FIFO Verification

Using UVM and Assertions

Ahmed Mohamed Abdelatif
Supervisor: Eng. Kareem Wassem

Table of Contents

FIFO Design

1) FIFO Overview

A FIFO (First-In, First-Out) is a specialized type of memory or buffer that allows data to be written into and read out in the same order it was entered. This design is frequently used in hardware systems where data must be processed in the order it arrives. It serves as a queue where the first element written is the first one to be read. FIFOs are crucial for managing data flow between two subsystems that operate at different speeds, such as when synchronizing communication between processors or between hardware modules.

Key Features of FIFO

Data Storage and Flow: Data is written into the FIFO through an input port and read out via an output port. The system ensures that data is removed in the order it was added, making it ideal for managing data streams.

Synchronous Operation: This design is synchronous, meaning that it operates with a clock signal (clk). All data transfers (write and read operations) happen on clock edges, ensuring synchronization between different parts of the system.

FIFO Width and Depth:

- **FIFO_WIDTH: The width of the data bus, which determines how many bits are written and read at once.**
- **FIFO_DEPTH: The total number of memory locations available in the FIFO, determining its storage capacity.**

Typical use cases

1. **Data Rate Matching: FIFOs are often used to smooth out differences in data rates between components that operate at different speeds.**
2. **Interfacing Different Systems: It is common in systems where data is transmitted between processors and peripherals, such as in communication devices, networking equipment, or audio/video processing.**
3. **Pipeline Buffers: In pipelined architectures, FIFOs are used to store intermediate results or data while the system continues processing other tasks.**

How it works

The FIFO works by utilizing two main operations: write and read. The flow is controlled by a series of signals:

1. **Write Operation:**
   - **Data is presented on the data_in input.**
   - **When the wr_en (write enable) signal is asserted and the FIFO is not full, data is written into the next available memory location.**
   - **If the FIFO becomes full, indicated by the full flag, no further write operations are allowed until space is freed.**
2. **Read Operation:**
   - **Data is read from the FIFO through the data_out port.**
   - **When the rd_en (read enable) signal is asserted and the FIFO is not empty, the oldest data is removed from the FIFO and presented at the output.**
   - **If the FIFO becomes empty, indicated by the empty flag, no more data can be read until more is written.**
3. **Overflow and Underflow:**
   - **Overflow: When a write is attempted but the FIFO is full, the overflow signal is asserted, and the new data is discarded to avoid corruption.**
   - **Underflow: When a read is attempted but the FIFO is empty, the underflow signal is asserted, and no data is read.**
4. **Status Signals:**
   - **Full/Empty: Indicate when the FIFO cannot accept more data (full) or when there is no data to read (empty).**
   - **Almost Full/Almost Empty: These intermediate flags (almostfull and almostempty) provide early warnings when the FIFO is about to become full or empty, allowing for better control over data flow.**
   - **Write Acknowledge (wr_ack): Indicates a successful write operation.**

In this design, assertions are added to verify correct FIFO behavior and ensure data integrity during write and read operations. The system is robust, handling edge cases like overflow and underflow gracefully.

