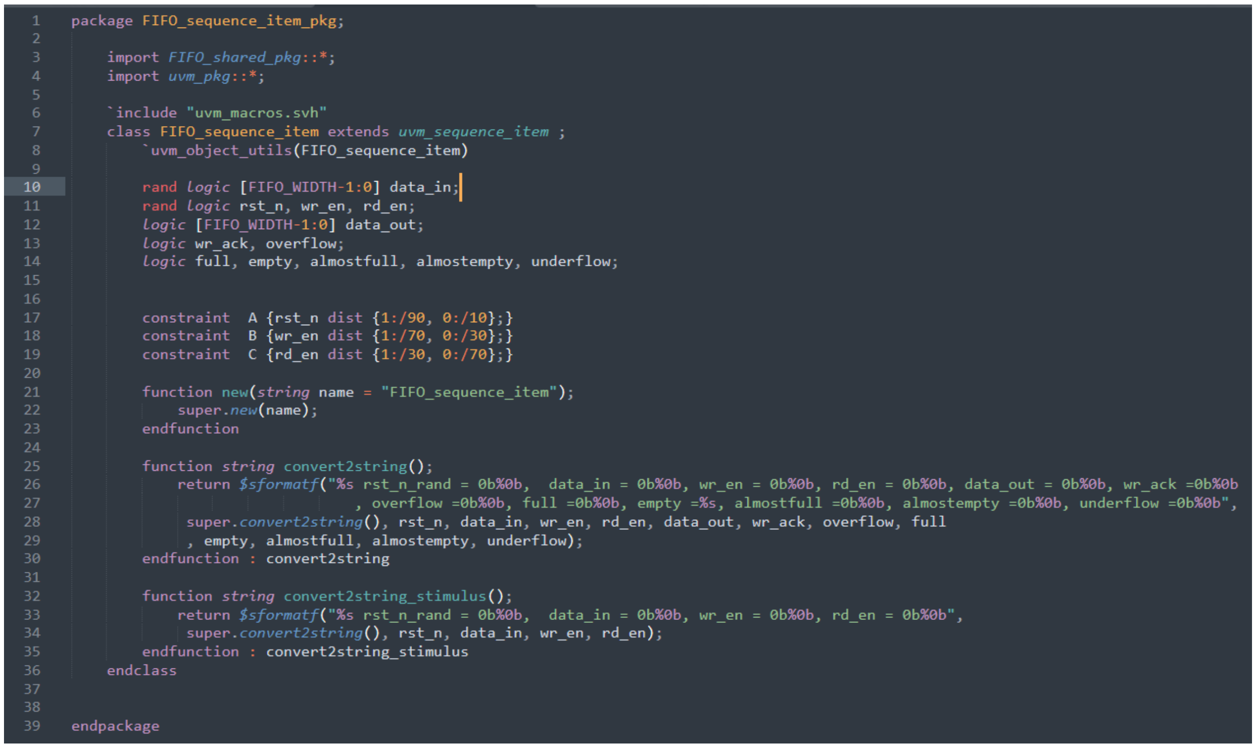
```
package FIFO_driver_pkg;

    import FIFO_sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_driver extends uvm_driver #(FIFO_sequence_item);
        `uvm_component_utils(FIFO_driver);
        virtual FIFO_if FIFO_Vif;
        FIFO_sequence_item stim_seq_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
                stim_seq_item=FIFO_sequence_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);
                FIFO_Vif.rst_n=stim_seq_item.rst_n;
                FIFO_Vif.data_in=stim_seq_item.data_in;
                FIFO_Vif.wr_en=stim_seq_item.wr_en;
                FIFO_Vif.rd_en=stim_seq_item.rd_en;

                @(negedge FIFO_Vif.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase",stim_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

Monitor:



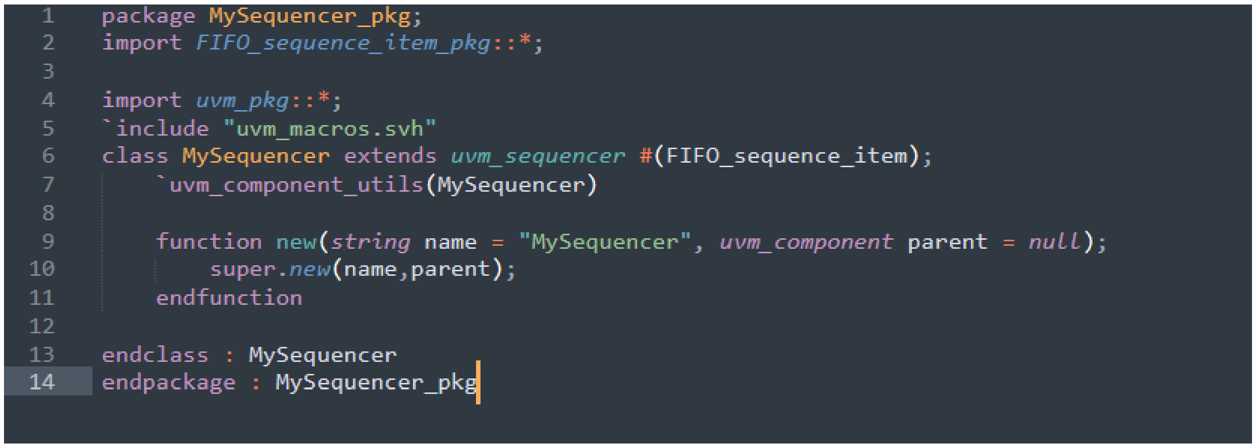
```
package FIFO_monitor_pkg;
    import FIFO_config_pkg::*;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class FIFO_monitor extends uvm_monitor ;
        `uvm_component_utils(FIFO_monitor);
        virtual FIFO_if FIFO_Vif;
        FIFO_sequence_item rsp_seq_item;
        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
                rsp_seq_item = FIFO_sequence_item::type_id::create("rsp_seq_item");
                @(negedge FIFO_Vif.clk);
                @(posedge FIFO_Vif.clk);
                rsp_seq_item.rst_n=FIFO_Vif.rst_n;
                rsp_seq_item.rd_en=FIFO_Vif.rd_en;
                rsp_seq_item.wr_en=FIFO_Vif.wr_en;
                rsp_seq_item.data_in=FIFO_Vif.data_in;
                rsp_seq_item.overflow=FIFO_Vif.overflow;
                rsp_seq_item.underflow=FIFO_Vif.underflow;
                rsp_seq_item.full=FIFO_Vif.full;
                rsp_seq_item.empty=FIFO_Vif.empty;
                rsp_seq_item.wr_ack=FIFO_Vif.wr_ack;
                rsp_seq_item.almostfull=FIFO_Vif.almostfull;
                rsp_seq_item.almostempty=FIFO_Vif.almostempty;
                rsp_seq_item.data_out=FIFO_Vif.data_out;

                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase",rsp_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

Sequence Item:



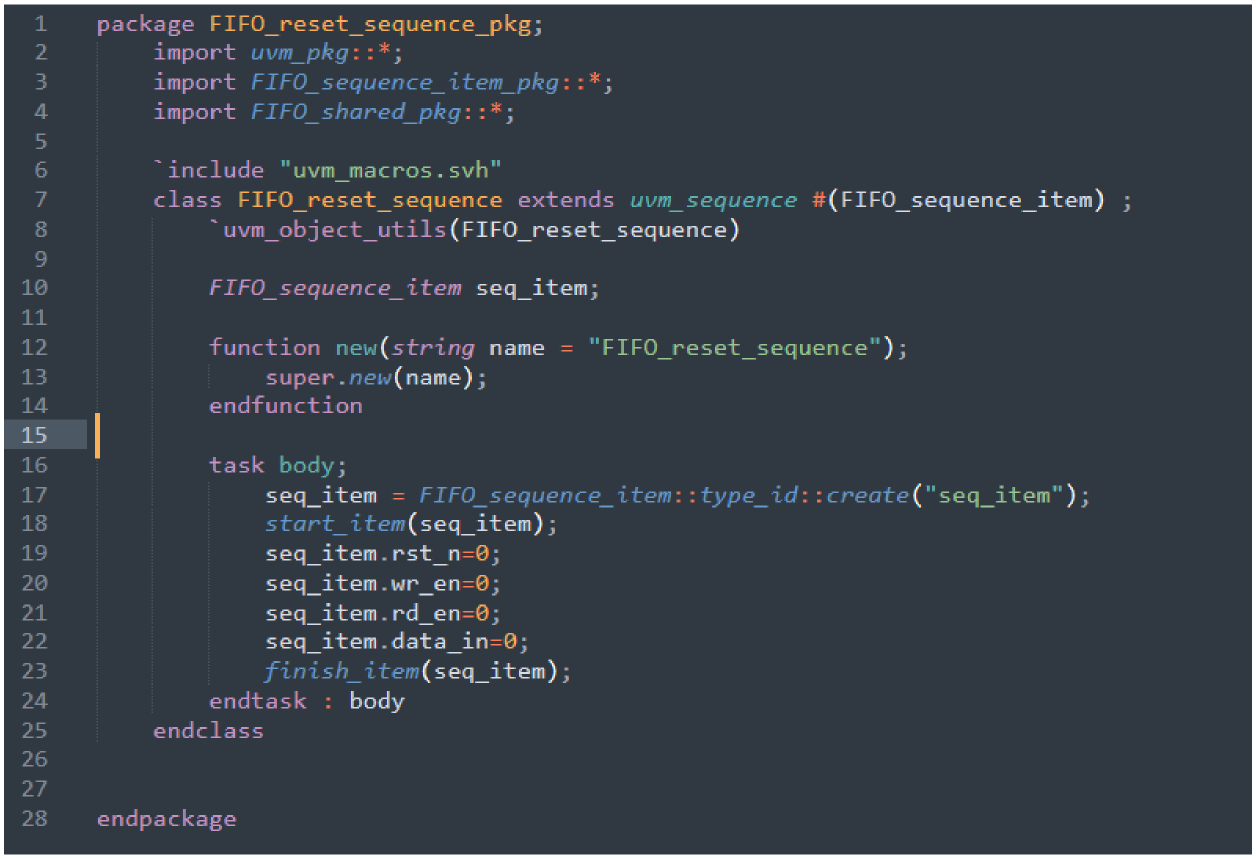
```
package FIFO_sequence_item_pkg;

    import FIFO_shared_pkg::*;
    import uvm_pkg::*;

    `include "uvm_macros.svh"
    class FIFO_sequence_item extends uvm_sequence_item ;
        `uvm_object_utils(FIFO_sequence_item)

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


        constraint  A {rst_n dist {1:/90, 0:/10};}
        constraint  B {wr_en dist {1:/70, 0:/30};}
        constraint  C {rd_en dist {1:/30, 0:/70};}

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

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

        function string convert2string_stimulus();
            return $sformatf("%s rst_n_rand = 0b%0b,  data_in = 0b%0b, wr_en = 0b%0b, rd_en = 0b%0b",
             super.convert2string(), rst_n, data_in, wr_en, rd_en);
        endfunction : convert2string_stimulus
    endclass


endpackage
```

