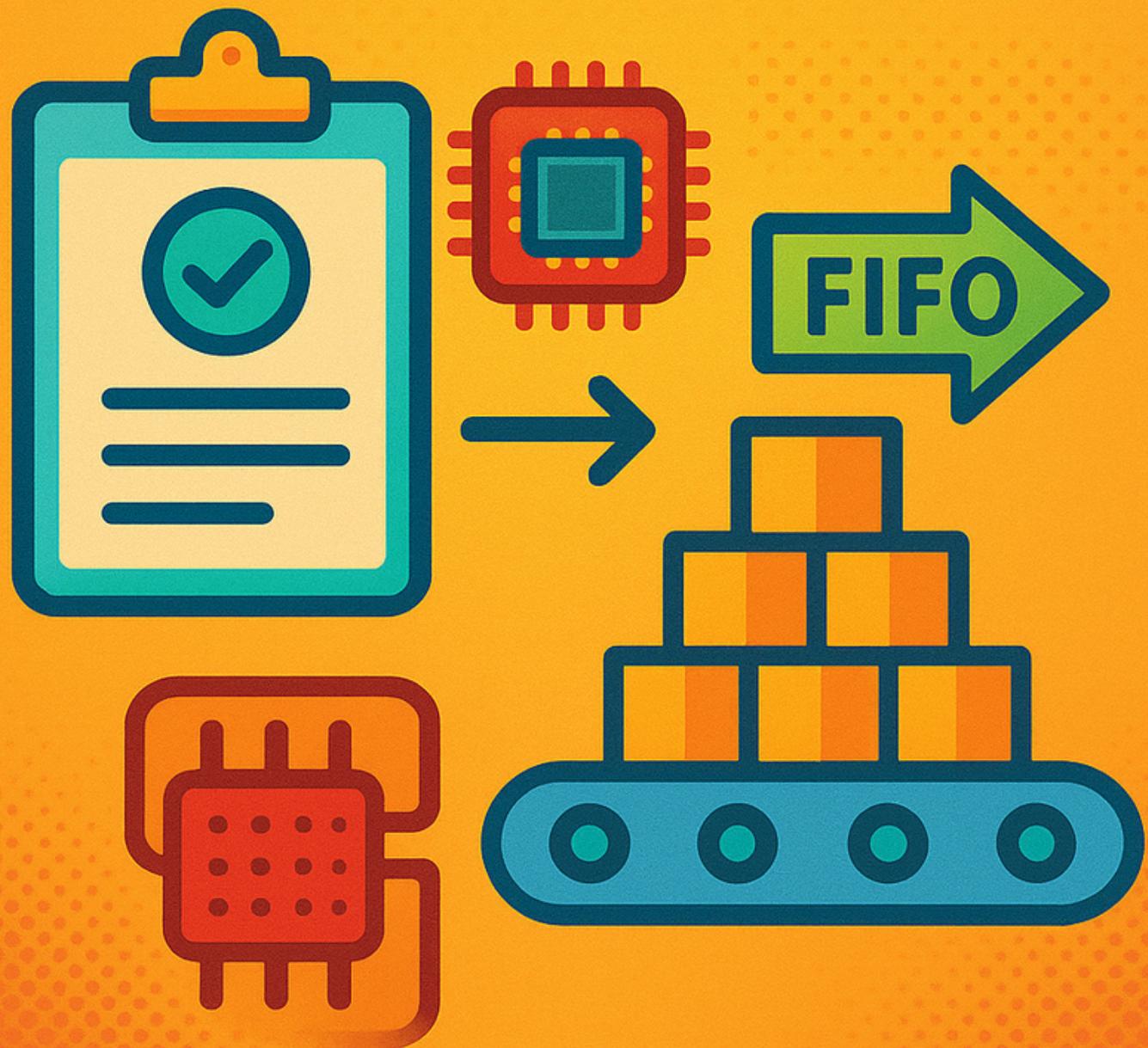


# FIFO VERIFICATION PROJECT



Made by: Ziad Kassem  
Under the supervision of Eng. Kareem Wassem

## Contents:

Introduction: .....	3
Verification Objectives:.....	3
Simulation Output: .....	4
Bugs Found:.....	5
In Write Operation : .....	5
In Read Operation: .....	5
In Count Operation: .....	5
In Combinational Flags: .....	5
Design Code After Modifications: .....	6
UVM Architecture Overview.....	8
Key UVM Components:.....	8
Configuration (FIFO_config).....	8
Sequence Item (FIFO_seq_item).....	8
Sequences (FIFO_sequence).....	9
Sequencer (FIFO_sequencer) .....	9
Driver (FIFO_driver).....	9
Monitor (FIFO_monitor).....	9
Agent (FIFO_agent) .....	9
Scoreboard (FIFO_scoreboard).....	9
Coverage (FIFO_coverage).....	10
Environment (FIFO_env).....	10
Test (FIFO_test).....	10
UVM Verification Flow .....	10
Configuration Phase: .....	10
Build Phase:.....	10
Connect Phase:.....	10
Run Phase: .....	10
Report Phase: .....	11
Assertion-Based Verification .....	11
Conclusion.....	11
Shared Package: .....	12

Interface: .....	12
Top Module: .....	12
Test Component: .....	13
Configuration Object: .....	14
Environment Component: .....	15
Sequence Item Object: .....	16
Sequencer Object: .....	17
Sequence Object : .....	18
Agent Component : .....	21
Driver Component : .....	22
Monitor Component: .....	23
Scoreboard Component: .....	24
Coverage Components: .....	26
Assertions Table : .....	28
SystemVerilog Assertions File:.....	29
Waveforms: .....	32
FIFO_1: .....	32
FIFO_2: .....	32
FIFO_3: .....	33
FIFO_4: .....	33
FIFO_5: .....	34
Run.do:.....	34
Source List: .....	35
Cover Report:.....	36
Covergroups Report:.....	57

## Introduction:

The goal of this project is to verify the functional correctness of a FIFO (First-In-First-Out) buffer using the Universal Verification Methodology (**UVM**) along with **SystemVerilog Assertions (SVA)**. FIFO buffers are critical components in digital designs, ensuring ordered data transfer between modules with different clock domains or processing rates.

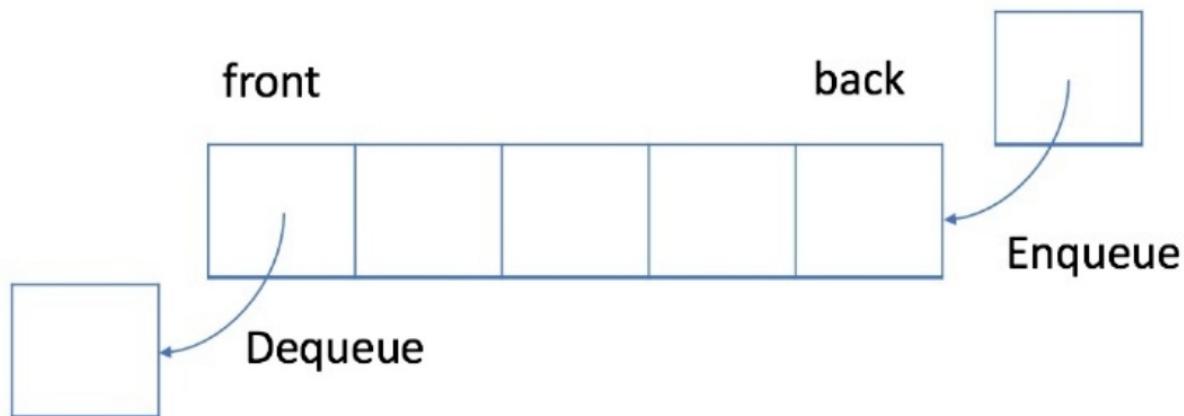
Unlike **traditional testbenches**, this project implements a structured and reusable UVM environment that supports randomized stimuli, constrained sequences, assertion-based verification, and functional/structural coverage collection. Both built-in (inline) and externally bound assertion files were used to validate design behavior, increasing modularity and reusability.

## Verification Objectives:

- The primary goals of this UVM-based verification effort include:
- Ensure the FIFO does not allow writes when full is asserted.
- Ensure the FIFO does not allow reads when empty is asserted.
- Confirm data order preservation — data must be read in the same order it was written.
- Detect and report illegal operations, such as reading when empty or writing when full.
- Verify pointer wrap-around behavior and overflow/underflow protection logic.
- Confirm wr\_ack, rd\_ack, overflow, and underflow flags behave as expected under reset and operational conditions.
- Use SystemVerilog Assertions and cover properties to ensure thorough checking of corner cases.
- Collect functional, code, and sequential domain coverage.

## Simulation Output:

- The UVM testbench includes the following components:
- UVM Sequence Items with constraints to generate valid/invalid stimuli.
- Multiple UVM Sequences to test:
- write\_only\_sequence
- read\_only\_sequence
- write\_read\_sequence
- illegal\_operation\_sequence
- Scoreboard to track data correctness.
- Coverage Collector with covergroups and coverpoints for full functional coverage.
- Assertions:
- Some are directly embedded in the design (inline).
- Others are written in external .sv files and bound using the bind mechanism in the top module.
- Bug detection & reporting mechanism.
- Detailed waveforms for each test via QuestaSim GUI.
- do file automating the simulation and waveform loading process.



## Bugs Found:

### In Write Operation :

The wr\_ack and overflow flags were not properly reset while the reset signal was asserted. Additionally, a warning may arise when handling the overflow condition: if the condition uses a bitwise operator (&) instead of a logical one (&&), it might not cause functional errors for single-bit signals, but it can lead to unintended behavior or warnings when the operands are wider than one bit.

```
//if (fifo_if.full & fifo_if.wr_en)----->Warning: this statement is not a bitwise operation  
//-> better to use && (meanwhile & will do the same job because oprands are 1 bit)
```

### In Read Operation:

The underflow flag was not properly reset while the reset signal was asserted. Moreover, the underflow condition was handled using combinational logic, which violates the specification that requires sequential handling.

```
//assign fifo_if.underflow = (fifo_if.empty && fifo_if.rd_en)? 1 : 0; ----->ERROR : This  
flag is sequential not combinational
```

### In Count Operation:

There were 2 missing conditions for count “internal signal” comparing to asked Specs

If both read and write enables are high simultaneously:

- When the FIFO is empty, only writing will take place.
- When the FIFO is full, only reading will take place.

### In Combinational Flags:

There was a single error in Almostfull flag as follows:

```
//assign fifo_if.almostfull = (count == FIFO_DEPTH-2)? 1 : 0; --->ERROR : rising the flag  
when only 1 slot Left not 2
```

## Design Code After Modifications:

```
module FIFO(FIFO_INT.DUT fifo_if);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

//Write Operation & sequential Flags handling:
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;      // all flags should be resetted
        fifo_if.overflow <= 0;    // all flags should be resetted
    end
    else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= fifo_if.data_in;
        fifo_if.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        fifo_if.overflow <= 0;
    end
    else begin
        fifo_if.wr_ack <= 0;
        if (fifo_if.full && fifo_if.wr_en)
            fifo_if.overflow <= 1;
        else begin
            fifo_if.overflow <= 0;
        end
    end
end
end

//Read operation:
always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        fifo_if.underflow <= 0; // all flags should be resetted
        rd_ptr <= 0;
    end
    else if (fifo_if.rd_en && count > 0) begin
        fifo_if.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        fifo_if.underflow <= 0; // handling Underflow sequentially
    end
    else if ( fifo_if.empty && fifo_if.rd_en)begin
        fifo_if.underflow <= 1; // handling Underflow sequentially
    end
end
```

```

        end
    end

//Counter handling:
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)
            count <= count + 1;
        else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)
            count <= count - 1;
    end
end

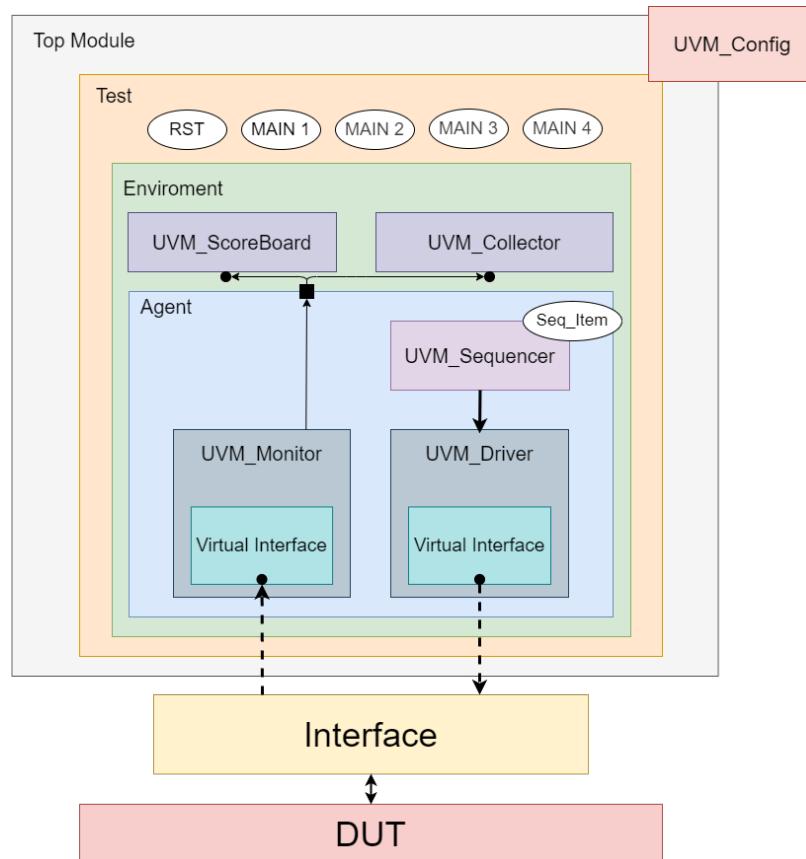
assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifo_if.empty = (count == 0)? 1 : 0;
assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
assign fifo_if.almostempty = (count == 1)? 1 : 0;

endmodule