2) Specs

Parameters

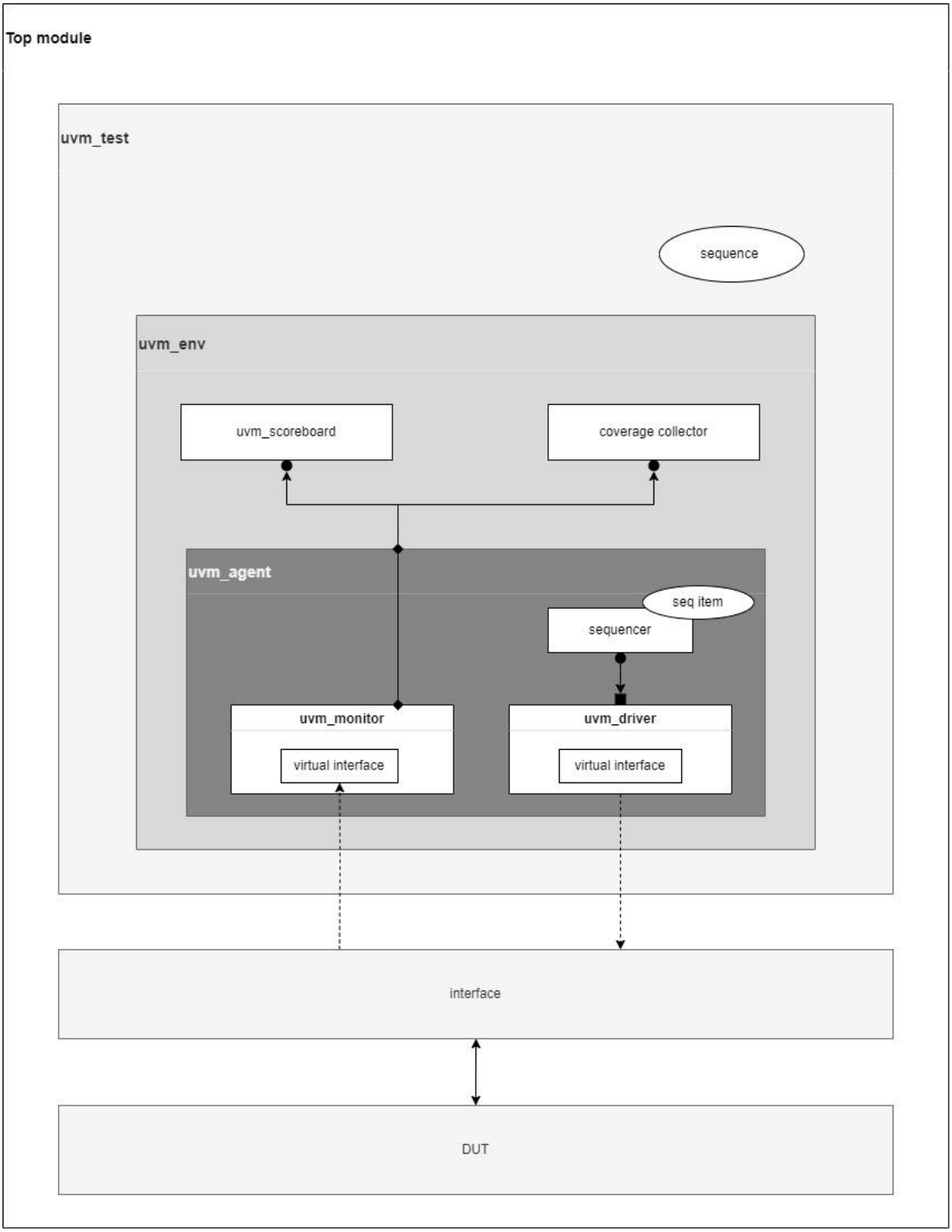
- **FIFO_WIDTH: DATA in/out and memory word width (default: 16)**
- **FIFO_DEPTH: Memory depth (default: 8)**

ports

Port	Direction	Function
data_in	Input	Write Data: The input data bus used when writing the FIFO.
wr_en		Write Enable: If the FIFO is not full, asserting this signal causes data (on data_in) to be written into the FIFO
rd_en		Read Enable: If the FIFO is not empty, asserting this signal causes data (on data_out) to be read from the FIFO
clk		Clock signal
rst_n		Active low asynchronous reset
data_out	Output	Read Data: The sequential output data bus used when reading from the FIFO.
full		Full Flag: When asserted, this combinational output signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
almostfull		Almost Full: When asserted, this combinational output signal indicates that only one more write can be performed before the FIFO is full.
empty		Empty Flag: When asserted, this combinational output signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.
almostempty		Almost Empty: When asserted, this output combinational signal indicates that only one more read can be performed before the FIFO goes to empty.
overflow		Overflow: This sequential output signal indicates that a write request (wr_en) was rejected because the FIFO is full. Overflowing the FIFO is not destructive to the contents of the FIFO.
underflow		Underflow: This sequential output signal Indicates that the read request (rd_en) was rejected because the FIFO is empty. Under flowing the FIFO is not destructive to the FIFO.
wr_ack		Write Acknowledge: This sequential output signal indicates that a write request (wr_en) has succeeded.