Sequencer:



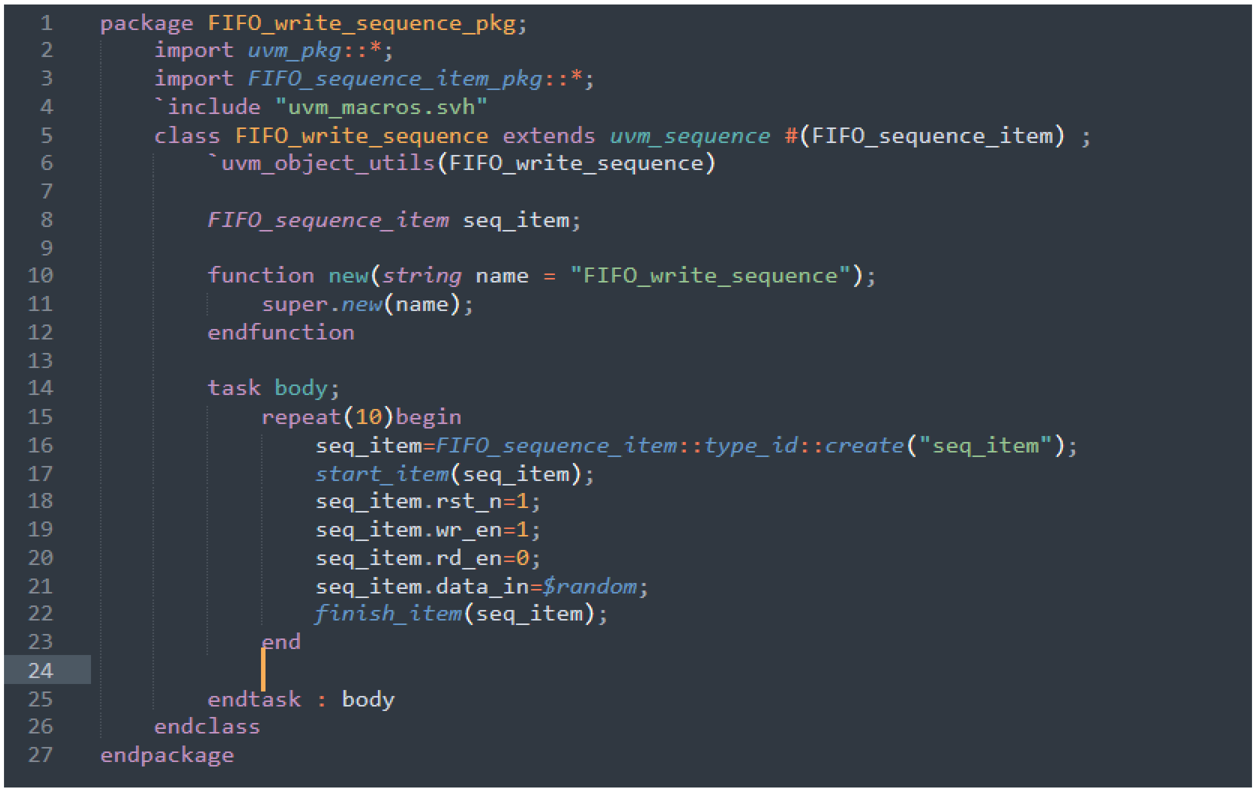
```
package MySequencer_pkg;
import FIFO_sequence_item_pkg::*;

import uvm_pkg::*;
`include "uvm_macros.svh"
class MySequencer extends uvm_sequencer #(FIFO_sequence_item);
    `uvm_component_utils(MySequencer)

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

endclass : MySequencer
endpackage : MySequencer_pkg
```

Reset sequence:



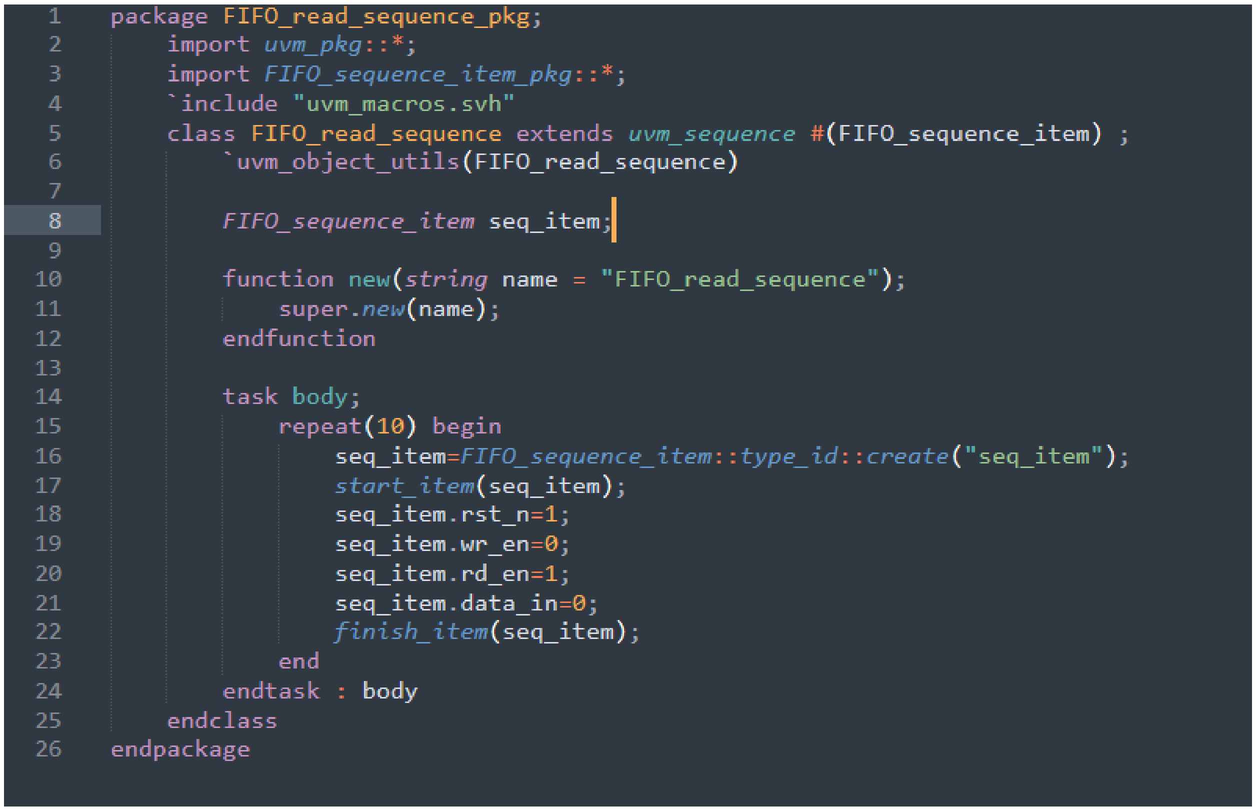
```
package FIFO_reset_sequence_pkg;
    import uvm_pkg::*;
    import FIFO_sequence_item_pkg::*;
    import FIFO_shared_pkg::*;

    `include "uvm_macros.svh"
    class FIFO_reset_sequence extends uvm_sequence #(FIFO_sequence_item) ;
        `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=0;
            seq_item.wr_en=0;
            seq_item.rd_en=0;
            seq_item.data_in=0;
            finish_item(seq_item);
        endtask : body
    endclass


endpackage
```

Write only sequence:



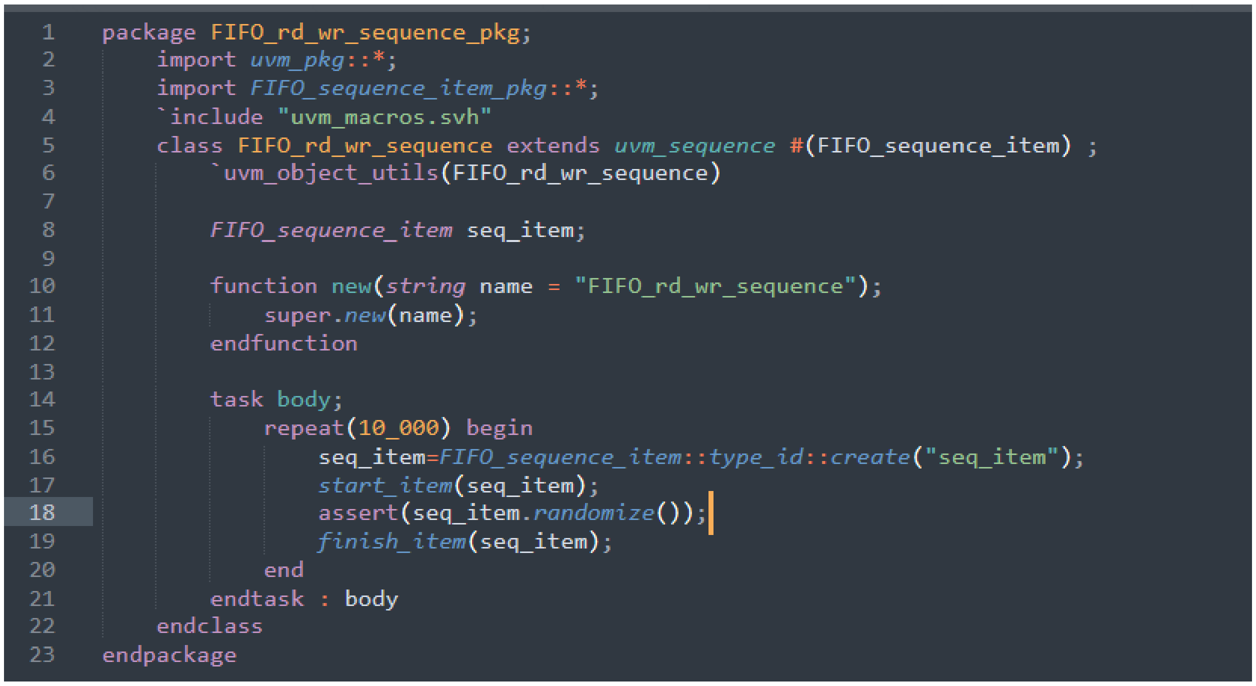
```systemverilog
package FIFO_write_sequence_pkg;
    import uvm_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_write_sequence extends uvm_sequence #(FIFO_sequence_item) ;
        `uvm_object_utils(FIFO_write_sequence)

        FIFO_sequence_item seq_item;

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

        task body;
            repeat(10)begin
                seq_item=FIFO_sequence_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.data_in=$random;
                finish_item(seq_item);
            end

        endtask : body
    endclass
endpackage
```

Read only sequence:



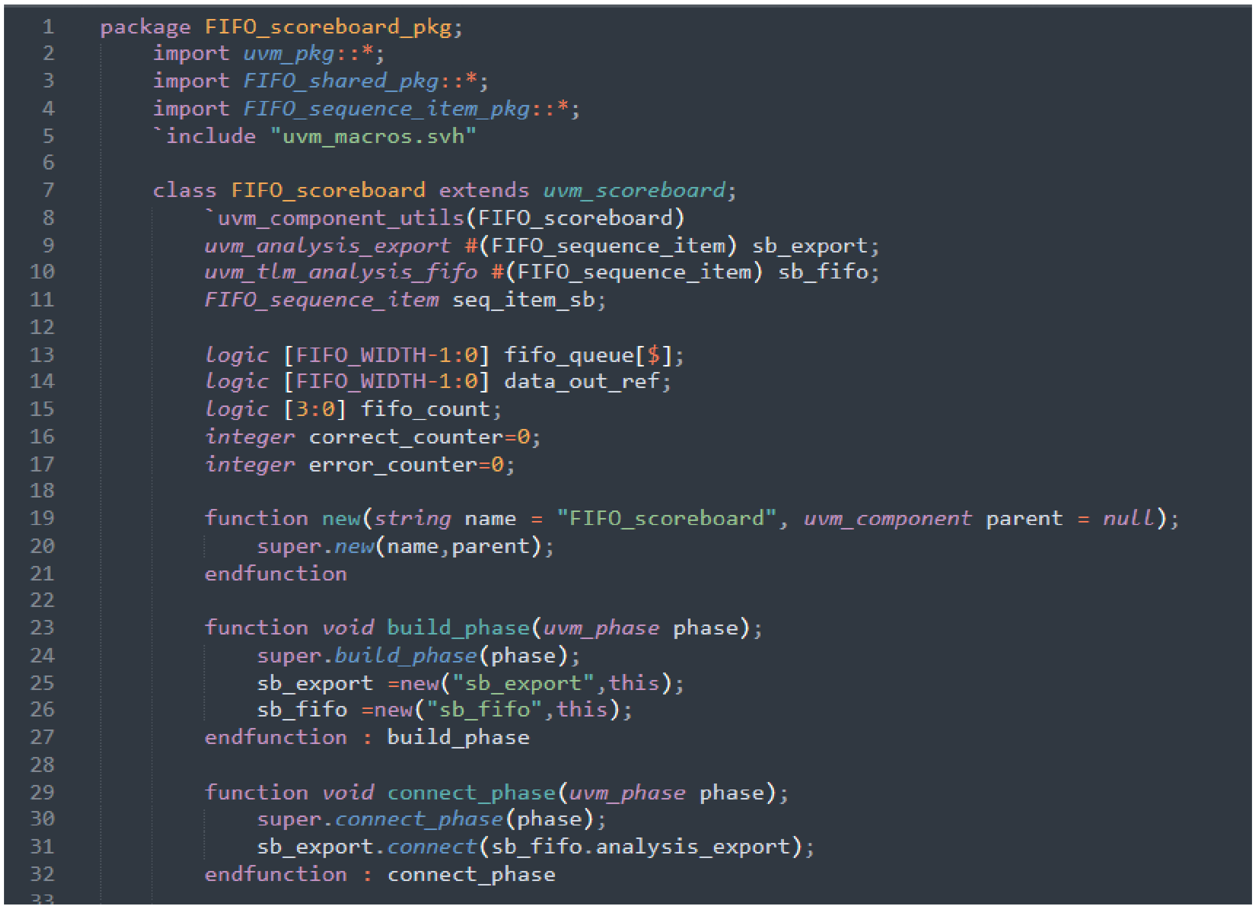
```systemverilog
package FIFO_read_sequence_pkg;
    import uvm_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_read_sequence extends uvm_sequence #(FIFO_sequence_item) ;
        `uvm_object_utils(FIFO_read_sequence)

        FIFO_sequence_item seq_item;

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

        task body;
            repeat(10) begin
                seq_item=FIFO_sequence_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.data_in=0;
                finish_item(seq_item);
            end
        endtask : body
    endclass
endpackage
```

Main Sequence:

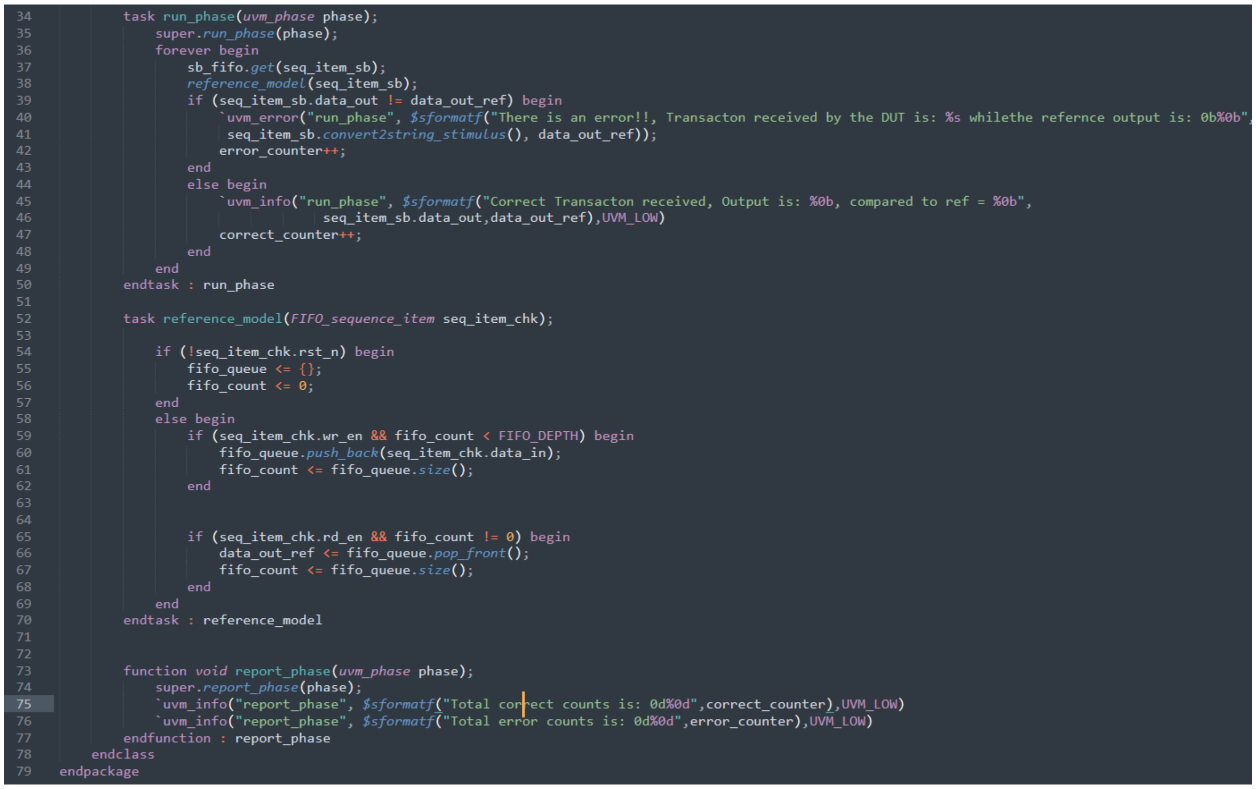


```
package FIFO_rd_wr_sequence_pkg;
    import uvm_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_rd_wr_sequence extends uvm_sequence #(FIFO_sequence_item) ;
        `uvm_object_utils(FIFO_rd_wr_sequence)

        FIFO_sequence_item seq_item;

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

        task body;
            repeat(10_000) begin
                seq_item=FIFO_sequence_item::type_id::create("seq_item");
                start_item(seq_item);
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body
    endclass
endpackage
```

Scoreboard:

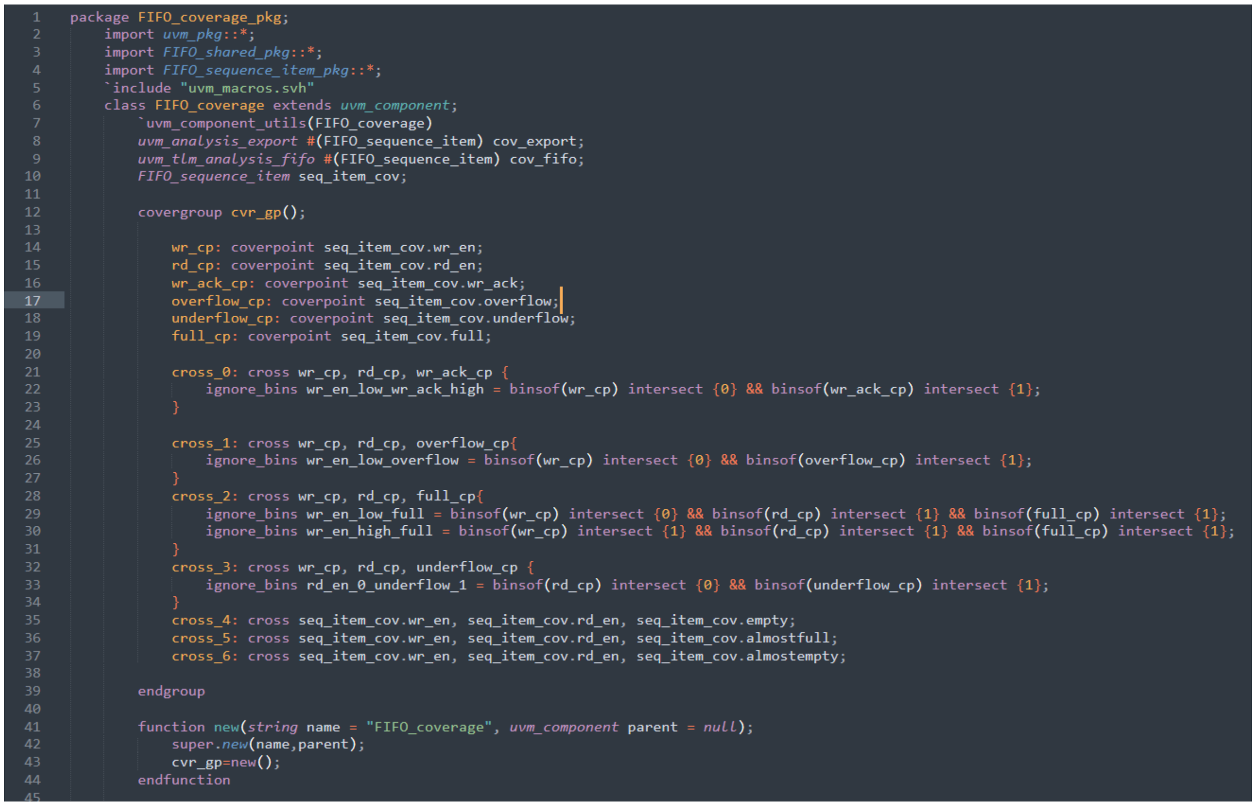


```
package FIFO_scoreboard_pkg;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh"

    class FIFO_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(FIFO_scoreboard)
        uvm_analysis_export #(FIFO_sequence_item) sb_export;
        uvm_tlm_analysis_fifo #(FIFO_sequence_item) sb_fifo;
        FIFO_sequence_item seq_item_sb;

        logic [FIFO_WIDTH-1:0] fifo_queue[$];
        logic [FIFO_WIDTH-1:0] data_out_ref;
        logic [3:0] fifo_count;
        integer correct_counter=0;
        integer error_counter=0;

        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 : build_phase

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

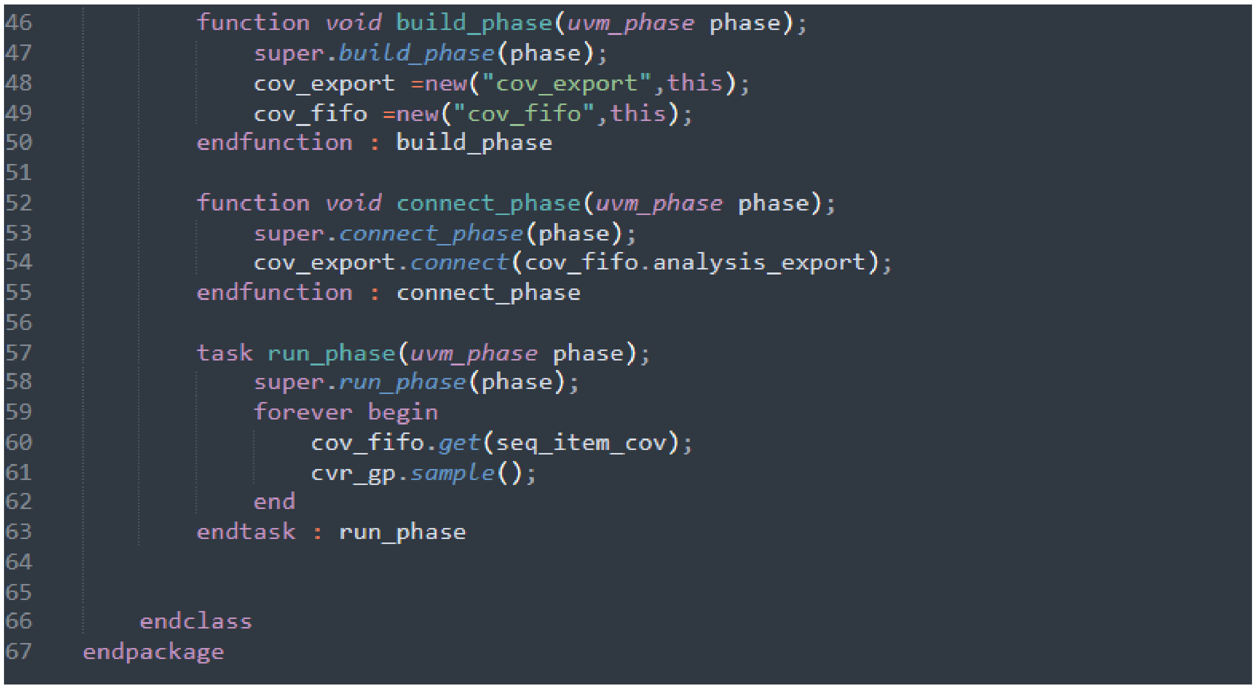


```
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                reference_model(seq_item_sb);
                if (seq_item_sb.data_out != data_out_ref) begin
                    `uvm_error("run_phase", $sformatf("There is an error!!, Transacton received by the DUT is: %s whilethe refernce output is: 0b%0b",
                     seq_item_sb.convert2string_stimulus(), data_out_ref));
                    error_counter++;
                end
                else begin
                    `uvm_info("run_phase", $sformatf("Correct Transacton received, Output is: %0b, compared to ref = %0b",
                                seq_item_sb.data_out,data_out_ref),UVM_LOW)
                    correct_counter++;
                end
            end
        endtask : run_phase

        task reference_model(FIFO_sequence_item seq_item_chk);

            if (!seq_item_chk.rst_n) begin
                fifo_queue <= {};
                fifo_count <= 0;
            end
            else begin
                if (seq_item_chk.wr_en && fifo_count < FIFO_DEPTH) begin
                    fifo_queue.push_back(seq_item_chk.data_in);
                    fifo_count <= fifo_queue.size();
                end


                if (seq_item_chk.rd_en && fifo_count != 0) begin
                    data_out_ref <= fifo_queue.pop_front();
                    fifo_count <= fifo_queue.size();
                end
            end
        endtask : reference_model


        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Total correct counts is: 0d%0d",correct_counter),UVM_LOW)
            `uvm_info("report_phase", $sformatf("Total error counts is: 0d%0d",error_counter),UVM_LOW)
        endfunction : report_phase
    endclass
endpackage
```

Coverage collector:

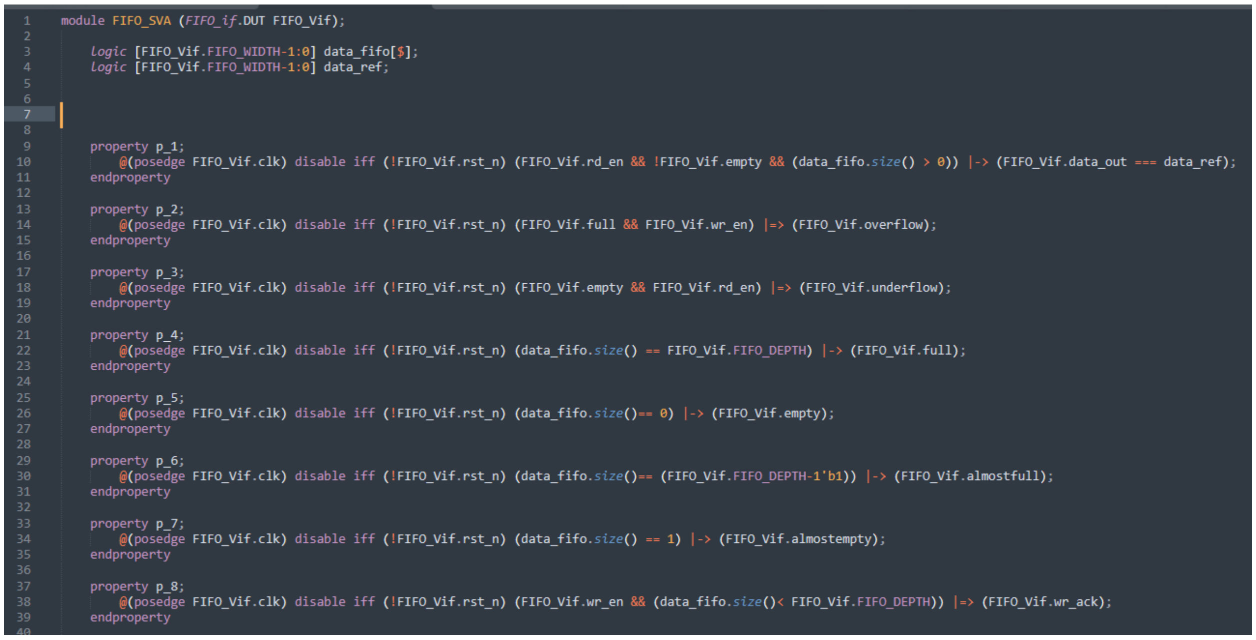