```

# UVM Architecture Overview

The Universal Verification Methodology (UVM) provides a structured approach to verification using a layered architecture. This project implements a complete UVM verification environment for a FIFO design with the following components:



## Key UVM Components:

### Configuration (FIFO\_config)

- Acts as a custom database that stores the virtual interface and other parameters
- Provides central access to configuration data across all verification components
- Avoids duplicating interface connections throughout the testbench

### Sequence Item (FIFO\_seq\_item)

- Defines the transaction object that passes between components
- Contains all signals with randomization constraints
- Includes convert2string methods for UVM reporting

## Sequences (FIFO\_sequence)

- Defines test stimulus as sequences of transactions
- The project implements multiple sequences for different test scenarios
- Each sequence creates and randomizes sequence items before sending them to the driver

## Sequencer (FIFO\_sequencer)

- Acts as a synchronization point between sequences and the driver
- Manages the flow of sequence items using TLM connections
- Functions similar to a FIFO that organizes the transfer of transactions

## Driver (FIFO\_driver)

- Converts sequence items into signal-level activity on the DUT interface
- Implements the following algorithm:
  1. Get next sequence item
  2. Drive signals to interface
  3. Wait for clock edge
  4. Report completion and repeat

## Monitor (FIFO\_monitor)

- Observes interface signals without modifying them
- Converts signal activity into sequence items for analysis
- Broadcasts transactions through its analysis port to scoreboard and coverage components

## Agent (FIFO\_agent)

- Encapsulates the driver, sequencer, and monitor
- Creates a reusable verification component for a specific protocol
- Connects components together through TLM connections

## Scoreboard (FIFO\_scoreboard)

- Validates DUT behavior against a reference model
- Implements a FIFO reference model for comparison
- Tracks and reports error and success statistics

## Coverage (FIFO\_coverage)

- Measures verification progress using functional coverage
- Implements covergroups to track coverage of important scenarios
- Uses cross coverage to verify interesting combinations of signals

## Environment (FIFO\_env)

- Instantiates and connects the agent, scoreboard, and coverage components
- Provides a complete verification environment for the DUT
- Uses TLM to connect agent analysis port to scoreboard and coverage

## Test (FIFO\_test)

- Configures and controls the verification environment
- Sequences test scenarios by running specific sequences
- Manages the UVM phases through objections

# UVM Verification Flow

## Configuration Phase:

- Create virtual interface in top module
- Set interface in the configuration database
- Pass configuration to all components

## Build Phase:

- Create all components from top down (test → environment → agents → components)
- Each component creates its child components

## Connect Phase:

- Connect TLM ports to exports
- Connect analysis ports to subscribers
- Connect drivers to interfaces

## Run Phase:

- Execute test sequences while raising objections
- Generate stimulus and collect responses
- Monitor transactions and verify behavior
- Drop objections when testing is complete

## Report Phase:

- Generate test results and coverage reports

## Assertion-Based Verification

The project implements SystemVerilog assertions (SVA) to verify FIFO behavior:

- **Combinational Flags:** Verify correct setting of full, empty, almost\_full and almost\_empty flags
- **Sequential Flags:** Verify overflow, underflow and write acknowledge behavior
- **Reset Conditions:** Verify proper reset of count and pointers
- **Pointer Behavior:** Verify pointer wraparound and bounds checking

SVA uses bind statements to connect assertions to the DUT while maintaining separation of concerns.

## Conclusion

This UVM verification environment provides comprehensive verification of the FIFO design through constrained-random stimulus, functional coverage, and assertions. The modular architecture allows for easy extension and reuse in future projects.

## Shared Package:

```
package Shared_pkg;
  parameter FIFO_WIDTH = 16;
  parameter FIFO_DEPTH = 8;
  localparam max_fifo_addr = $clog2(FIFO_DEPTH);
  //signals for seq_item
  logic RD_ACTIVE,WR_ACTIVE;
  int RD_EN_ON_DIST=30,WR_EN_ON_DIST=70;
endpackage
```

## Interface:

```
interface FIFO_if (input bit clk);
  logic [15:0] data_in;
  logic rst_n, wr_en, rd_en;
  logic [15:0] data_out;
  logic wr_ack, overflow;
  logic full, empty, almostfull, almostempty, underflow;

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

## Top Module:

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

module top();
  //CLK Generation
  bit clk;
  always #1 clk = ~clk;
  //Interface Instantiation
  FIFO_if fifo_if(clk);
  //DUT instantiation
  FIFO DUT(.fifo_if(fifo_if));
  //Assertion
  bind FIFO FIFO_SVA SVA(.fifo_if(fifo_if));

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

endmodule
```

## Test Component:

```
package FIFO_test_pkg;

import FIFO_env_pkg::*;
import FIFO_config_pkg::*;
import FIFO_sequence_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;
    FIFO_main1_seq main1_seq;
    FIFO_main2_seq main2_seq;
    FIFO_main3_seq main3_seq;
    FIFO_main4_seq main4_seq;
    FIFO_rst_seq rst_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",this);
        main1_seq=FIFO_main1_seq::type_id::create("main1_seq",this);
        main2_seq=FIFO_main2_seq::type_id::create("main2_seq",this);
        main3_seq=FIFO_main3_seq::type_id::create("main3_seq",this);
        main4_seq=FIFO_main4_seq::type_id::create("main4_seq",this);
        rst_seq=FIFO_rst_seq::type_id::create("rst_seq",this);

        if(!uvm_config_db #(virtual FIFO_if)::get(this,"","FIFO_IF",FIFO_cfg.FIFO_vif))begin
            `uvm_fatal("build_phase","test-unable to get the configuration object")
        end
        //                                         anyone
        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);
    // To display a msg                                ----> verbosity
        `uvm_info("run_phase","welcome to the uvm env.",UVM_MEDIUM);
    endtask
endclass
```

```

//RST SEQUENCE
`uvm_info("run_phase","Reset Asserted.",UVM_LOW);
rst_seq.start(env.agt.sqr);
`uvm_info("run_phase","Reset Desserted.",UVM_LOW);

//MAIN1_SEQUENCE
`uvm_info("run_phase","Main1 started.",UVM_LOW);
main1_seq.start(env.agt.sqr);
`uvm_info("run_phase","Main1 Finished.",UVM_LOW);

//MAIN2_SEQUENCE
`uvm_info("run_phase","Main2 started.",UVM_LOW);
main2_seq.start(env.agt.sqr);
`uvm_info("run_phase","Main2 Finished.",UVM_LOW);

//MAIN3_SEQUENCE
`uvm_info("run_phase","Main3 started.",UVM_LOW);
main3_seq.start(env.agt.sqr);
`uvm_info("run_phase","Main3 Finished.",UVM_LOW);

//MAIN4_SEQUENCE
`uvm_info("run_phase","Main4 started.",UVM_LOW);
main4_seq.start(env.agt.sqr);
`uvm_info("run_phase","Main4 Finished.",UVM_LOW);

phase.drop_objection(this);
endtask:run_phase

endclass: FIFO_test
endpackage

```

## Configuration Object:

```

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

```

## Environment Component:

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

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

  // Build the driver in the build phase
  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

endclass
endpackage
```

## Sequence Item Object:

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

class FIFO_seq_item extends uvm_sequence_item;
  `uvm_object_utils(FIFO_seq_item)

  rand logic [15:0] data_in;
  rand logic rst_n, wr_en, rd_en;
  logic [15:0] data_out;
  logic wr_ack, overflow;
  logic full, empty, almostfull, almostempty, underflow;

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

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

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

  constraint rst_const{
    rst_n dist{0:=5,1:=95};
  }

  constraint wr_en_const {
    if (WR_ACTIVE)
      wr_en dist {1:=WR_EN_ON_DIST, 0:=(100-WR_EN_ON_DIST)};
    else
      wr_en == 0;
  }

  constraint rd_en_const {
    if (RD_ACTIVE)
      rd_en dist {1:=RD_EN_ON_DIST, 0:=(100-RD_EN_ON_DIST)};
    else
      rd_en == 0;
  }
```

```

constraint data_in_const {
    data_in dist {16'h0000:=10, 16'hFFFF:=10, [16'h0001:16'hFFFE]:=80};
}

constraint sequential_ops {
    rd_en && wr_en dist {1:=30, 0:=70};
}

endclass
endpackage

```

## Sequencer Object:

```

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

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

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

endclass
endpackage

```

## Sequence Object :

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

//FIFO_1
class FIFO_RST_seq extends uvm_sequence #(FIFO_seq_item);
`uvm_object_utils(FIFO_RST_seq)

    FIFO_seq_item seq_item;

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

    task body;
        seq_item = FIFO_seq_item::type_id::create("seq_item");
        start_item(seq_item);
        seq_item.rst_n=0;
        finish_item(seq_item);
    endtask

endclass
//FIFO_2
class FIFO_main1_seq extends uvm_sequence #(FIFO_seq_item);
`uvm_object_utils(FIFO_main1_seq)

    FIFO_seq_item seq_item;

    function new(string name ="FIFO_main1_sequence");
        super.new(name);
    endfunction

    task body;
        repeat(10)begin
            seq_item = FIFO_seq_item::type_id::create("seq_item");
            WR_ACTIVE=1;
            RD_ACTIVE=0;
            start_item(seq_item);
            assert(seq_item.randomize());
            finish_item(seq_item);
        end
    endtask
endclass

//FIFO_3
class FIFO_main2_seq extends uvm_sequence #(FIFO_seq_item);
```

```

`uvm_object_utils(FIFO_main2_seq)

FIFO_seq_item seq_item;

function new(string name ="FIFO_main2_sequence");
    super.new(name);
endfunction

task body;
repeat(10)begin
    seq_item = FIFO_seq_item::type_id::create("seq_item");
    WR_ACTIVE=0;
    RD_ACTIVE=1;
    start_item(seq_item);
    assert(seq_item.randomize());
    finish_item(seq_item);
end
endtask
endclass

//FIFO_4
class FIFO_main3_seq extends uvm_sequence #(FIFO_seq_item);
`uvm_object_utils(FIFO_main3_seq)

FIFO_seq_item seq_item;

function new(string name ="FIFO_main3_sequence");
    super.new(name);
endfunction

task body;
repeat(1000) begin
    repeat(6)begin
        seq_item = FIFO_seq_item::type_id::create("seq_item");
        WR_ACTIVE=1;
        RD_ACTIVE=0;
        start_item(seq_item);
        assert(seq_item.randomize());
        finish_item(seq_item);
    end
    repeat(6)begin
        seq_item = FIFO_seq_item::type_id::create("seq_item");
        WR_ACTIVE=0;
        RD_ACTIVE=1;
        start_item(seq_item);
        assert(seq_item.randomize());
        finish_item(seq_item);
    end
end
endtask
endclass

```

```

endclass
//FIFO_5
class FIFO_main4_seq extends uvm_sequence #(FIFO_seq_item);
`uvm_object_utils(FIFO_main4_seq)

FIFO_seq_item seq_item;

function new(string name ="FIFO_main4_sequence");
    super.new(name);
endfunction

task body;
repeat(5000)begin
    seq_item = FIFO_seq_item::type_id::create("seq_item");
    WR_ACTIVE=1;
    RD_ACTIVE=1;
    start_item(seq_item);
    assert(seq_item.randomize());
    finish_item(seq_item);
end
endtask
endclass

endpackage

```

## Agent Component :