Note: If a read and write enables were high and the FIFO was empty, only writing will take place and vice versa if the FIFO was full.

3) UVM structure



Description

- *First, we assign the "design", and "interface", and bind the "SVA" in the "top" module, the we set the virtual interface and give an access to the "test_pkg".*

- *In the "test" in "build_phase" we get the virtual interface we set from "top", then* **create an object block and assign the virtual interface in it, after that we set the** *block, in "run_phase" we raise an objection and run the "reset_sequence" and the "main_sequence" after they done we drop the objection.*

- *In the "environment" first in "build_phase" we create the "agent", "scoreboard", and "coverage collector", and in "connect_phase" we connect "scoreboard", and "coverage collector" (2 ports, each port* **has a FIFO to save the data from the** *export) with the created connection named"agt_ap"(1 export) by using "analysis_port"*

- *In the "agent" in "build_phase" we create the "sequencer", "driver", "monitor", and the analysis connection, after that in "connect_phase" we connect the "sequencer" to the "driver" with a normal connection (1 port, 1 export), connect* **the driver to the interface, and connect the monitor to the interface and to the** *"agt_ap"*

- *In the "diver" we take the randomized input variables with s*ome constraints from *the "sequence_item" and assign these variables to the interface by the connection we created in the "agent".*

- *In the "monitor" we take the randomized variables (that the "driver" sent to the* **interface) from the interface and assign these variables and the outputs given** *from interface to the "sequence_item" (we made this step because we hadn't have the output), after we assign all variables we sent it to the "agt_ap" that is connected to the "scoreboard", and "coverage collector".*

- **In** *"scoreboard" in "run_phase" we compare the output with the expected value calculated from a function named "ref_model" and increment the error count if* **there is an error and do the same with the correct count.**

- *In "coverage collector" we created a covergr***oup to check the functionality of the design.**

Notes:

- *In "scoreboard" and "coverage collector" we should create an analysis_export* **node and create an analysis_fifo to save the values from the export.**
- *In "monitor" we should create an analysis port becaus***e it is the one which send e***very "analysis port" or "single port" should be connected in the topper pkg in the "connect_phase"***.**

4) Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	when the reset is asserted the ouput and internal counters should be low	Directed at the start of simlation then randomized during simulation with constrain to be low most of the time		immediate assertion and a checker in the scoreboard to check for reset functionality
FIFO_2	when reset is de asserted and write signal is asserted and the FIFO is not full then the value of data_in writed in the write pointer location and the write_ack flag will be high and the counter wil increase	Randomized during simulation with constrains on write signal to be high 70% of time	cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and increasing internal counters
FIFO_3	when reset is de asserted and read signal is asserted and the FIFO is not empty then data_out will take the value in the read pointer location and the counter wil decrease	Randomized during simulation with constrains on the read signal to be high 30% of time	cover group use cross coverage between write, read and output flags	a checker of the data_out value in the scoreboard and cocurrent assertion to check the output flags and decreasing internal counters
FIFO_4	when count is equal to the FIFO depth the full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and increasing internal
FIFO_5	when count is equal 0 depth the full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_6	when count is equal to the FIFO depth-1 the almost full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_7	when count is equal to 1 the almost empty flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_8	when reset is deasserted and write signal is asserted and the FIFO is full then overflow flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_9	when reset is deasserted and read signal is asserted and the FIFO is empty then underflow flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters

5) Bug Report

Detected Bugs:

- **The almostfull flag is raised when there are 2 empty places not 1.**

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

- **When resetting the overflow, underflow** *doesn't reset*.
- **Underflow signal is supposed to be sequential.**

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

- **The counter handling was missing the possibility that the read and write signal could be high together so we should handle this.**

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

6) The design

Code without bugs + assertions + interface:

Design

```
module FIFO (data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty,
wr_ack, overflow, underflow, data_out);



    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    input [FIFO_WIDTH-1:0] data_in;
    input clk, rst_n, wr_en, rd_en;
    output reg [FIFO_WIDTH-1:0] data_out;
    output reg wr_ack, overflow, underflow;
    output full, empty, almostfull, almostempty;

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

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


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

Interface

```
   interface FIFO_if (input clk);

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    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;

endinterface : FIFO_if
```

Assertions

```
        // Assertions
        // full flag
        assert property (@(posedge clk) (count == FIFO_DEPTH) |-> full );
        cover property (@(posedge clk) (count == FIFO_DEPTH) |-> full );
        // empty flag
        assert property (@(posedge clk) (count == 0) |-> empty );
        cover property (@(posedge clk) (count == 0) |-> empty );
        // almost full flag
        assert property (@(posedge clk) (count == FIFO_DEPTH-1) |-> almostfull );
        cover property (@(posedge clk) (count == FIFO_DEPTH-1) |-> almostfull);
        // almost empty flag
        assert property (@(posedge clk) (count == 1) |-> almostempty );
        cover property (@(posedge clk) (count == 1) |-> almostempty );
        // Over flow flag
  assert property (@(posedge clk) disable iff(!rst_n) (full && wr_en) |=> (overflow));
  cover property (@(posedge clk) disable iff(!rst_n) (full && wr_en) |=> (overflow));
        // Under flow flag
  assert property (@(posedge clk) disable iff(!rst_n) (empty && rd_en) |=> (underflow));
  cover property (@(posedge clk) disable iff(!rst_n) (empty && rd_en) |=> (underflow));
        // Write acknoledge flag
  assert property (@(posedge clk) disable iff(!rst_n) (wr_en && !full) |=> (wr_ack));
  cover property (@(posedge clk) disable iff(!rst_n) (wr_en && !full) |=> (wr_ack));
        // internal counters
        // Write pointer
     assert property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && wr_ptr!=7)
|=> (wr_ptr==$past(wr_ptr)+1));
     cover property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && wr_ptr!=7)
|=> (wr_ptr==$past(wr_ptr)+1));
        // Write pointer if we wrote in the 8 places we will return to the beginning
      assert property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && wr_ptr==7)
|=> (wr_ptr==0));
      cover property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && wr_ptr==7)
|=> (wr_ptr==0));
        // Read pointer
     assert property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && rd_ptr!=7)
|=> (rd_ptr==$past(rd_ptr)+1));
     cover property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && rd_ptr!=7)
|=> (rd_ptr==$past(rd_ptr)+1));
        // Read pointer if we read the 8 places we will return to the beginning
     assert property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && rd_ptr==7)
|=> (rd_ptr==0));
     cover property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && rd_ptr==7)
|=> (rd_ptr==0));
        // counter
```

```
        // counter
        assert property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && !rd_en)
|=> (count==$past(count)+1));
        cover property (@(posedge clk) disable iff(!rst_n) (wr_en && !full && !rd_en)
|=> (count==$past(count)+1));
        assert property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && !wr_en)
|=> (count==$past(count)-1));
        cover property (@(posedge clk) disable iff(!rst_n) (rd_en && !empty && !wr_en)
|=> (count==$past(count)-1));
        assert property (@(posedge clk) disable iff(!rst_n) (wr_en && empty && rd_en)
|=> (count==$past(count)+1));
        cover property (@(posedge clk) disable iff(!rst_n) (wr_en && empty && rd_en) |=>
(count==$past(count)+1));
        assert property (@(posedge clk) disable iff(!rst_n) (wr_en && full && rd_en) |=>
(count==$past(count)-1));
        cover property (@(posedge clk) disable iff(!rst_n) (wr_en && full && rd_en) |=>
(count==$past(count)-1));
```

7) Testbench

The UVM Environment:

FIFO test