```
package FIFO_coverage_pkg;
    import uvm_pkg::*;
    import FIFO_shared_pkg::*;
    import FIFO_sequence_item_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_coverage extends uvm_component;
        `uvm_component_utils(FIFO_coverage)
        uvm_analysis_export #(FIFO_sequence_item) cov_export;
        uvm_tlm_analysis_fifo #(FIFO_sequence_item) cov_fifo;
        FIFO_sequence_item seq_item_cov;

        covergroup cvr_gp();

            wr_cp: coverpoint seq_item_cov.wr_en;
            rd_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;

            cross_0: cross wr_cp, rd_cp, wr_ack_cp {
                ignore_bins wr_en_low_wr_ack_high = binsof(wr_cp) intersect {0} && binsof(wr_ack_cp) intersect {1};
            }

            cross_1: cross wr_cp, rd_cp, overflow_cp{
                ignore_bins wr_en_low_overflow = binsof(wr_cp) intersect {0} && binsof(overflow_cp) intersect {1};
            }
            cross_2: cross wr_cp, rd_cp, full_cp{
                ignore_bins wr_en_low_full = binsof(wr_cp) intersect {0} && binsof(rd_cp) intersect {1} && binsof(full_cp) intersect {1};
                ignore_bins wr_en_high_full = binsof(wr_cp) intersect {1} && binsof(rd_cp) intersect {1} && binsof(full_cp) intersect {1};
            }
            cross_3: cross wr_cp, rd_cp, underflow_cp {
                ignore_bins rd_en_0_underflow_1 = binsof(rd_cp) intersect {0} && binsof(underflow_cp) intersect {1};
            }
            cross_4: cross seq_item_cov.wr_en, seq_item_cov.rd_en, seq_item_cov.empty;
            cross_5: cross seq_item_cov.wr_en, seq_item_cov.rd_en, seq_item_cov.almostfull;
            cross_6: cross seq_item_cov.wr_en, seq_item_cov.rd_en, seq_item_cov.almostempty;

        endgroup

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

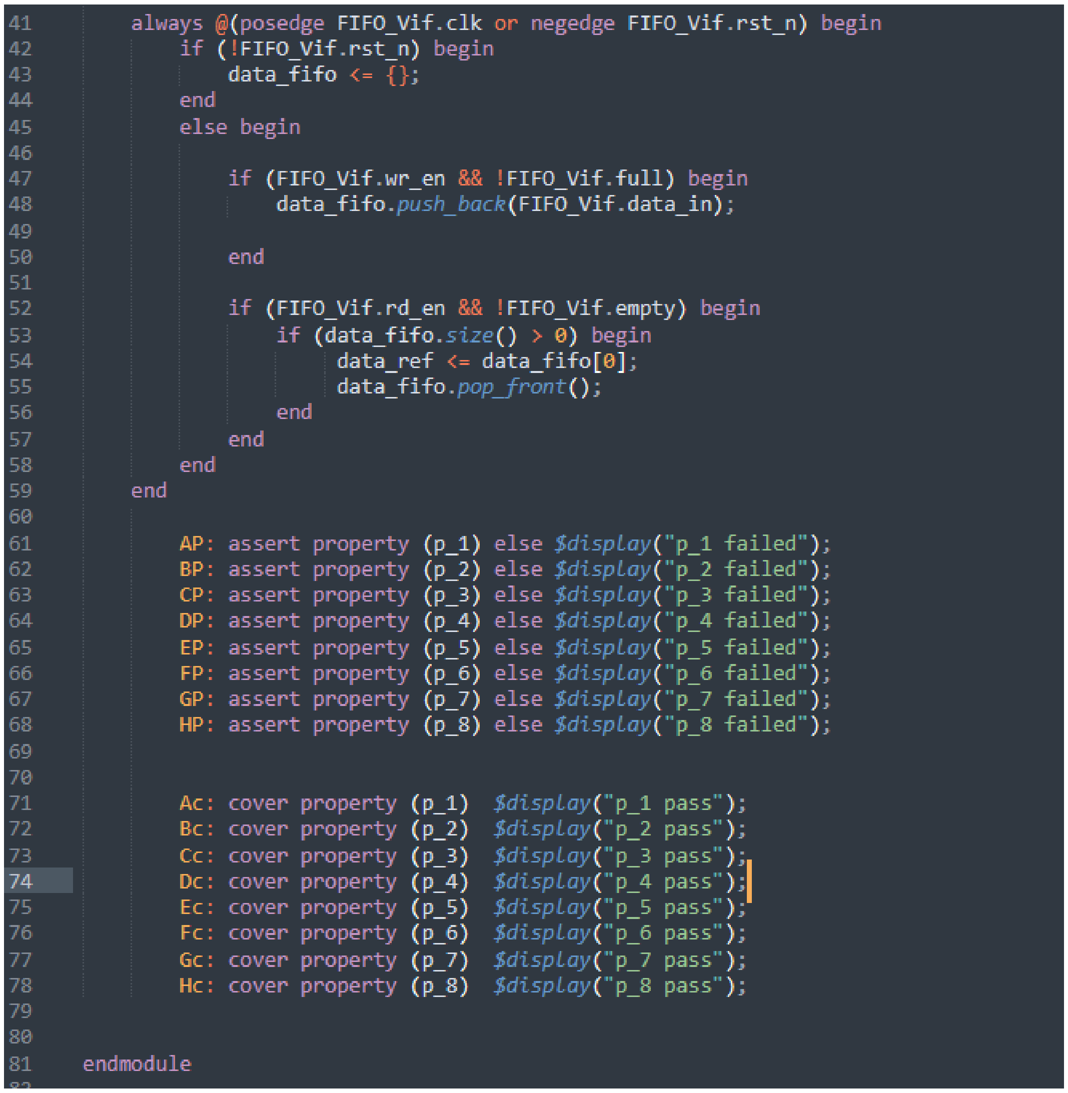


```
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            cov_export =new("cov_export",this);
            cov_fifo =new("cov_fifo",this);
        endfunction : build_phase

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item_cov);
                cvr_gp.sample();
            end
        endtask : run_phase


    endclass
endpackage
```

SVA:

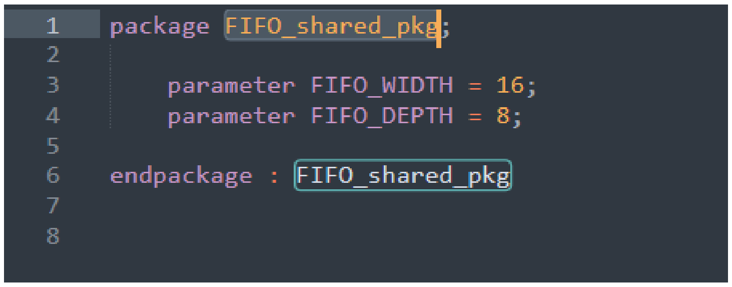


```
module FIFO_SVA (FIFO_if.DUT FIFO_Vif);

    logic [FIFO_Vif.FIFO_WIDTH-1:0] data_fifo[$];
    logic [FIFO_Vif.FIFO_WIDTH-1:0] data_ref;




    property p_1;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (FIFO_Vif.rd_en && !FIFO_Vif.empty && (data_fifo.size() > 0)) |-> (FIFO_Vif.data_out === data_ref);
    endproperty

    property p_2;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (FIFO_Vif.full && FIFO_Vif.wr_en) |=> (FIFO_Vif.overflow);
    endproperty

    property p_3;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (FIFO_Vif.empty && FIFO_Vif.rd_en) |=> (FIFO_Vif.underflow);
    endproperty

    property p_4;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (data_fifo.size() == FIFO_Vif.FIFO_DEPTH) |-> (FIFO_Vif.full);
    endproperty

    property p_5;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (data_fifo.size()== 0) |-> (FIFO_Vif.empty);
    endproperty

    property p_6;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (data_fifo.size()== (FIFO_Vif.FIFO_DEPTH-1'b1)) |-> (FIFO_Vif.almostfull);
    endproperty

    property p_7;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (data_fifo.size() == 1) |-> (FIFO_Vif.almostempty);
    endproperty

    property p_8;
        @(posedge FIFO_Vif.clk) disable iff (!FIFO_Vif.rst_n) (FIFO_Vif.wr_en && (data_fifo.size()< FIFO_Vif.FIFO_DEPTH)) |=> (FIFO_Vif.wr_ack);
    endproperty
```



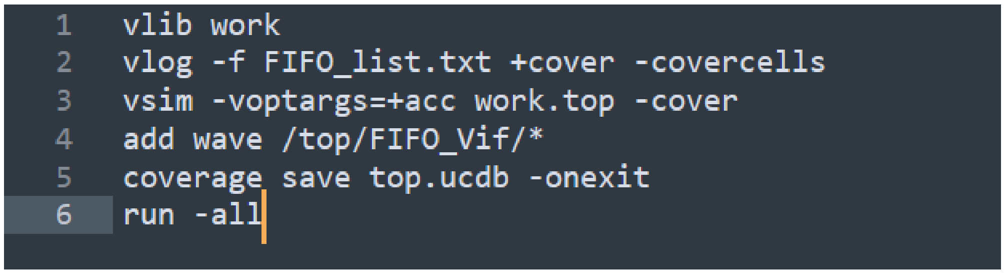
```
    always @(posedge FIFO_Vif.clk or negedge FIFO_Vif.rst_n) begin
        if (!FIFO_Vif.rst_n) begin
            data_fifo <= {};
        end
        else begin

            if (FIFO_Vif.wr_en && !FIFO_Vif.full) begin
                data_fifo.push_back(FIFO_Vif.data_in);

            end

            if (FIFO_Vif.rd_en && !FIFO_Vif.empty) begin
                if (data_fifo.size() > 0) begin
                     data_ref <= data_fifo[0];
                     data_fifo.pop_front();
                end
            end
        end
    end

        AP: assert property (p_1) else $display("p_1 failed");
        BP: assert property (p_2) else $display("p_2 failed");
        CP: assert property (p_3) else $display("p_3 failed");
        DP: assert property (p_4) else $display("p_4 failed");
        EP: assert property (p_5) else $display("p_5 failed");
        FP: assert property (p_6) else $display("p_6 failed");
        GP: assert property (p_7) else $display("p_7 failed");
        HP: assert property (p_8) else $display("p_8 failed");


        Ac: cover property (p_1)  $display("p_1 pass");
        Bc: cover property (p_2)  $display("p_2 pass");
        Cc: cover property (p_3)  $display("p_3 pass");
        Dc: cover property (p_4)  $display("p_4 pass");
        Ec: cover property (p_5)  $display("p_5 pass");
        Fc: cover property (p_6)  $display("p_6 pass");
        Gc: cover property (p_7)  $display("p_7 pass");
        Hc: cover property (p_8)  $display("p_8 pass");


endmodule
```