```
package FIFO_agent_pkg;
  import uvm_pkg::*;
  import FIFO_config_pkg::*;
  import FIFO_drv_pkg::*;
  import FIFO_monitor_pkg::*;
  import FIFO_sequencer_pkg::*;
  import FIFO_seq_item_pkg::*;
  import FIFO_sequence_pkg::*;
  `include "uvm_macros.svh"

class FIFO_agent extends uvm_agent;
  `uvm_component_utils(FIFO_agent)

  FIFO_sequencer sqr;
  FIFO_monitor mon;
  FIFO_drv drv;
  FIFO_config FIFO_cfg;
  uvm_analysis_port #(FIFO_seq_item) agt_ap;

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

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db #(FIFO_config)::get(this,"","CFG",FIFO_cfg))begin
      `uvm_fatal("build_phase","unable to get the configuration object")
    end
    sqr=FIFO_sequencer::type_id::create("sqr",this);
    mon=FIFO_monitor::type_id::create("mon",this);
    drv=FIFO_drv::type_id::create("drv",this);
    agt_ap = new ("agt_ap",this);
  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(agt_ap);
  endfunction
endclass
endpackage
```

## Driver Component :

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

class FIFO_drv extends uvm_driver #(FIFO_seq_item);
    `uvm_component_utils(FIFO_drv)

    virtual FIFO_if FIFO_vif;
    FIFO_config FIFO_cfg;
    FIFO_seq_item stim_seq_item;

    function new(string name ="FIFO_drv",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_seq_item::type_id::create("stim_seq_item");
            seq_item_port.get_next_item(stim_seq_item);
            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;
            FIFO_vif.rst_n = stim_seq_item.rst_n;
            @(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 Component:

```
package FIFO_monitor_pkg;
import uvm_pkg::*;
import FIFO_config_pkg::*;
import FIFO_seq_item_pkg::*;
import Shared_pkg::*;
`include "uvm_macros.svh"
class FIFO_monitor extends uvm_monitor;

  `uvm_component_utils(FIFO_monitor)

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

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

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

  task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
      rsp_seq_item =FIFO_seq_item::type_id::create("rsp_seq_item");
      @(negedge FIFO_vif.clk);
      rsp_seq_item.data_in    = FIFO_vif.data_in    ;
      rsp_seq_item.wr_en     = FIFO_vif.wr_en     ;
      rsp_seq_item.rd_en     = FIFO_vif.rd_en     ;
      rsp_seq_item.rst_n     = FIFO_vif.rst_n     ;
      rsp_seq_item.data_out   = FIFO_vif.data_out   ;
      rsp_seq_item.full      = FIFO_vif.full      ;
      rsp_seq_item.almostfull = FIFO_vif.almostfull ;
      rsp_seq_item.empty      = FIFO_vif.empty      ;
      rsp_seq_item.almostempty= FIFO_vif.almostempty;
      rsp_seq_item.overflow   = FIFO_vif.overflow   ;
      rsp_seq_item.underflow  = FIFO_vif.underflow  ;
      rsp_seq_item.wr_ack     = FIFO_vif.wr_ack     ;
      mon_ap.write(rsp_seq_item);
      `uvm_info("run_phase",rsp_seq_item.convert2string(),UVM_HIGH);
    end
  endtask

endclass
endpackage
```

## Scoreboard Component:

```
package FIFO_scoreboard_pkg;
import uvm_pkg::*;
import Shared_pkg::*;
import FIFO_seq_item_pkg::*;
`include "uvm_macros.svh"

class FIFO_scoreboard extends uvm_scoreboard;
  `uvm_component_utils(FIFO_scoreboard)

  uvm_analysis_export #(FIFO_seq_item) sb_export;
  uvm_tlm_analysis_fifo #(FIFO_seq_item) sb_fifo;
  FIFO_seq_item seq_item_sb;
  logic [15:0] data_out_ref;
  logic wr_ack_ref;
  logic [7:0] arr [15:0];
  int error_count=0;
  int correct_count=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

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

  task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
      sb_fifo.get(seq_item_sb);
      // Compare with task output or with golden model output
      if((data_out_ref != seq_item_sb.data_out) || (wr_ack_ref !=seq_item_sb.wr_ack))begin
        `uvm_error("run_phase",$sformatf("comparison failed,Check output:%s",seq_item_sb.convert2string()));
        error_count++;
      end
      else begin
        `uvm_info("run_phase",$sformatf("correct FIFO_out:%s",seq_item_sb.convert2string()),UVM_HIGH);
        correct_count++;
      end
    end
  endtask
```

```

task ref_model(FIFO_seq_item seq_item_chk);
    static logic [2:0] rd_ptr = 0;
    static logic [2:0] wr_ptr = 0;
    static logic [3:0] count = 0;
    logic read;

    if (!seq_item_chk.rst_n) begin
        count = 0;
        rd_ptr = 0;
        wr_ptr = 0;
        wr_ack_ref = 0;
    end
    else begin

        // Handle write operation
        if (seq_item_chk.wr_en && count < 8) begin
            arr[wr_ptr] = seq_item_chk.data_in;
            wr_ptr++;
            wr_ack_ref = 1;
        end
        else begin
            wr_ack_ref = 0;
        end

        // Handle read operation using saved value if appropriate
        if (seq_item_chk.rd_en && count > 0) begin
            data_out_ref = arr[rd_ptr];
            rd_ptr++;
            read = 1;
        end
        else begin
            read = 0;
        end

        // Update count based on operations
        if (wr_ack_ref && !read ) begin
            count++;
        end
        else if (!wr_ack_ref && read) begin
            count --;
        end
    end
endtask

```

```

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

```

## Coverage Components:

```

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

class FIFO_coverage extends uvm_component;
    `uvm_component_utils(FIFO_coverage)

    uvm_analysis_export #(FIFO_seq_item) cov_export;
    uvm_tlm_analysis_fifo #(FIFO_seq_item) cov_fifo;
    FIFO_seq_item seq_item_cov;

    // Coverage groups
    covergroup C_GP_1;
        read_en:      coverpoint seq_item_cov.rd_en;
        write_en:     coverpoint seq_item_cov.wr_en;
        write_ack:   coverpoint seq_item_cov.wr_ack;
        overflow:    coverpoint seq_item_cov.overflow;
        underflow:   coverpoint seq_item_cov.underflow;
        full:        coverpoint seq_item_cov.full;
        empty:       coverpoint seq_item_cov.empty;
        almostempty: coverpoint seq_item_cov.almostempty;
        almostfull:  coverpoint seq_item_cov.almostfull;

        cross_write_ack:      cross read_en, write_en, write_ack {
            ignore_bins impossible_write_ack = binsof(write_en) intersect {0} && binsof(write_ack) intersect {1};
            //Impossible to have Write_ack with WR_EN off
        }
        cross_overflow:       cross read_en, write_en, overflow {
            ignore_bins impossible_overflow = binsof(write_en) intersect {0} && binsof(overflow) intersect {1};
            //Impossible to have overflow when Write_en is off
        }
        cross_full:          cross read_en, write_en, full {
            ignore_bins impossible_full = binsof(read_en) intersect {1} && binsof(full) intersect {1};
            //Impossible to see Full and Read_en on same cycle
        }
    endgroup
endclass

```

```

cross_underflow:      cross read_en, write_en, underflow;
cross_empty:         cross read_en, write_en, empty;
cross_almostempty:   cross read_en, write_en, almostempty;
cross_almostfull:    cross read_en, write_en, almostfull;

endgroup

function new(string name = "FIFO_cov", uvm_component parent = null);
    super.new(name, parent);
    C_GP_1=new();
endfunction

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

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

task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
        cov_fifo.get(seq_item_cov);
        C_GP_1.sample();
    end
endtask

endclass
endpackage

```

## Assertions Table :

Feature	Assertion
<b>FIFO signals 'full' when count equals maximum capacity</b>	<code>@(posedge fifo_if.clk) fifo_if.rst_n  -&gt; (fifo_if.full == (top.DUT.count == FIFO_DEPTH))</code>
<b>FIFO signals 'empty' when no elements are stored</b>	<code>@(posedge fifo_if.clk) fifo_if.rst_n  -&gt; (fifo_if.empty == (top.DUT.count == 0))</code>
<b>FIFO signals 'almost full' when only one slot remains</b>	<code>@(posedge fifo_if.clk) fifo_if.rst_n  -&gt; (fifo_if.almostfull == (top.DUT.count == FIFO_DEPTH-1))</code>
<b>FIFO signals 'almost empty' when only one element remains</b>	<code>@(posedge fifo_if.clk) fifo_if.rst_n  -&gt; (fifo_if.almostempty == (top.DUT.count == 1))</code>
<b>FIFO counter resets to zero when reset is active</b>	<code>@(posedge fifo_if.clk) !fifo_if.rst_n  -&gt; (top.DUT.count == 0)</code>
<b>Write pointer resets to zero when reset is active</b>	<code>@(posedge fifo_if.clk) !fifo_if.rst_n  -&gt; (top.DUT.wr_ptr == 0)</code>
<b>Read pointer resets to zero when reset is active</b>	<code>@(posedge fifo_if.clk) !fifo_if.rst_n  -&gt; (top.DUT.rd_ptr == 0)</code>
<b>Writing to a full FIFO triggers the overflow flag</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) fifo_if.full &amp;&amp; fifo_if.wr_en  =&gt; fifo_if.overflow</code>
<b>Reading from an empty FIFO triggers the underflow flag</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) fifo_if.empty &amp;&amp; fifo_if.rd_en  =&gt; fifo_if.underflow</code>
<b>Write acknowledge is set after valid write operation</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) fifo_if.wr_en &amp;&amp; top.DUT.count&lt;FIFO_DEPTH  =&gt; fifo_if.wr_ack</code>
<b>Read and write pointers never exceed FIFO depth</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (top.DUT.wr_ptr &lt; FIFO_DEPTH) &amp;&amp; (top.DUT.rd_ptr &lt; FIFO_DEPTH)</code>
<b>Write pointer wraps around to zero after reaching maximum</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) top.DUT.wr_ptr == {max_fifo_addr{1'b1}} &amp;&amp; fifo_if.wr_en &amp;&amp; top.DUT.count &lt; FIFO_DEPTH  =&gt; top.DUT.wr_ptr == {max_fifo_addr{1'b0}}</code>
<b>Read pointer wraps around to zero after reaching maximum</b>	<code>@(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) top.DUT.rd_ptr == {max_fifo_addr{1'b1}} &amp;&amp; fifo_if.rd_en &amp;&amp; top.DUT.count &gt; 0  =&gt; top.DUT.rd_ptr == 0</code>

## SystemVerilog Assertions File:

```
import Shared_pkg::*;

module FIFO_SVA (FIFO_if.DUT fifo_if);
//Combinational flags
property FULL_FLAG;
    @(posedge fifo_if.clk) fifo_if.rst_n |-> (fifo_if.full == (top.DUT.count == FIFO_DEPTH));
endproperty

property EMPTY_FLAG;
    @(posedge fifo_if.clk) fifo_if.rst_n |-> (fifo_if.empty == (top.DUT.count == 0));
endproperty

property ALMOST_FULL_FLAG;
    @(posedge fifo_if.clk) fifo_if.rst_n |-> (fifo_if.almostfull == (top.DUT.count == FIFO_DEPTH-1));
endproperty

property ALMOST_EMPTY_FLAG;
    @(posedge fifo_if.clk) fifo_if.rst_n |-> (fifo_if.almostempty == (top.DUT.count == 1));
endproperty

// Reset condition assertions
property COUNT_RESET;
    @(posedge fifo_if.clk) !fifo_if.rst_n |-> (top.DUT.count == 0);
endproperty

property WR_PTR_RESET;
    @(posedge fifo_if.clk) !fifo_if.rst_n |-> (top.DUT.wr_ptr == 0);
endproperty

property RD_PTR_RESET;
    @(posedge fifo_if.clk) !fifo_if.rst_n |-> (top.DUT.rd_ptr == 0);
endproperty

//Sequential Flags:
property OVERFLOW;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) fifo_if.full && fifo_if.wr_en |=> fifo_if.overflow;
endproperty

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

property WR_ACK;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) fifo_if.wr_en && top.DUT.count<FIFO_DEPTH |=> fifo_if.wr_ack;
endproperty
```

```

//Pointers assertions:
property PTR_EXCEEDED;
  @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (top.DUT.wr_ptr < FIFO_DEPTH) &&
  (top.DUT.rd_ptr < FIFO_DEPTH);
endproperty

property WR_PTR_WRAPAROUND;
  @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) top.DUT.wr_ptr ==
  {max_fifo_addr{1'b1}} && fifo_if.wr_en && top.DUT.count < FIFO_DEPTH |=> top.DUT.wr_ptr ==
  {max_fifo_addr{1'b0}};
endproperty

property RD_PTR_WRAPAROUND;
  @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) top.DUT.rd_ptr ==
  {max_fifo_addr{1'b1}} && fifo_if.rd_en && top.DUT.count > 0 |=> top.DUT.rd_ptr == 0;
endproperty

// Assertions for sequential flags:
assert property(OVERFLOW)           else $error("Overflow flag assertion failed");
assert property(UNDERFLOW)          else $error("Underflow flag assertion failed");

// Assert Combinational flags:
assert property(FULL_FLAG)          else $error("Full flag incorrect");
assert property(EMPTY_FLAG)          else $error("Empty flag assertion failed");
assert property(ALMOST_FULL_FLAG)   else $error("Almost Full flag assertion failed");
assert property(ALMOST_EMPTY_FLAG)  else $error("Almost Empty flag assertion failed");
assert property(COUNT_RESET)        else $error("Count reset assertion failed");
assert property(WR_PTR_RESET)       else $error("Write ptr reset assertion failed");
assert property(RD_PTR_RESET)       else $error("Read ptr reset assertion failed");
// Assertions for sequential flags:
assert property(WR_ACK)            else $error("Write Acknowledge flag assertion failed");

// Assertion for pointers not exceeding FIFO depth:
assert property(PTR_EXCEEDED)      else $error("Pointers exceeded FIFO depth");

// Assertions for pointer wraparound:
assert property(WR_PTR_WRAPAROUND)  else $error("Write pointer wraparound assertion failed");
assert property(RD_PTR_WRAPAROUND)  else $error("Read pointer wraparound assertion failed");

// Cover properties to verify they can be reached:
cover property(OVERFLOW);
cover property(UNDERFLOW);

// Cover properties to verify they can be reached:
cover property(FULL_FLAG);
cover property(EMPTY_FLAG);

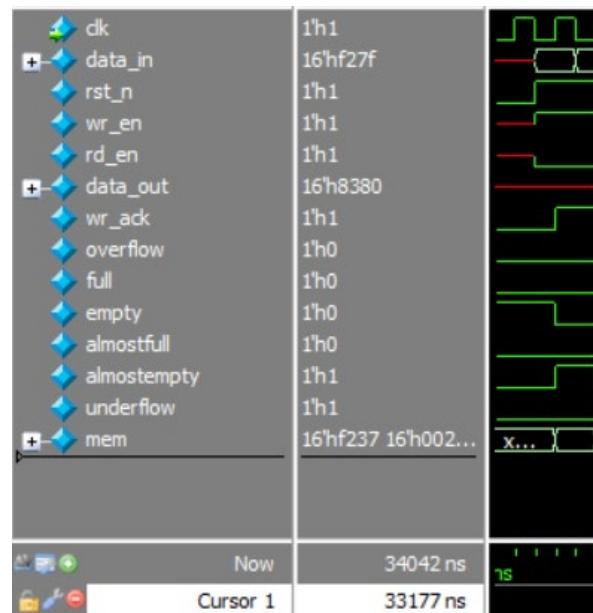
```

```
cover property(ALMOST_FULL_FLAG);
cover property(ALMOST_EMPTY_FLAG);
cover property(WR_ACK);
cover property(PTR_EXCEEDED);
cover property(WR_PTR_WRAPAROUND);
cover property(RD_PTR_WRAPAROUND);
cover property(COUNT_RESET);
cover property(WR_PTR_RESET);
cover property(RD_PTR_RESET);

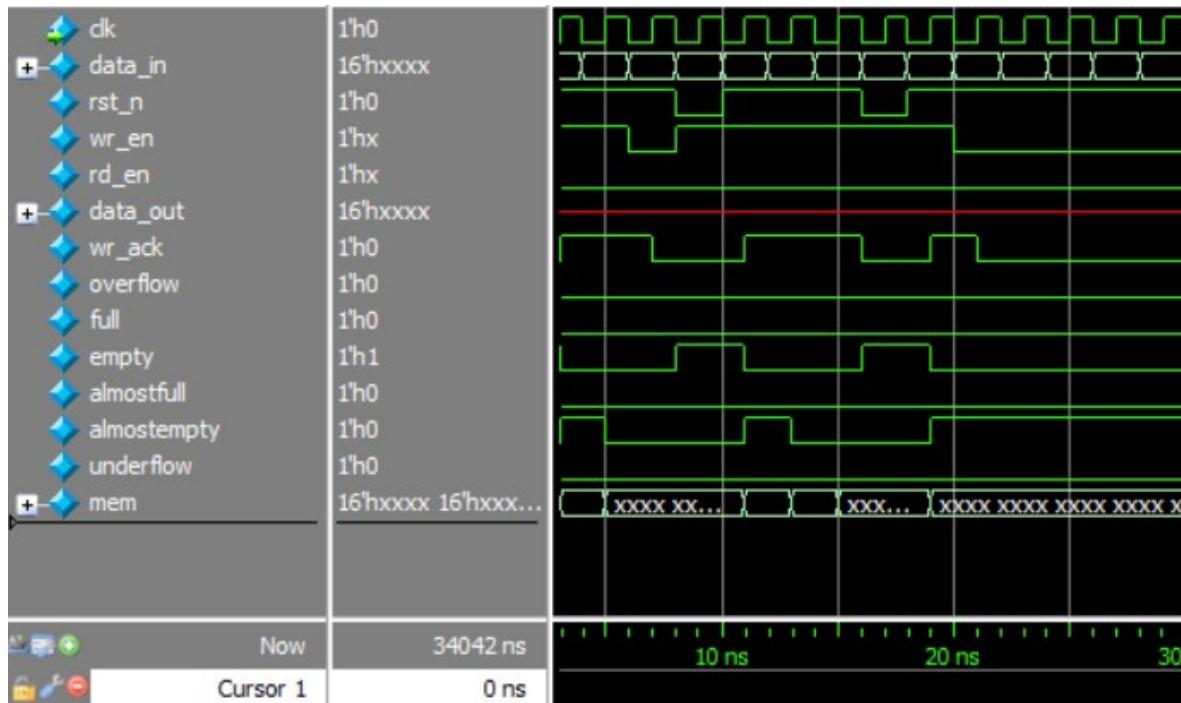
endmodule
```

## Waveforms:

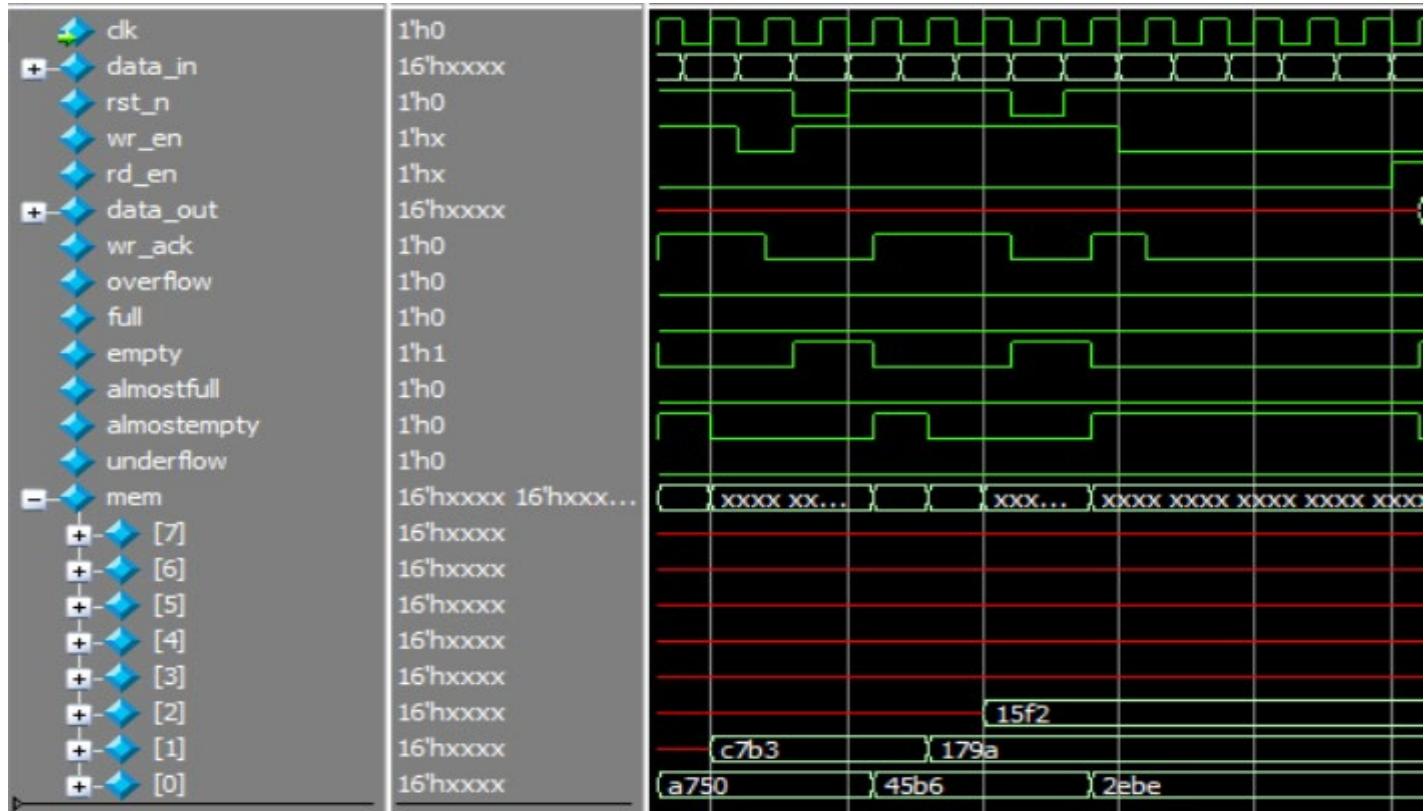
### FIFO\_1:



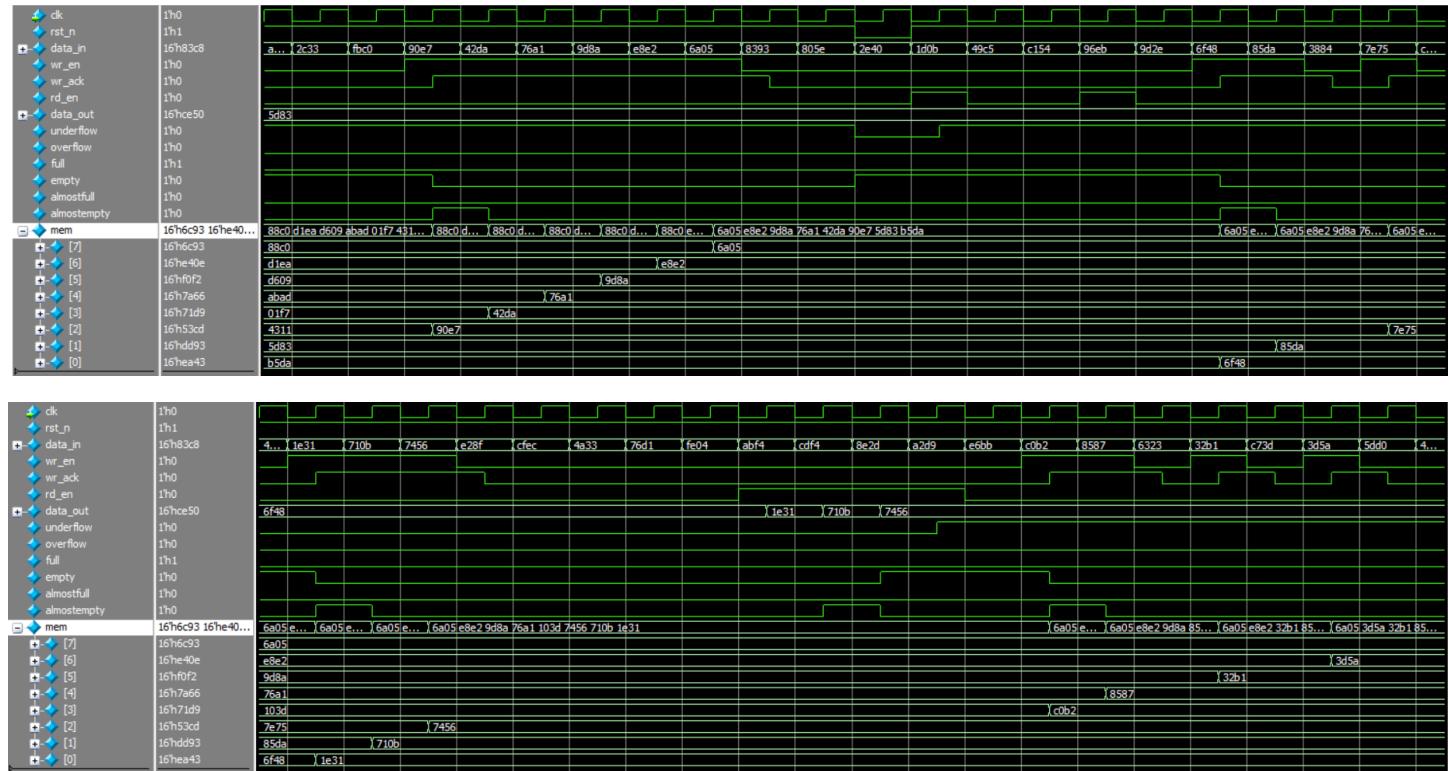
### FIFO\_2:



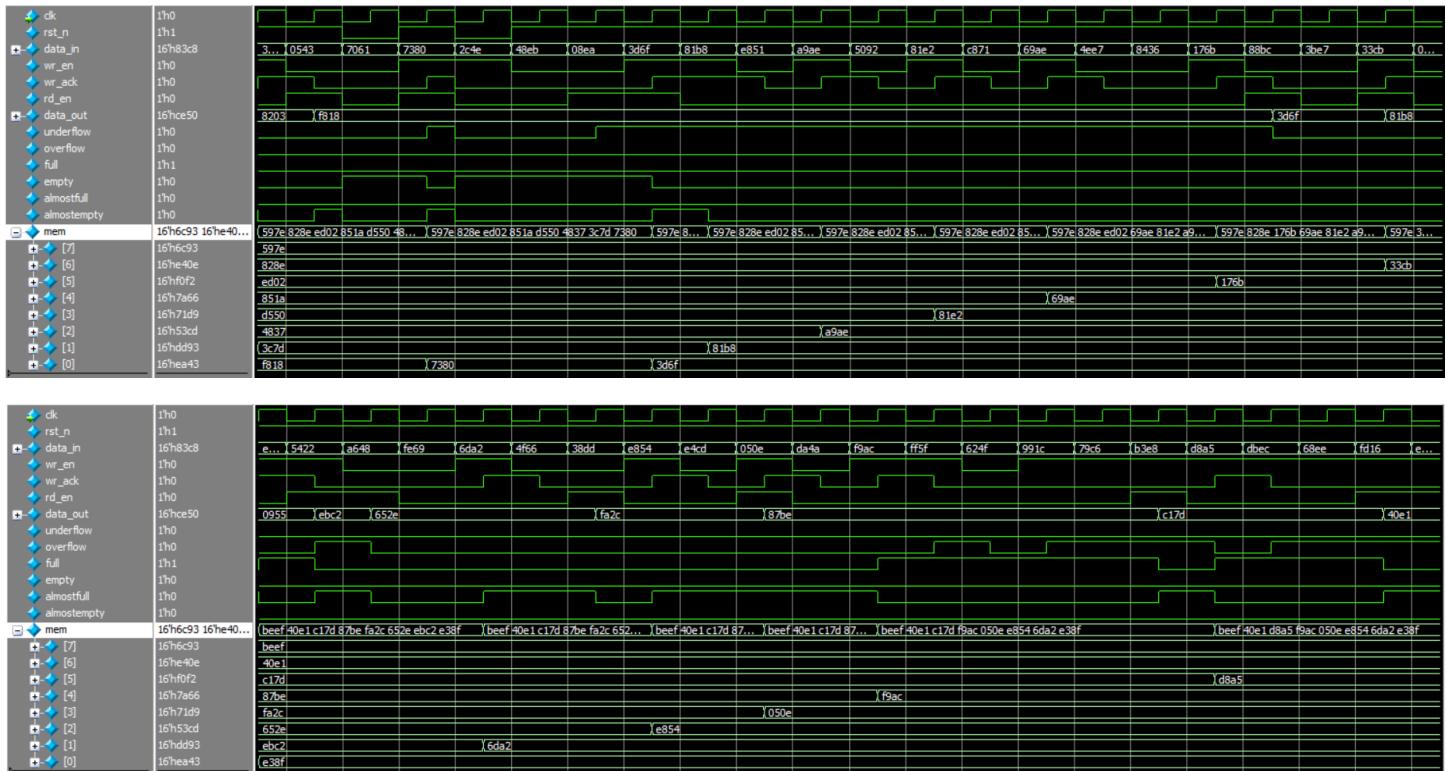
## FIFO\_3:



## FIFO\_4:



## FIFO\_5:



Run.do:

```
vlib work
vlog +acc -f src_files.list +cover=bces -covercells
vsim -voptargs=+acc work.top -classdebug -assertdebug -uvmcontrol=all -cover
run 0
add wave -position insertpoint sim:/top/fifo_if/*
add wave -position insertpoint \
sim:/top/DUT/mem
coverage save top.ucdb -onexit
run -all
quit -sim
vcover report top.ucdb -details -annotate -all -output cov_report.txt
```

## Source List:

```
FIFO.sv
shared_pkg.sv
FIFO_if.sv
FIFO_driver.sv
FIFO_conf.sv
FIFO_seq_item.sv
FIFO_sequencer.sv
FIFO_sequence.sv
FIFO_monitor.sv
FIFO_cov.sv
FIFO_scoreboard.sv
FIFO_agent.sv
FIFO_env.sv
FIFO_test.sv
FIFO_sva.sv
top.sv
```

# Cover Report:

Coverage Report by instance with details									
<pre>===== --- Instance: /top/DUT/SVA --- Design Unit: work.FIFO_SVA =====</pre>									
Assertion Coverage:									
Assertions	13	13	0	100.00%					
Name	File(Line)	Failure Count	Pass Count	Vacuous Count	Disable Count	Attempt Count	Active Count	Peak Active	ATV
/top/DUT/SVA/assert__RD_PTR_WRAPAROUND	FIFO_sva.sv(82)	0	139	15974	908	17021	0	1	off
/top/DUT/SVA/assert__WR_PTR_WRAPAROUND	FIFO_sva.sv(81)	0	424	15676	921	17021	0	1	off
/top/DUT/SVA/assert__PTR_EXCEEDED	FIFO_sva.sv(78)	0	16123	0	898	17021	0	1	off
/top/DUT/SVA/assert__WR_ACK	FIFO_sva.sv(75)	0	5815	9964	1241	17021	1	2	off
/top/DUT/SVA/assert__RD_PTR_RESET	FIFO_sva.sv(73)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__WR_PTR_RESET	FIFO_sva.sv(72)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__COUNT_RESET	FIFO_sva.sv(71)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__ALMOST_EMPTY_FLAG	FIFO_sva.sv(70)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__ALMOST_FULL_FLAG	FIFO_sva.sv(69)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__EMPTY_FLAG	FIFO_sva.sv(68)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__FULL_FLAG	FIFO_sva.sv(67)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__UNDERFLOW	FIFO_sva.sv(63)	0	522	15576	923	17021	0	2	off
/top/DUT/SVA/assert__OVERFLOW	FIFO_sva.sv(62)	0	1030	15041	950	17021	0	2	off

Directive Coverage:									
Directives	13	13	0	100.00%					
DIRECTIVE COVERAGE:									
Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status			
/top/DUT/SVA/cover__RD_PTR_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(99)	898	Covered			
/top/DUT/SVA/cover__WR_PTR_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(98)	898	Covered			
/top/DUT/SVA/cover__COUNT_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(97)	898	Covered			
/top/DUT/SVA/cover__RD_PTR_WRAPAROUND	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(96)	139	Covered			
/top/DUT/SVA/cover__WR_PTR_WRAPAROUND	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(95)	424	Covered			
/top/DUT/SVA/cover__PTR_EXCEEDED	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(94)	16123	Covered			
/top/DUT/SVA/cover__WR_ACK	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(93)	5815	Covered			
/top/DUT/SVA/cover__ALMOST_EMPTY_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(92)	16123	Covered			
/top/DUT/SVA/cover__ALMOST_FULL_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(91)	16123	Covered			
/top/DUT/SVA/cover__EMPTY_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(90)	16123	Covered			
/top/DUT/SVA/cover__FULL_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(89)	16123	Covered			
/top/DUT/SVA/cover__UNDERFLOW	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(86)	522	Covered			
/top/DUT/SVA/cover__OVERFLOW	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(85)	1030	Covered			

Assertion Coverage:								
Assertions	11	11	0	100.00%				
Name	File(Line)	Failure Count	Pass Count	Vacuous Count	Disable Count	Attempt Count	Active Count	Peak Active Count
/top/DUT/assert__RD_PTR_WRAPAROUND	FIFO.sv(150)	0	139	15974	908	17021	0	1 off
/top/DUT/assert__WR_PTR_WRAPAROUND	FIFO.sv(149)	0	424	15676	921	17021	0	1 off
/top/DUT/assert__PTR_EXCEEDED	FIFO.sv(146)	0	16123	0	898	17021	0	1 off
/top/DUT/assert__WR_ACK	FIFO.sv(143)	0	5815	9964	1241	17021	1	2 off
/top/DUT/assert__RD_PTR_RESET	FIFO.sv(141)	0	898	16123	0	17021	0	1 off
/top/DUT/assert__WR_PTR_RESET	FIFO.sv(140)	0	898	16123	0	17021	0	1 off
/top/DUT/assert__COUNT_RESET	FIFO.sv(139)	0	898	16123	0	17021	0	1 off
/top/DUT/assert__ALMOST_EMPTY_FLAG	FIFO.sv(138)	0	16123	898	0	17021	0	1 off
/top/DUT/assert__ALMOST_FULL_FLAG	FIFO.sv(137)	0	16123	898	0	17021	0	1 off
/top/DUT/assert__EMPTY_FLAG	FIFO.sv(136)	0	16123	898	0	17021	0	1 off
/top/DUT/assert__FULL_FLAG	FIFO.sv(135)	0	16123	898	0	17021	0	1 off

```

Branch Coverage:
  Enabled Coverage      Bins    Hits    Misses  Coverage
  -----      -----  -----  -----
  Branches          24      24      0   100.00%
=====
=====Branch Details=====
Branch Coverage for instance /top/DUT

  Line      Item      Count      Source
  ---      ---      ----
File FIFO.sv
  -----IF Branch-----
  21          17865  Count coming in to IF
  21          1742   if (!fifo_if.rst_n) begin
  26          6159   else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
  32          9964   else begin

Branch totals: 3 hits of 3 branches = 100.00%

  -----IF Branch-----
  36          9964   Count coming in to IF
  36          1082   if (fifo_if.full && fifo_if.wr_en)
  38          8882   else begin

Branch totals: 2 hits of 2 branches = 100.00%

  -----IF Branch-----
  46          17865  Count coming in to IF
  46          1742   if (!fifo_if.rst_n) begin
  50          2832   else if (fifo_if.rd_en && count > 0) begin
  55          547    else if ( fifo_if.empty && fifo_if.rd_en)begin
  55          12744  All False Count
Branch totals: 4 hits of 4 branches = 100.00%

  -----IF Branch-----
  62          14007  Count coming in to IF
  62          1711   if (!fifo_if.rst_n) begin
  65          12296  else begin

Branch totals: 2 hits of 2 branches = 100.00%

  -----IF Branch-----
  66          12296  Count coming in to IF
  66          5275   if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b10) && !fifo_if.full)
  68          1849   else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b01) && !fifo_if.empty)

```

```

-----IF Branch-----
71          1           110      else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.empty)
73          1           209      else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)

4853      All False Count
Branch totals: 5 hits of 5 branches = 100.00%

-----IF Branch-----
78          1           8194    Count coming in to IF
78          1           600     assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;

78          2           7594    assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

-----IF Branch-----
79          1           8194    Count coming in to IF
79          1           946     assign fifo_if.empty = (count == 0)? 1 : 0;
79          2           7248    assign fifo_if.empty = (count == 0)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

-----IF Branch-----
82          1           8194    Count coming in to IF
82          1           940     assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
82          2           7254    assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

-----IF Branch-----
83          1           8194    Count coming in to IF
83          1           1169    assign fifo_if.almostempty = (count == 1)? 1 : 0;
83          2           7025    assign fifo_if.almostempty = (count == 1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

```

```

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

Condition Coverage for instance /top/DUT --

  File FIFO.sv
  -----Focused Condition View-----
Line      26 Item    1 (fifo_if.wr_en && (count < 8))
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term  Covered  Reason for no coverage  Hint
  -----  -----  -----
 fifo_if.wr_en      Y
 (count < 8)      Y

  Rows:      Hits  FEC Target           Non-masking condition(s)
  -----  -----
 Row  1:        1  fifo_if.wr_en_0      -
 Row  2:        1  fifo_if.wr_en_1      (count < 8)
 Row  3:        1  (count < 8)_0        fifo_if.wr_en
 Row  4:        1  (count < 8)_1        fifo_if.wr_en

  -----Focused Condition View-----
Line      36 Item    1 (fifo_if.full && fifo_if.wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