Shared package:



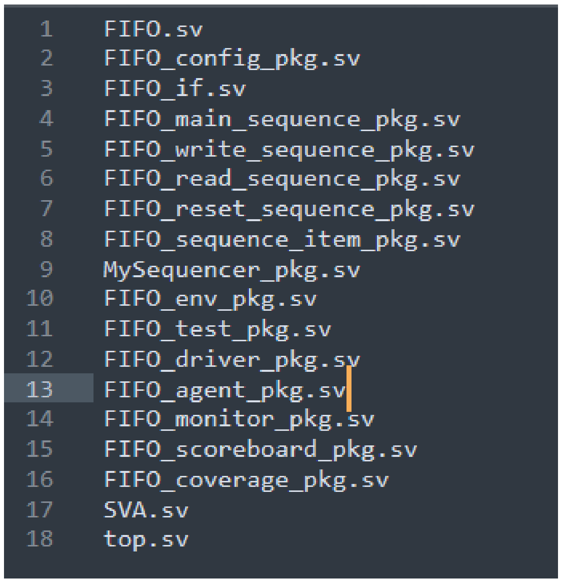
```
package FIFO_shared_pkg;

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

endpackage : FIFO_shared_pkg
```

Do:



```
vlib work
vlog -f FIFO_list.txt +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave /top/FIFO_Vif/*
coverage save top.ucdb -onexit
run -all
```

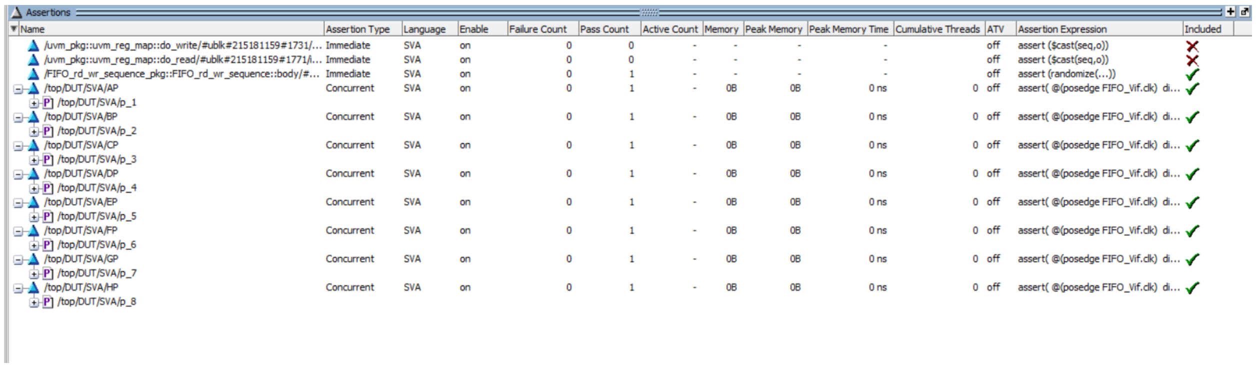
File:



```
FIFO.sv
FIFO_config_pkg.sv
FIFO_if.sv
FIFO_main_sequence_pkg.sv
FIFO_write_sequence_pkg.sv
FIFO_read_sequence_pkg.sv
FIFO_reset_sequence_pkg.sv
FIFO_sequence_item_pkg.sv
MySequencer_pkg.sv
FIFO_env_pkg.sv
FIFO_test_pkg.sv
FIFO_driver_pkg.sv
FIFO_agent_pkg.sv
FIFO_monitor_pkg.sv
FIFO_scoreboard_pkg.sv
FIFO_coverage_pkg.sv
SVA.sv
top.sv
```

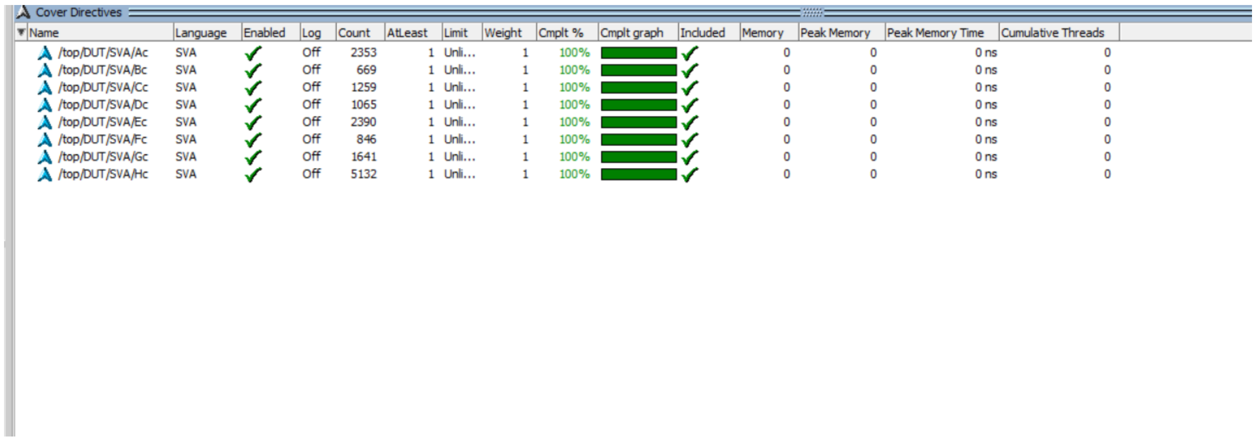
Code & Functional coverages:

Sequential domain coverage report:



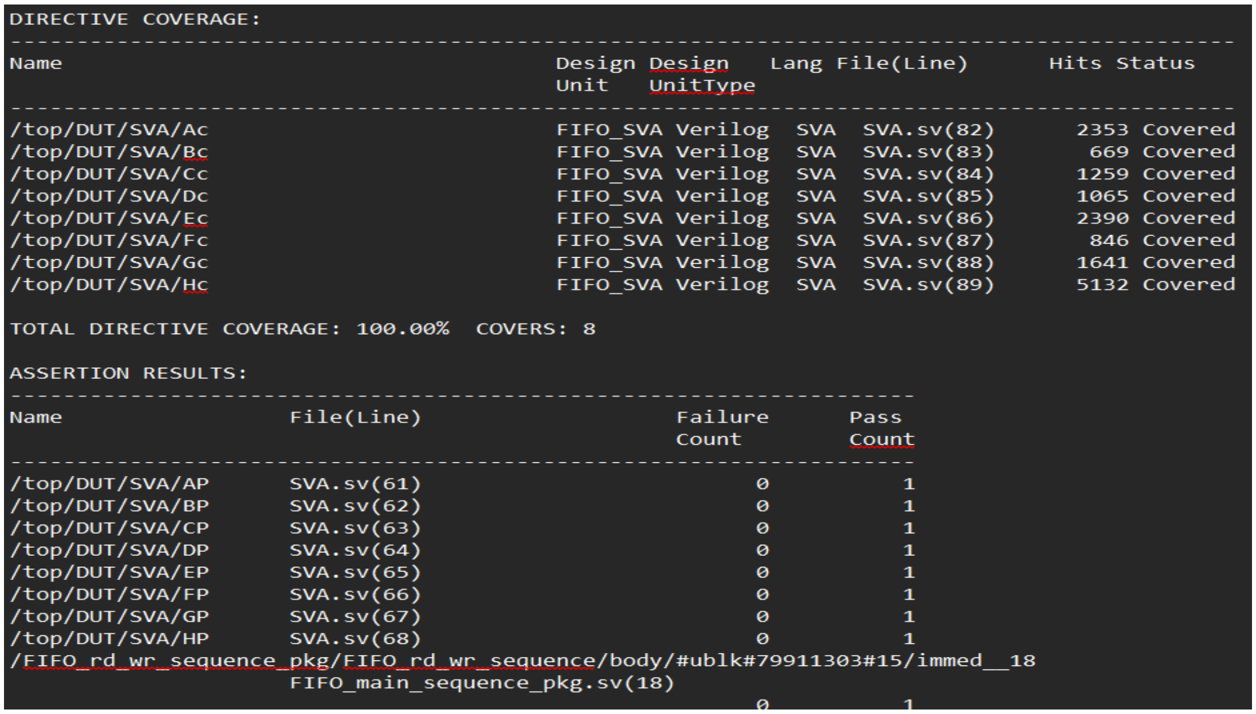
Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/i...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/FIFO_rd_wr_sequence_pkg::FIFO_rd_wr_sequence::body/#...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/DUT/SVA/AP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_1													
/top/DUT/SVA/BP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_2													
/top/DUT/SVA/CP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_3													
/top/DUT/SVA/DP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_4													
/top/DUT/SVA/EP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_5													
/top/DUT/SVA/FP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_6													
/top/DUT/SVA/GP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_7													
/top/DUT/SVA/HP	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_Vif.clk) di...	✓
/top/DUT/SVA/p_8													

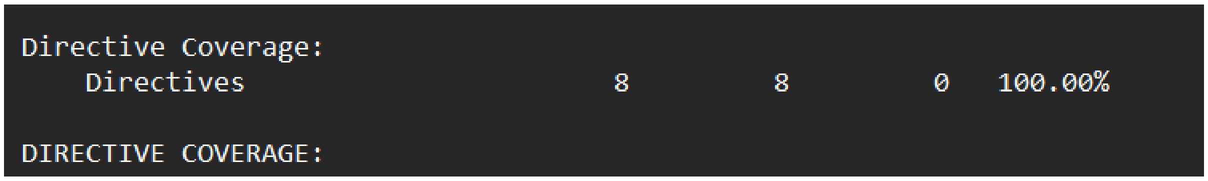


Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/SVA/Ac	SVA	✓	Off	2353	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Bc	SVA	✓	Off	669	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Cc	SVA	✓	Off	1259	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Dc	SVA	✓	Off	1065	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Ec	SVA	✓	Off	2390	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Fc	SVA	✓	Off	846	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Gc	SVA	✓	Off	1641	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SVA/Hc	SVA	✓	Off	5132	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0

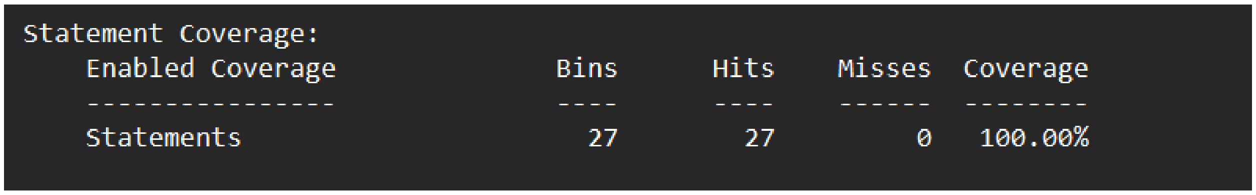


```
DIRECTIVE COVERAGE:
------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
------------------------------------------------------------------------------
/top/DUT/SVA/Ac                          FIFO_SVA Verilog  SVA  SVA.sv(82)      2353 Covered
/top/DUT/SVA/Bc                          FIFO_SVA Verilog  SVA  SVA.sv(83)       669 Covered
/top/DUT/SVA/Cc                          FIFO_SVA Verilog  SVA  SVA.sv(84)      1259 Covered
/top/DUT/SVA/Dc                          FIFO_SVA Verilog  SVA  SVA.sv(85)      1065 Covered
/top/DUT/SVA/Ec                          FIFO_SVA Verilog  SVA  SVA.sv(86)      2390 Covered
/top/DUT/SVA/Fc                          FIFO_SVA Verilog  SVA  SVA.sv(87)       846 Covered
/top/DUT/SVA/Gc                          FIFO_SVA Verilog  SVA  SVA.sv(88)      1641 Covered
/top/DUT/SVA/Hc                          FIFO_SVA Verilog  SVA  SVA.sv(89)      5132 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 8

ASSERTION RESULTS:
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUT/SVA/AP      SVA.sv(61)                         0          1
/top/DUT/SVA/BP      SVA.sv(62)                         0          1
/top/DUT/SVA/CP      SVA.sv(63)                         0          1
/top/DUT/SVA/DP      SVA.sv(64)                         0          1
/top/DUT/SVA/EP      SVA.sv(65)                         0          1
/top/DUT/SVA/FP      SVA.sv(66)                         0          1
/top/DUT/SVA/GP      SVA.sv(67)                         0          1
/top/DUT/SVA/HP      SVA.sv(68)                         0          1
/FIFO_rd_wr_sequence_pkg/FIFO_rd_wr_sequence/body/#ublk#79911303#15/immed__18
                     FIFO_main_sequence_pkg.sv(18)
                                                        0          1
```

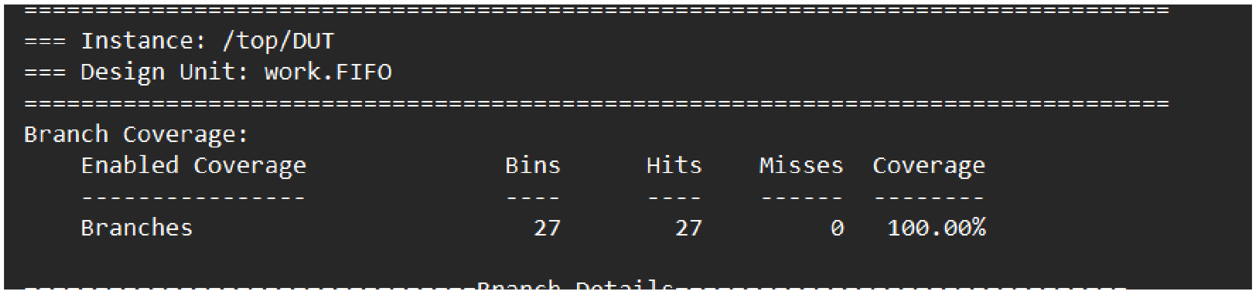


```
Directive Coverage:
    Directives                    8         8         0    100.00%

DIRECTIVE COVERAGE:
```

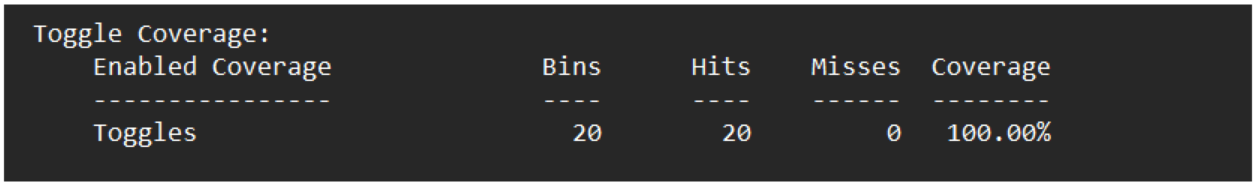
Code coverage:



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

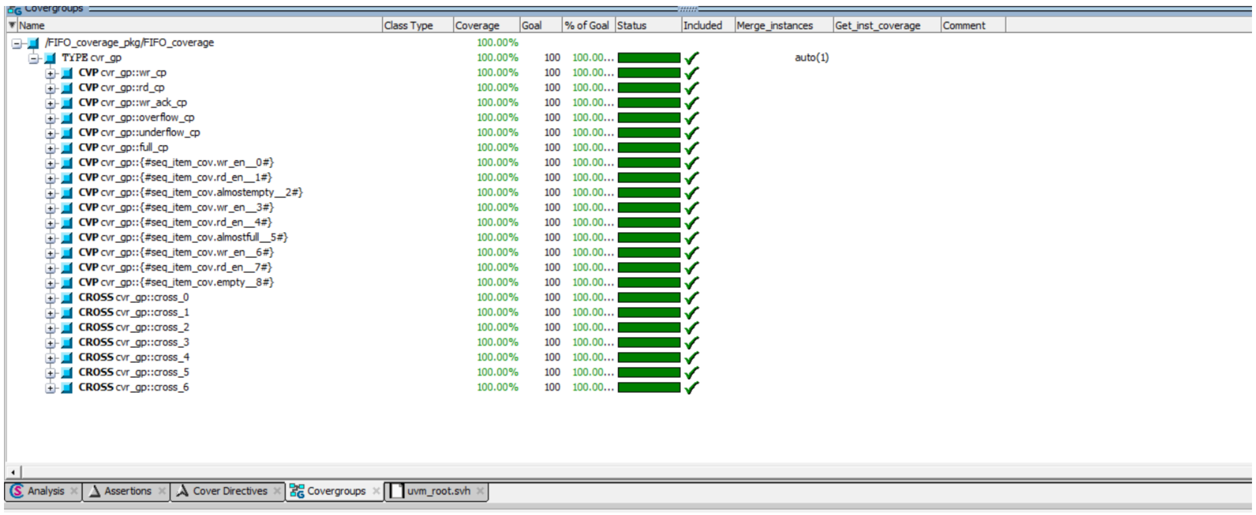


```
=========================================================================
=== Instance: /top/DUT
=== Design Unit: work.FIFO
=========================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------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

------------------------------------Branch Details------------------------------------
```