  Rows:      Hits  FEC Target           Non-masking condition(s)
  -----  -----
 Row  1:        1  fifo_if.full_0      -
 Row  2:        1  fifo_if.full_1      fifo_if.wr_en
 Row  3:        1  fifo_if.wr_en_0    fifo_if.full
 Row  4:        1  fifo_if.wr_en_1    fifo_if.full

  -----Focused Condition View-----
Line      50 Item    1 (fifo_if.rd_en && (count > 0))
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term  Covered  Reason for no coverage  Hint
  -----  -----  -----
 fifo_if.rd_en      Y
 (count > 0)      Y

```

```

Condition totals: 2 of 2 input terms covered = 100.00%
 fifo_if.rd_en      Y
 (count > 0)      Y

 Rows:     Hits  FEC Target          Non-masking condition(s)
 -----
 Row 1:      1  fifo_if.rd_en_0      -
 Row 2:      1  fifo_if.rd_en_1      (count > 0)
 Row 3:      1  (count > 0)_0      fifo_if.rd_en
 Row 4:      1  (count > 0)_1      fifo_if.rd_en

-----Focused Condition View-----
Line      55 Item   1  (fifo_if.empty && fifo_if.rd_en)
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term  Covered  Reason for no coverage  Hint
  -----
 fifo_if.empty      Y
 fifo_if.rd_en      Y

  Rows:     Hits  FEC Target          Non-masking condition(s)
 -----
 Row 1:      1  fifo_if.empty_0      -
 Row 2:      1  fifo_if.empty_1      fifo_if.rd_en
 Row 3:      1  fifo_if.rd_en_0      fifo_if.empty
 Row 4:      1  fifo_if.rd_en_1      fifo_if.empty

-----Focused Condition View-----
Line      66 Item   1  ((~fifo_if.rd_en && fifo_if.wr_en) && ~fifo_if.full)
Condition totals: 3 of 3 input terms covered = 100.00%

  Input Term  Covered  Reason for no coverage  Hint
  -----
 fifo_if.rd_en      Y
 fifo_if.wr_en      Y
 fifo_if.full      Y

  Rows:     Hits  FEC Target          Non-masking condition(s)
 -----
 Row 1:      1  fifo_if.rd_en_0      (~fifo_if.full && fifo_if.wr_en)
 Row 2:      1  fifo_if.rd_en_1      -
 Row 3:      1  fifo_if.wr_en_0      ~fifo_if.rd_en
 Row 4:      1  fifo_if.wr_en_1      (~fifo_if.full && ~fifo_if.rd_en)
 Row 5:      1  fifo_if.full_0      (~fifo_if.rd_en && fifo_if.wr_en)
 Row 6:      1  fifo_if.full_1      (~fifo_if.rd_en && fifo_if.wr_en)

-----Focused Condition View-----
Line      68 Item   1  ((fifo_if.rd_en && ~fifo_if.wr_en) && ~fifo_if.empty)
Condition totals: 3 of 3 input terms covered = 100.00%

```

Condition totals: 3 of 3 input terms covered = 100.00%

Input Term	Covered	Reason for no coverage	Hint
fifo_if.rd_en	Y		
fifo_if.wr_en	Y		
fifo_if.empty	Y		

Rows: Hits FEC Target Non-masking condition(s)

Row 1:	1	fifo_if.rd_en_0	-
Row 2:	1	fifo_if.rd_en_1	(~fifo_if.empty && ~fifo_if.wr_en)
Row 3:	1	fifo_if.wr_en_0	(~fifo_if.empty && fifo_if.rd_en)
Row 4:	1	fifo_if.wr_en_1	fifo_if.rd_en
Row 5:	1	fifo_if.empty_0	(fifo_if.rd_en && ~fifo_if.wr_en)
Row 6:	1	fifo_if.empty_1	(fifo_if.rd_en && ~fifo_if.wr_en)

-----Focused Condition View-----

Line 71 Item 1 ((fifo\_if.rd\_en && fifo\_if.wr\_en) && fifo\_if.empty)

Condition totals: 3 of 3 input terms covered = 100.00%

Input Term Covered Reason for no coverage Hint

fifo_if.rd_en	Y
fifo_if.wr_en	Y
fifo_if.empty	Y

Rows: Hits FEC Target Non-masking condition(s)

Row 1:	1	fifo_if.rd_en_0	-
Row 2:	1	fifo_if.rd_en_1	(fifo_if.empty && fifo_if.wr_en)
Row 3:	1	fifo_if.wr_en_0	fifo_if.rd_en
Row 4:	1	fifo_if.wr_en_1	(fifo_if.empty && fifo_if.rd_en)
Row 5:	1	fifo_if.empty_0	(fifo_if.rd_en && fifo_if.wr_en)
Row 6:	1	fifo_if.empty_1	(fifo_if.rd_en && fifo_if.wr_en)

-----Focused Condition View-----

Line 73 Item 1 ((fifo\_if.rd\_en && fifo\_if.wr\_en) && fifo\_if.full)

Condition totals: 3 of 3 input terms covered = 100.00%

Input Term Covered Reason for no coverage Hint

fifo_if.rd_en	Y
fifo_if.wr_en	Y
fifo_if.full	Y

Rows: Hits FEC Target Non-masking condition(s)

Row 1:	1	fifo_if.rd_en_0	-
Row 2:	1	fifo_if.rd_en_1	(fifo_if.full && fifo_if.wr_en)
Row 3:	1	fifo_if.wr_en_0	fifo_if.rd_en
Row 4:	1	fifo_if.wr_en_1	(fifo_if.full && fifo_if.rd_en)
Row 5:	1	fifo_if.full_0	(fifo_if.rd_en && fifo_if.wr_en)
Row 6:	1	fifo_if.full_1	(fifo_if.rd_en && fifo_if.wr_en)

-----Focused Condition View-----

Line 78 Item 1 (count == 8)

Condition totals: 1 of 1 input term covered = 100.00%

Input Term	Covered	Reason for no coverage	Hint
(count == 8)	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
-------	------	------------	--------------------------

Row 1:	1	(count == 8)_0	-
Row 2:	1	(count == 8)_1	-

-----Focused Condition View-----

Line 79 Item 1 (count == 0)

Condition totals: 1 of 1 input term covered = 100.00%

Input Term	Covered	Reason for no coverage	Hint
(count == 0)	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
-------	------	------------	--------------------------

Row 1:	1	(count == 0)_0	-
Row 2:	1	(count == 0)_1	-

-----Focused Condition View-----

Line 82 Item 1 (count == (8 - 1))

Condition totals: 1 of 1 input term covered = 100.00%

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

Rows:	Hits	FEC Target	Non-masking condition(s)
-------	------	------------	--------------------------

Row 1:	1	(count == (8 - 1))_0	-
Row 2:	1	(count == (8 - 1))_1	-

-----Focused Condition View-----

Line 83 Item 1 (count == 1)

Condition totals: 1 of 1 input term covered = 100.00%

Input Term	Covered	Reason for no coverage	Hint
(count == 1)	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
-------	------	------------	--------------------------

Row 1:	1	(count == 1)_0	-
Row 2:	1	(count == 1)_1	-

```

Directive Coverage:
  Directives          11        11        0  100.00%
DIRECTIVE COVERAGE:
-----
Name          Design  Design  Lang File(Line)  Hits Status
      Unit    UnitType
-----
/top/DUT/cover__RD_PTR_RESET      FIFO  Verilog  SVA  FIFO.sv(163)  898 Covered
/top/DUT/cover__WR_PTR_RESET      FIFO  Verilog  SVA  FIFO.sv(162)  898 Covered
/top/DUT/cover__COUNT_RESET      FIFO  Verilog  SVA  FIFO.sv(161)  898 Covered
/top/DUT/cover__RD_PTR_WRAPAROUND  FIFO  Verilog  SVA  FIFO.sv(160)  139 Covered
/top/DUT/cover__WR_PTR_WRAPAROUND  FIFO  Verilog  SVA  FIFO.sv(159)  424 Covered
/top/DUT/cover__PTR_EXCEEDED      FIFO  Verilog  SVA  FIFO.sv(158)  16123 Covered
/top/DUT/cover__WR_ACK            FIFO  Verilog  SVA  FIFO.sv(157)  5815 Covered
/top/DUT/cover__ALMOST_EMPTY_FLAG  FIFO  Verilog  SVA  FIFO.sv(156)  16123 Covered
/top/DUT/cover__ALMOST_FULL_FLAG  FIFO  Verilog  SVA  FIFO.sv(155)  16123 Covered
/top/DUT/cover__EMPTY_FLAG        FIFO  Verilog  SVA  FIFO.sv(154)  16123 Covered
/top/DUT/cover__FULL_FLAG         FIFO  Verilog  SVA  FIFO.sv(153)  16123 Covered
Statement Coverage:
  Enabled Coverage
  -----
  Statements          Bins   Hits   Misses  Coverage
  -----
  Statements           28     28      0  100.00%
=====
=====Statement Details=====
Statement Coverage for instance /top/DUT --
Line      Item          Count      Source
---      ---
File FIFO.sv
 8          module FIFO(FIFO_if.DUT fifo_if);
 9          parameter FIFO_WIDTH = 16;
10         parameter FIFO_DEPTH = 8;
11
12         localparam max_fifo_addr = $clog2(FIFO_DEPTH);
13
14         reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
15
16         reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
17         reg [max_fifo_addr:0] count;
18
19         //Write Operation & sequential Flags handling:

```