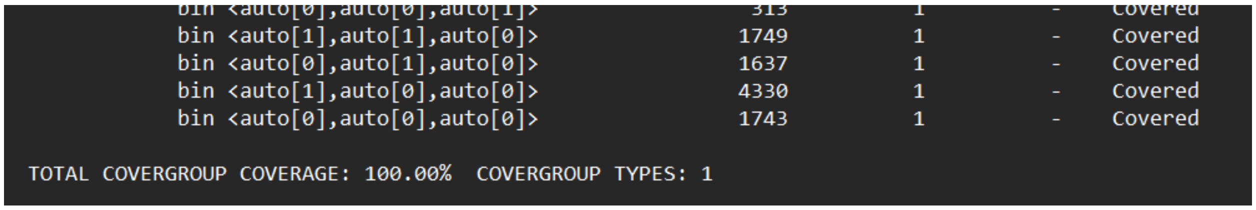
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         20        20         0   100.00%
```

Functional coverage:



Covergroups

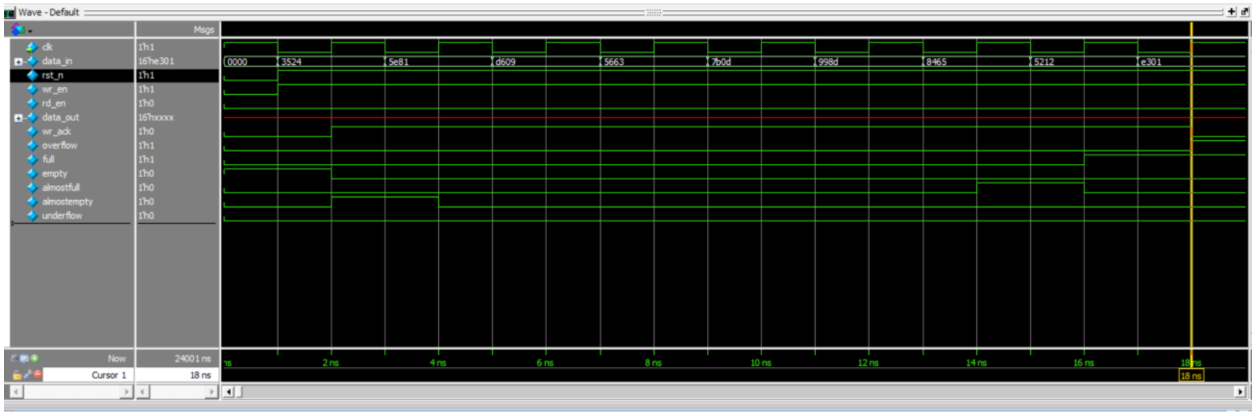
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_coverage_pkg/FIFO_coverage		100.00%							
TYPE cvr_gp		100.00%	100	100.00...	🟩	✓	auto(1)		
CVP cvr_gp::wr_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::rd_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::wr_ack_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::overflow_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::underflow_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::full_cp		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.wr_en__0#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.rd_en__1#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.almostempty__2#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.wr_en__3#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.rd_en__4#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.almostfull__5#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.wr_en__6#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.rd_en__7#}		100.00%	100	100.00...	🟩	✓			
CVP cvr_gp::{#seq_item_cov.empty__8#}		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_0		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_1		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_2		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_3		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_4		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_5		100.00%	100	100.00...	🟩	✓			
CROSS cvr_gp::cross_6		100.00%	100	100.00...	🟩	✓			



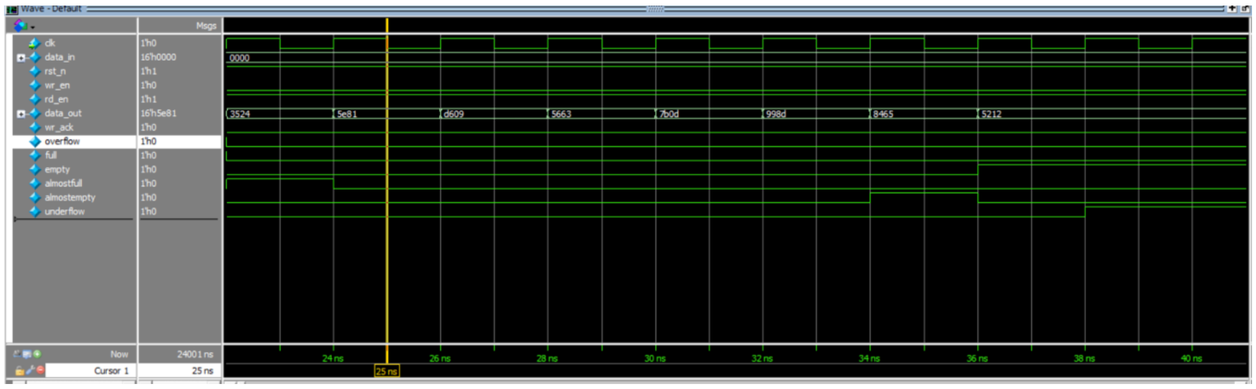
```
            bin <auto[0],auto[0],auto[1]>          313         1          -    Covered
            bin <auto[1],auto[1],auto[0]>         1749         1          -    Covered
            bin <auto[0],auto[1],auto[0]>         1637         1          -    Covered
            bin <auto[1],auto[0],auto[0]>         4330         1          -    Covered
            bin <auto[0],auto[0],auto[0]>         1743         1          -    Covered

 TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

Simulations:

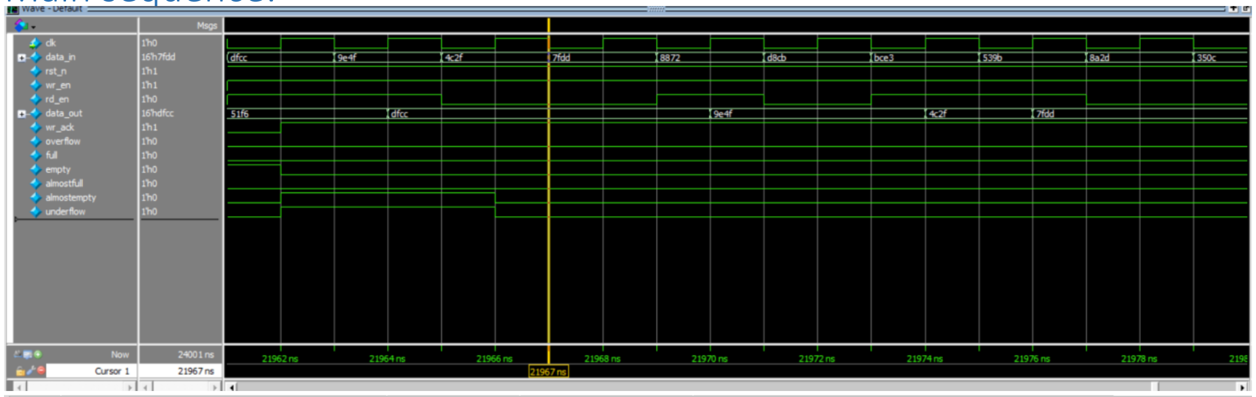
Write only sequence:

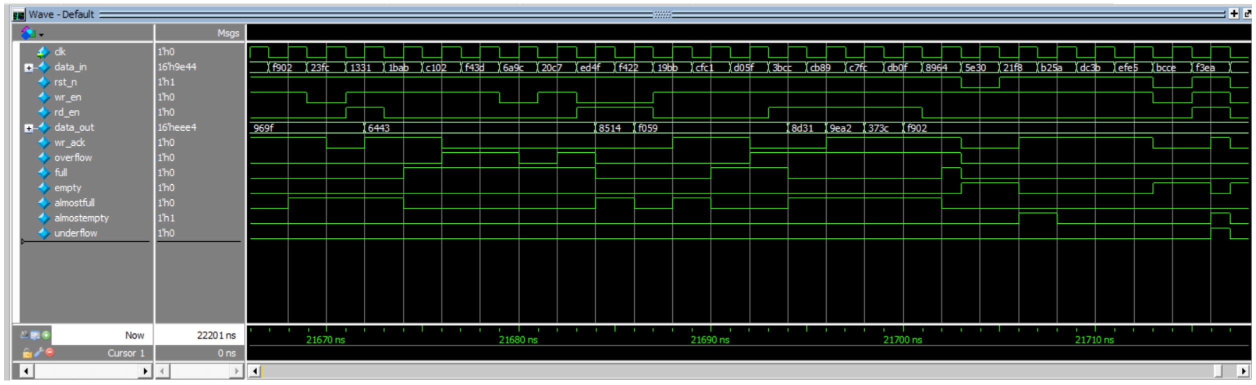


Read only sequence:



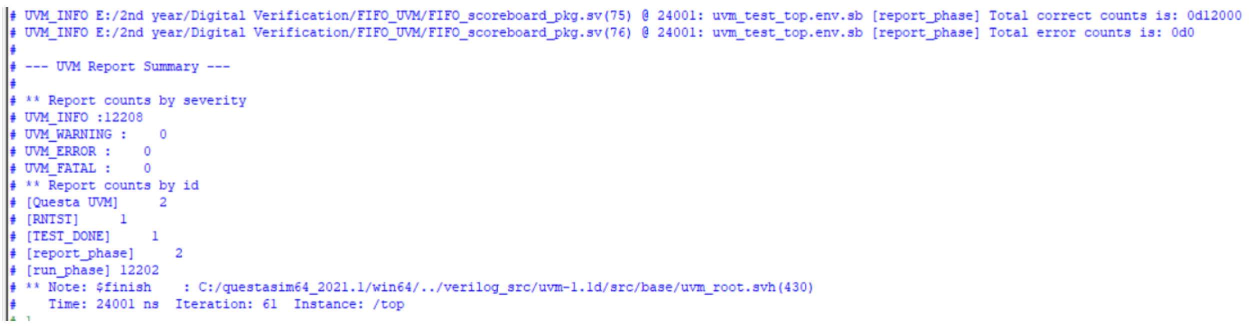
Main sequence:





Transcripts:

```
# p_1 pass
# p_6 pass
# p_8 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22164: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 11000110000100, compared to ref = 11000110000100
# p_6 pass
# p_8 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22166: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 11000110000100, compared to ref = 11000110000100
# p_4 pass
# p_8 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22168: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 11000110000100, compared to ref = 11000110000100
# p_1 pass
# p_2 pass
# p_4 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22170: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1010101101110110, compared to ref = 1010101101110110
# p_6 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22172: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1010101101110110, compared to ref = 1010101101110110
# p_4 pass
# p_8 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22174: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1010101101110110, compared to ref = 1010101101110110
# p_2 pass
# p_4 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22176: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1010101101110110, compared to ref = 1010101101110110
# p_2 pass
# p_4 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22178: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1010101101110110, compared to ref = 1010101101110110
# p_1 pass
# p_2 pass
# p_4 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22180: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1000101000110001, compared to ref = 1000101000110001
# p_2 pass
# p_6 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22182: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1000101000110001, compared to ref = 1000101000110001
# p_4 pass
# p_8 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22184: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1000101000110001, compared to ref = 1000101000110001
# p_4 pass
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(45) @ 22186: uvm_test_top.env.sb [run_phase] Correct Transacton received, Output is: 1000101000110001, compared to ref = 1000101000110001
```



```
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(75) @ 24001: uvm_test_top.env.sb [report_phase] Total correct counts is: 0d12000
# UVM_INFO E:/2nd year/Digital Verification/FIFO_UVM/FIFO_scoreboard_pkg.sv(76) @ 24001: uvm_test_top.env.sb [report_phase] Total error counts is: 0d0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :12208
# 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] 12202
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 24001 ns  Iteration: 61  Instance: /top
```