```
Statement Coverage for instance /top/DUT --
File FIFO.sv

 20      1          17865    always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
 21                      if (!fifo_if.rst_n) begin
 22              1          1742      wr_ptr <= 0;
 23              1          1742      fifo_if.wr_ack <= 0;      // all flags should be resetted
 24              1          1742      fifo_if.overflow <= 0; // all flags should be resetted
 25
 26                      end
 27
 28                      else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
 29              1          6159      mem[wr_ptr] <= fifo_if.data_in;
 30              1          6159      fifo_if.wr_ack <= 1;
 31              1          6159      wr_ptr <= wr_ptr + 1;
 32              1          6159      fifo_if.overflow <= 0;
 33
 34                      end
 35
 36                      else begin
 37              1          9964      fifo_if.wr_ack <= 0;
 38
 39                      //if (fifo_if.full & fifo_if.wr_en)----->Warning: this statement is not a bitwise operation
 40
 41                      //-- better to use && (meanwhile & will do the same job because oprands are 1 bit)
 42
 43
 44                      if (fifo_if.full && fifo_if.wr_en)
 45              1          1082      fifo_if.overflow <= 1;
 46
 47                      else begin
 48              1          8882      fifo_if.overflow <= 0;
 49
 50                      end
 51
 52                      end
 53
 54
 55                      end
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
50310
50311
50312
50313
50314
50315
50316
50317
50318
50319
50320
50321
50322
50323
50324
50325
50326
50327
50328
50329
50330
50331
50332
50333
50334
50335
50336
50337
50338
50339
50340
50341
50342
50343
50344
50345
50346
50347
50348
50349
50350
50351
50352
50353
50354
50355
50356
50357
50358
50359
50360
50361
50362
50363
50364
50365
50366
50367
50368
50369
50370
50371
50372
50373
50374
50375
50376
50377
50378
50379
50380
50381
50382
50383
50384
50385
50386
50387
50388
50389
503810
503811
503812
503813
503814
503815
503816
503817
503818
503819
503820
503821
503822
503823
503824
503825
503826
503827
503828
503829
503830
503831
503832
503833
503834
503835
503836
503837
503838
503839
503840
503841
503842
503843
503844
503845
503846
503847
503848
503849
503850
503851
503852
503853
503854
503855
503856
503857
503858
503859
503860
503861
503862
503863
503864
503865
503866
503867
503868
503869
503870
503871
503872
503873
503874
503875
503876
503877
503878
503879
503880
503881
503882
503883
503884
503885
503886
503887
503888
503889
5038810
5038811
5038812
5038813
5038814
5038815
5038816
5038817
5038818
5038819
5038820
5038821
5038822
5038823
5038824
5038825
5038826
5038827
5038828
5038829
5038830
5038831
5038832
5038833
5038834
5038835
5038836
5038837
5038838
5038839
5038840
5038841
5038842
5038843
5038844
5038845
5038846
5038847
5038848
5038849
5038850
5038851
5038852
5038853
5038854
5038855
5038856
5038857
5038858
5038859
5038860
5038861
5038862
5038863
5038864
5038865
5038866
5038867
5038868
5038869
5038870
5038871
5038872
5038873
5038874
5038875
5038876
5038877
5038878
5038879
5038880
5038881
5038882
5038883
5038884
5038885
5038886
5038887
5038888
5038889
50388810
50388811
50388812
50388813
50388814
50388815
50388816
50388817
50388818
50388819
50388820
50388821
50388822
50388823
50388824
50388825
50388826
50388827
50388828
50388829
50388830
50388831
50388832
50388833
50388834
50388835
50388836
50388837
50388838
50388839
50388840
50388841
50388842
50388843
50388844
50388845
50388846
50388847
50388848
50388849
50388850
50388851
50388852
50388853
50388854
50388855
50388856
50388857
50388858
50388859
50388860
50388861
50388862
50388863
50388864
50388865
50388866
50388867
50388868
50388869
50388870
50388871
50388872
50388873
50388874
50388875
50388876
50388877
50388878
50388879
50388880
50388881
50388882
50388883
50388884
50388885
50388886
50388887
50388888
50388889
503888810
503888811
503888812
503888813
503888814
503888815
503888816
503888817
503888818
503888819
503888820
503888821
503888822
503888823
503888824
503888825
503888826
503888827
503888828
503888829
503888830
503888831
503888832
503888833
503888834
503888835
503888836
503888837
503888838
503888839
503888840
503888841
503888842
503888843
503888844
503888845
503888846
503888847
503888848
503888849
503888850
503888851
503888852
503888853
503888854
503888855
503888856
503888857
503888858
503888859
503888860
503888861
503888862
503888863
503888864
503888865
503888866
503888867
503888868
503888869
503888870
503888871
503888872
503888873
503888874
503888875
503888876
503888877
503888878
503888879
503888880
503888881
503888882
503888883
503888884
503888885
503888886
503888887
503888888
503888889
5038888810
5038888811
5038888812
5038888813
5038888814
5038888815
5038888816
5038888817
5038888818
5038888819
5038888820
5038888821
5038888822
5038888823
5038888824
5038888825
5038888826
5038888827
5038888828
5038888829
5038888830
5038888831
5038888832
5038888833
5038888834
5038888835
5038888836
5038888837
5038888838
5038888839
5038888840
5038888841
5038888842
5038888843
5038888844
5038888845
5038888846
5038888847
5038888848
5038888849
5038888850
5038888851
5038888852
5038888853
5038888854
5038888855
5038888856
5038888857
5038888858
5038888859
5038888860
5038888861
5038888862
5038888863
5038888864
5038888865
5038888866
5038888867
5038888868
5038888869
5038888870
5038888871
5038888872
5038888873
5038888874
5038888875
5038888876
5038888877
5038888878
5038888879
5038888880
5038888881
5038888882
5038888883
5038888884
5038888885
5038888886
5038888887
5038888888
5038888889
50388888810
50388888811
50388888812
50388888813
50388888814
50388888815
50388888816
50388888817
50388888818
50388888819
50388888820
50388888821
50388888822
50388888823
50388888824
50388888825
50388888826
50388888827
50388888828
50388888829
50388888830
50388888831
50388888832
50388888833
50388888834
50388888835
50388888836
50388888837
50388888838
50388888839
50388888840
50388888841
50388888842
50388888843
50388888844
50388888845
50388888846
50388888847
50388888848
50388888849
50388888850
50388888851
50388888852
50388888853
50388888854
50388888855
50388888856
50388888857
50388888858
50388888859
50388888860
50388888861
50388888862
50388888863
50388888864
50388888865
50388888866
50388888867
50388888868
50388888869
50388888870
50388888871
50388888872
50388888873
50388888874
50388888875
50388888876
50388888877
50388888878
50388888879
50388888880
50388888881
50388888882
50388888883
50388888884
50388888885
50388888886
50388888887
50388888888
50388888889
503888888810
503888888811
503888888812
503888888813
503888888814
503888888815
503888888816
503888888817
503888888818
503888888819
503888888820
503888888821
503888888822
503888888823
503888888824
503888888825
503888888826
503888888827
503888888828
503888888829
503888888830
503888888831
503888888832
503888888833
503888888834
503888888835
503888888836
503888888837
503888888838
503888888839
503888888840
503888888841
503888888842
503888888843
503888888844
503888888845
503888888846
503888888847
503888888848
503888888849
503888888850
503888888851
503888888852
503888888853
503888888854
503888888855
503888888856
503888888857
503888888858
503888888859
503888888860
503888888861
503888888862
503888888863
503888888864
503888888865
503888888866
503888888867
503888888868
503888888869
503888888870
503888888871
503888888872
503888888873
503888888874
503888888875
503888888876
503888888877
503888888878
503888888879
503888888880
503888888881
503888888882
503888888883
503888888884
503888888885
503888888886
503888888887
503888888888
503888888889
5038888888810
5038888888811
5038888888812
5038888888813
5038888888814
5038888888815
5038888888816
5038888888817
5038888888818
5038888888819
5038888888820
5038888888821
5038888888822
5038888888823
5038888888824
5038888888825
5038888888826
5038888888827
5038888888828
5038888888829
5038888888830
5038888888831
5038888888832
5038888888833
5038888888834
5038888888835
5038888888836
5038888888837
5038888888838
5038888888839
5038888888840
5038888888841
5038888888842
5038888888843
5038888888844
5038888888845
5038888888846
5038888888847
5038888888848
5038888888849
5038888888850
5038888888851
5038888888852
5038888888853
5038888888854
5038888888855
5038888888856
5038888888857
5038888888858
5038888888859
5038888888860
5038888888861
5038888888862
5038888888863
5038888888864
5038888888865
5038888888866
5038888888867
5038888888868
5038888888869
5038888888870
5038888888871
5038888888872
5038888888873
5038888888874
5038888888875
5038888888876
5038888888877
5038888888878
5038888888879
5038888888880
5038888888881
5038888888882
5038888888883
5038888888884
5038888888885
5038888888886
5038888888887
5038888888888
5038888888889
50388888888810
50388888888811
50388888888812
50388888888813
50388888888814
50388888888815
50388888888816
50388888888817
50388888888818
50388888888819
50388888888820
50388888888821
50388888888822
50388888888823
50388888888824
50388888888825
50388888888826
50388888888827
50388888888828
50388888888829
50388888888830
50388888888831
50388888888832
50388888888833
50388888888834
50388888888835
50388888888836
50388888888837
50388888888838
50388888888839
50388888888840
50388888888841
50388888888842
50388888888843
50388888888844
50388888888845
50388888888846
50388888888847
50388888888848
50388888888849
50388888888850
50388888888851
50388888888852
50388888888853
50388888888854
50388888888855
50388888888856
50388888888857
50388888888858
50388888888859
50388888888860
50388888888861
50388888888862
50388888888863
50388888888864
50388888888865
50388888888866
50388888888867
50388888888868
50388888888869
50388888888870
50388888888871
50388888888872
50388888888873
50388888888874
50388888888875
50388888888876
50388888888877
50388888888878
50388888888879
50388888888880
50388888888881
50388888888882
50388888888883
50388888888884
50388888888885
50388888888886
50388888888887
50388888888888
50388888888889
503888888888810
503888888888811
503888888888812
503888888888813
503888888888814
503888888888815
503888888888816
503888888888817
503888888888818
503888888888819
503888888888820
503888888888821
503888888888822
503888888888823
503888888888824
503888888888825
503888888888826
503888888888827
503888888888828
503888888888829
503888888888830
503888888888831
503888888888832
503888888888833
503888888888834
503888888888835
503888888888836
503888888888837
503888888888838
503888888888839
503888888888840
503888888888841
503888888888842
503888888888843
503888888888844
503888888888845
503888888888846
503888888888847
503888888888848
503888888888849
503888888888850
503888888888851
503888888888852
503888888888853
503888888888854
503888888888855
503888888888856
503888888888857
503888888888858
503888888888859
503888888888860
503888888888861
503888888888862
503888888888863
503888888888864
503888888888865
503888888888866
503888888888867
503888888888868
503888888888869
503888888888870
503888888888871
503888888888872
503888888888873
503888888888874
503888888888875
503888888888876
503888888888877
503888888888878
503888888888879
503888888888880
503888888888881
503888888888882
503888888888883
503888888888884
503888888888885
503888888888886
503888888888887
503888888888888
503888888888889
5038888888888810
5038888888888811
5038888888888812
5038888888888813
5038888888888814
5038888888888815
5038888888888816
5038888888888817
5038888888888818
5038888888888819
5038888888888820
50
```

```

Statement Coverage for instance /top/DUT --
File FIFO.sv
 46                               if (!fifo_if.rst_n) begin
 47       1                           1742      fifo_if.underflow <= 0; // all flags should be resetted
 48       1                           1742      rd_ptr <= 0;
 49
 50                               end
 51       1                           2832      fifo_if.data_out <= mem[rd_ptr];
 52       1                           2832      rd_ptr <= rd_ptr + 1;
 53       1                           2832      fifo_if.underflow <= 0; // handling Underflow sequentially
 54
 55                               end
 56       1                           547       fifo_if.underflow <= 1; // handling Underflow sequentially
 57
 58                               end
 59
 60                               //Counter handling:
 61       1                           14007    always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
 62
 63       1                           1711      if (!fifo_if.rst_n) begin
 64
 65                               count <= 0;
 66
 67       1                           5275      end
 68
 69       1                           1849      else begin
 70
 71                               if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b10) && !fifo_if.full)
 72
 73                               count <= count + 1;
 74
 75                               else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b01) && !fifo_if.empty)
 76
 77                               count <= count - 1;
 78
 79                               //----->No handling for cases when both fifo_if.wr_en & fifo_if.rd_en are enabled
 80
 81                               else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.empty)

```

```

Statement Coverage for instance /top/DUT --
File FIFO.sv
 71                               else if ( {{fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.empty)
 72     1                         110      count <= count + 1;
 73                               else if ( {{fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)
 74     1                         209      count <= count - 1;
 75                               end
 76                               end
 77
 78     1                         8195    assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
 79     1                         8195    assign fifo_if.empty = (count == 0)? 1 : 0;
 80     1                         8195    assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
 81     1                         8195    assign fifo_if.almostempty = (count == 1)? 1 : 0;

=====
== Instance: /top
== Design Unit: work.top
=====
Statement Coverage:
  Enabled Coverage      Bins      Hits      Misses   Coverage
  -----      -----      -----      -----      -----
  Statements          4        4        0    100.00%
=====Statement Details=====
Statement Coverage for instance /top --
Line      Item            Count      Source
---      ---            ----      -----
File top.sv
 12           module top();
 13           //CLK Generation
 14           bit clk;
 15     1           always #1 clk = ~clk;
 15     2           34042
 16           //Interface Instantiation
 17           FIFO_if fifo_if(clk);

```

```

Statement Coverage for instance /top --
File top.sv
 17           FIFO_if fifo_if(clk);
 18           //DUT instantiation
 19           FIFO DUT(.fifo_if(fifo_if));
 20           //Assertion
 21           bind FIFO FIFO_SVA SVA(.fifo_if(fifo_if));
 22
 23           initial begin
 24               //                                     who can get access
 25               //saving func  data type of variable      which class      "known name"      key to call the variable back      what you will pass to the database
 26     1           1       uvm_config_db #(virtual FIFO_if)::set(    null      , "uvm_test_top"      , "FIFO_IF"      ,      fifo_if      );
 27     1           1       run_test("FIFO_test");

```

```

=====
== Instance: /FIFO_coverage_pkg
== Design Unit: work.FIFO_coverage_pkg
=====

Covergroup Coverage:
  Covergroups          1      na      na  100.00%
  Coverpoints/Crosses 16      na      na   na
  Covergroup Bins     68      68      0   100.00%
-----

Covergroup
          Metric    Goal    Bins Status
-----
```

Covergroup	Metric	Goal	Bins	Status
TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
covered/total bins:	68	68	-	
missing/total bins:	0	68	-	
% Hit:	100.00%	100	-	
Coverpoint read_en	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	13461	1	-	Covered
bin auto[1]	3559	1	-	Covered
Coverpoint write_en	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	9384	1	-	Covered
bin auto[1]	7636	1	-	Covered
Coverpoint write_ack	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	10862	1	-	Covered
bin auto[1]	6159	1	-	Covered
Coverpoint overflow	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15939	1	-	Covered
bin auto[1]	1082	1	-	Covered
Coverpoint underflow	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14461	1	-	Covered
bin auto[1]	2560	1	-	Covered
Coverpoint full	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14794	1	-	Covered
bin auto[1]	2227	1	-	Covered
Coverpoint empty	100.00%	100	-	Covered

TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
Coverpoint empty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14287	1	-	Covered
bin auto[1]	2734	1	-	Covered
Coverpoint almostempty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15095	1	-	Covered
bin auto[1]	1926	1	-	Covered
Coverpoint almostfull	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15259	1	-	Covered
bin auto[1]	1762	1	-	Covered
Cross cross_write_ack	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	884	1	-	Covered
bin <auto[1],auto[1],auto[0]>	266	1	-	Covered
bin <auto[0],auto[1],auto[1]>	5275	1	-	Covered
bin <auto[0],auto[1],auto[0]>	1211	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6975	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_write_ack	0		-	ZERO
Cross cross_overflow	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	209	1	-	Covered
bin <auto[1],auto[1],auto[0]>	941	1	-	Covered
bin <auto[0],auto[1],auto[1]>	873	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5613	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6975	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_overflow	0		-	ZERO
Cross cross_full	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[0],auto[1],auto[1]>	1473	1	-	Covered
bin <auto[0],auto[0],auto[1]>	754	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1150	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5013	1	-	Covered

TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
Cross cross_full	100.00%	100	-	Covered
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[0],auto[1],auto[1]>	1473	1	-	Covered
bin <auto[0],auto[0],auto[1]>	754	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1150	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5013	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6221	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_full	0		-	ZERO
Cross cross_underflow	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	110	1	-	Covered
bin <auto[0],auto[1],auto[1]>	994	1	-	Covered
bin <auto[1],auto[0],auto[1]>	437	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1019	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1040	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5492	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1972	1	-	Covered
bin <auto[0],auto[0],auto[0]>	5956	1	-	Covered
Cross cross_empty	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	57	1	-	Covered
bin <auto[0],auto[1],auto[1]>	338	1	-	Covered
bin <auto[1],auto[0],auto[1]>	755	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1583	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1093	1	-	Covered
bin <auto[0],auto[1],auto[0]>	6148	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1654	1	-	Covered
bin <auto[0],auto[0],auto[0]>	5392	1	-	Covered
Cross cross_almostempty	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	226	1	-	Covered
bin <auto[0],auto[1],auto[1]>	836	1	-	Covered
bin <auto[1],auto[0],auto[1]>	223	1	-	Covered
bin <auto[0],auto[0],auto[1]>	641	1	-	Covered
bin <auto[1],auto[1],auto[0]>	924	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5650	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2186	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6334	1	-	Covered

TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
Cross cross_almostfull	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	411	1	-	Covered
bin <auto[0],auto[1],auto[1]>	446	1	-	Covered
bin <auto[1],auto[0],auto[1]>	285	1	-	Covered
bin <auto[0],auto[0],auto[1]>	620	1	-	Covered
bin <auto[1],auto[1],auto[0]>	739	1	-	Covered
bin <auto[0],auto[1],auto[0]>	6040	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2124	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6355	1	-	Covered

## COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
covered/total bins:	68	68	-	
missing/total bins:	0	68	-	
% Hit:	100.00%	100	-	
Coverpoint read_en	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	13461	1	-	Covered
bin auto[1]	3559	1	-	Covered
Coverpoint write_en	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	9384	1	-	Covered
bin auto[1]	7636	1	-	Covered
Coverpoint write_ack	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	10862	1	-	Covered
bin auto[1]	6159	1	-	Covered
Coverpoint overflow	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15939	1	-	Covered
bin auto[1]	1082	1	-	Covered
Coverpoint underflow	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14461	1	-	Covered
bin auto[1]	2560	1	-	Covered
Coverpoint full	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14794	1	-	Covered
bin auto[1]	2227	1	-	Covered
Coverpoint empty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	14287	1	-	Covered
bin auto[1]	2734	1	-	Covered
Coverpoint almostempty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	

TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
Coverpoint almostempty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15095	1	-	Covered
bin auto[1]	1926	1	-	Covered
Coverpoint almostfull	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	15259	1	-	Covered
bin auto[1]	1762	1	-	Covered
Cross cross_write_ack	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	884	1	-	Covered
bin <auto[1],auto[1],auto[0]>	266	1	-	Covered
bin <auto[0],auto[1],auto[1]>	5275	1	-	Covered
bin <auto[0],auto[1],auto[0]>	1211	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6975	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_write_ack	0		-	ZERO
Cross cross_overflow	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	209	1	-	Covered
bin <auto[1],auto[1],auto[0]>	941	1	-	Covered
bin <auto[0],auto[1],auto[1]>	873	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5613	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6975	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_overflow	0		-	ZERO
Cross cross_full	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[0],auto[1],auto[1]>	1473	1	-	Covered
bin <auto[0],auto[0],auto[1]>	754	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1150	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5013	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2409	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6221	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin impossible_full	0		-	ZERO

TYPE /FIFO_coverage_pkg/FIFO_coverage/C_GP_1	100.00%	100	-	Covered
Cross cross_underflow	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	110	1	-	Covered
bin <auto[0],auto[1],auto[1]>	994	1	-	Covered
bin <auto[1],auto[0],auto[1]>	437	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1019	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1040	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5492	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1972	1	-	Covered
bin <auto[0],auto[0],auto[0]>	5956	1	-	Covered
Cross cross_empty	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	57	1	-	Covered
bin <auto[0],auto[1],auto[1]>	338	1	-	Covered
bin <auto[1],auto[0],auto[1]>	755	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1583	1	-	Covered
bin <auto[1],auto[1],auto[0]>	1093	1	-	Covered
bin <auto[0],auto[1],auto[0]>	6148	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1654	1	-	Covered
bin <auto[0],auto[0],auto[0]>	5392	1	-	Covered
Cross cross_almostempty	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	226	1	-	Covered
bin <auto[0],auto[1],auto[1]>	836	1	-	Covered
bin <auto[1],auto[0],auto[1]>	223	1	-	Covered
bin <auto[0],auto[0],auto[1]>	641	1	-	Covered
bin <auto[1],auto[1],auto[0]>	924	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5650	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2186	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6334	1	-	Covered
Cross cross_almostfull	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	411	1	-	Covered
bin <auto[0],auto[1],auto[1]>	446	1	-	Covered
bin <auto[1],auto[0],auto[1]>	285	1	-	Covered
bin <auto[0],auto[0],auto[1]>	620	1	-	Covered
bin <auto[1],auto[1],auto[0]>	739	1	-	Covered
bin <auto[0],auto[1],auto[0]>	6040	1	-	Covered
bin <auto[1],auto[0],auto[0]>	2124	1	-	Covered
bin <auto[0],auto[0],auto[0]>	6355	1	-	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:

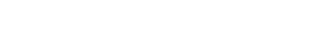
Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status
/top/DUT/cover__RD_PTR_RESET	FIFO	Verilog	SVA	FIFO.sv(163)	898	Covered
/top/DUT/cover__WR_PTR_RESET	FIFO	Verilog	SVA	FIFO.sv(162)	898	Covered
/top/DUT/cover__COUNT_RESET	FIFO	Verilog	SVA	FIFO.sv(161)	898	Covered
/top/DUT/cover__RD_PTR_WRAPAROUND	FIFO	Verilog	SVA	FIFO.sv(160)	139	Covered
/top/DUT/cover__WR_PTR_WRAPAROUND	FIFO	Verilog	SVA	FIFO.sv(159)	424	Covered
/top/DUT/cover__PTR_EXCEEDED	FIFO	Verilog	SVA	FIFO.sv(158)	16123	Covered
/top/DUT/cover__WR_ACK	FIFO	Verilog	SVA	FIFO.sv(157)	5815	Covered
/top/DUT/cover__ALMOST_EMPTY_FLAG	FIFO	Verilog	SVA	FIFO.sv(156)	16123	Covered
/top/DUT/cover__ALMOST_FULL_FLAG	FIFO	Verilog	SVA	FIFO.sv(155)	16123	Covered
/top/DUT/cover__EMPTY_FLAG	FIFO	Verilog	SVA	FIFO.sv(154)	16123	Covered
/top/DUT/cover__FULL_FLAG	FIFO	Verilog	SVA	FIFO.sv(153)	16123	Covered
/top/DUT/SVA/cover__RD_PTR_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(99)	898	Covered
/top/DUT/SVA/cover__WR_PTR_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(98)	898	Covered
/top/DUT/SVA/cover__COUNT_RESET	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(97)	898	Covered
/top/DUT/SVA/cover__RD_PTR_WRAPAROUND	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(96)	139	Covered
/top/DUT/SVA/cover__WR_PTR_WRAPAROUND	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(95)	424	Covered
/top/DUT/SVA/cover__PTR_EXCEEDED	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(94)	16123	Covered
/top/DUT/SVA/cover__WR_ACK	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(93)	5815	Covered
/top/DUT/SVA/cover__ALMOST_EMPTY_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(92)	16123	Covered
/top/DUT/SVA/cover__ALMOST_FULL_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(91)	16123	Covered
/top/DUT/SVA/cover__EMPTY_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(90)	16123	Covered
/top/DUT/SVA/cover__FULL_FLAG	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(89)	16123	Covered
/top/DUT/SVA/cover__UNDERFLOW	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(86)	522	Covered
/top/DUT/SVA/cover__OVERFLOW	FIFO_SVA	Verilog	SVA	FIFO_sva.sv(85)	1030	Covered

TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 24

## ASSERTION RESULTS:

Name	File(Line)	Failure Count	Pass Count	Vacuous Count	Disable Count	Attempt Count	Active Count	Peak Active Count	ATV
/top/DUT/assert__RD_PTR_WRAPAROUND	FIFO.sv(150)	0	139	15974	908	17021	0	1	off
/top/DUT/assert__WR_PTR_WRAPAROUND	FIFO.sv(149)	0	424	15676	921	17021	0	1	off
/top/DUT/assert__PTR_EXCEEDED	FIFO.sv(146)	0	16123	0	898	17021	0	1	off
/top/DUT/assert__WR_ACK	FIFO.sv(143)	0	5815	9964	1241	17021	1	2	off
/top/DUT/assert__RD_PTR_RESET	FIFO.sv(141)	0	898	16123	0	17021	0	1	off
/top/DUT/assert__WR_PTR_RESET	FIFO.sv(140)	0	898	16123	0	17021	0	1	off
/top/DUT/assert__COUNT_RESET	FIFO.sv(139)	0	898	16123	0	17021	0	1	off
/top/DUT/assert__ALMOST_EMPTY_FLAG	FIFO.sv(138)	0	16123	898	0	17021	0	1	off
/top/DUT/assert__ALMOST_FULL_FLAG	FIFO.sv(137)	0	16123	898	0	17021	0	1	off
/top/DUT/assert__EMPTY_FLAG	FIFO.sv(136)	0	16123	898	0	17021	0	1	off
/top/DUT/assert__FULL_FLAG	FIFO.sv(135)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__RD_PTR_WRAPAROUND	FIFO_sva.sv(82)	0	139	15974	908	17021	0	1	off
/top/DUT/SVA/assert__WR_PTR_WRAPAROUND	FIFO_sva.sv(81)	0	424	15676	921	17021	0	1	off
/top/DUT/SVA/assert__PTR_EXCEEDED	FIFO_sva.sv(78)	0	16123	0	898	17021	0	1	off
/top/DUT/SVA/assert__WR_ACK	FIFO_sva.sv(75)	0	5815	9964	1241	17021	1	2	off
/top/DUT/SVA/assert__RD_PTR_RESET	FIFO_sva.sv(73)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__WR_PTR_RESET	FIFO_sva.sv(72)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__COUNT_RESET	FIFO_sva.sv(71)	0	898	16123	0	17021	0	1	off
/top/DUT/SVA/assert__ALMOST_EMPTY_FLAG	FIFO_sva.sv(70)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__ALMOST_FULL_FLAG	FIFO_sva.sv(69)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__EMPTY_FLAG	FIFO_sva.sv(68)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__FULL_FLAG	FIFO_sva.sv(67)	0	16123	898	0	17021	0	1	off
/top/DUT/SVA/assert__UNDERFLOW	FIFO_sva.sv(63)	0	522	15576	923	17021	0	2	off
/top/DUT/SVA/assert__OVERFLOW	FIFO_sva.sv(62)	0	1030	15041	950	17021	0	2	off

## Covergroups Report:

/FIFO_coverage_pkg/FIFO_coverage		100.00%	100	100.00...		✓
-	TYPE C_GP_1	100.00%	100	100.00...		✓
+	CVP C_GP_1::read_en	100.00%	100	100.00...		✓
+	CVP C_GP_1::write_en	100.00%	100	100.00...		✓
+	CVP C_GP_1::write_ack	100.00%	100	100.00...		✓
+	CVP C_GP_1::overflow	100.00%	100	100.00...		✓
+	CVP C_GP_1::underflow	100.00%	100	100.00...		✓
+	CVP C_GP_1::full	100.00%	100	100.00...		✓
+	CVP C_GP_1::empty	100.00%	100	100.00...		✓
+	CVP C_GP_1::almostempty	100.00%	100	100.00...		✓
+	CVP C_GP_1::almostfull	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_write_ack	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_overflow	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_full	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_underflow	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_empty	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_almostempty	100.00%	100	100.00...		✓
+	CROSS C_GP_1::cross_almostfull	100.00%	100	100.00...		✓