```
package FIFO_test_;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_env_::*;
import FIFO_config_obj_::*;
import FIFO_main_sequence_::*;
import FIFO_reset_seq_::*;
import FIFO_agent_::*;
import FIFO_seq_item_::*;

    class FIFO_test extends  uvm_test;

    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        FIFO_env env;
        FIFO_config_obj FIFO_config_obj_test;
        FIFO_main_sequence main_seq;
        FIFO_reset_seq rst_seq;
        FIFO_agent agt;

    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_test)

    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_test", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new
```

```
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = FIFO_env::type_id::create("env",this);
            agt = FIFO_agent::type_id::create("agt",this);
            FIFO_config_obj_test =
FIFO_config_obj::type_id::create("FIFO_config_obj_test");
            rst_seq = FIFO_reset_seq::type_id::create("rst_seq");
            main_seq = FIFO_main_sequence::type_id::create("main_seq");

            if (!uvm_config_db#(virtual FIFO_if)::get(this, "",
"FIFO_if",FIFO_config_obj_test.FIFO_config_vif )) begin
                `uvm_fatal("build_phase","test can not get the interface")
            end
            uvm_config_db#(FIFO_config_obj)::set(this, "*",
"FIFO_vif_CO",FIFO_config_obj_test );
        endfunction

        // Run phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("run_phase","reset asserted",UVM_MEDIUM)
            rst_seq.start(agt.seq);
            `uvm_info("run_phase","reset desserted",UVM_MEDIUM)
```

```
            `uvm_info("run_phase","rand starrted",UVM_MEDIUM)
            main_seq.start(agt.seq);
            `uvm_info("run_phase","rand ended",UVM_MEDIUM)
            phase.drop_objection(this);
        endtask

    endclass : FIFO_test
endpackage : FIFO_test_
```

FIFO Seq item

```
package FIFO_seq_item_;
import shared_package::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
    class FIFO_seq_item extends  uvm_sequence_item;
    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;
        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;
    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_object_utils(FIFO_seq_item)
    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_seq_item");
            super.new(name);
        endfunction : new

        function string convert2string_sim();
            return $sformatf("rst_n=%b ,data_in=%b ,wr_en=%b ,rd_en=%b ,data_out=%b "
                ,rst_n,data_in,wr_en,rd_en,data_out);
        endfunction
    /*-----------------------------------------------------------------------------
    -- Constrain
    -----------------------------------------------------------------------------*/
        constraint c {
            rst_n dist {1:/95 , 0:/5};
            wr_en dist {1:/WR_EN_ON_DIST , 0:/(100-WR_EN_ON_DIST)};
            rd_en dist {1:/RD_EN_ON_DIST , 0:/(100-RD_EN_ON_DIST)};
        }

    endclass : FIFO_seq_item
```

FIFO main sequence

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

    class FIFO_main_sequence extends  uvm_sequence #(FIFO_seq_item);
    /*------------------------------------------------------------------------------
    -- Interface, port, fields
    ------------------------------------------------------------------------------*/
        FIFO_seq_item seq_item;

    /*------------------------------------------------------------------------------
    -- UVM Factory register
    ------------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_object_utils(FIFO_main_sequence)

    /*------------------------------------------------------------------------------
    -- Functions
    ------------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_main_sequence");
            super.new(name);
        endfunction : new

        task body();
            // Write only
            repeat (1000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.wr_en = 1;
                seq_item.rd_en= 0;
                seq_item.rst_n = $random();
                seq_item.data_in = $random();
                finish_item(seq_item);
            end
```

```
            // Read only
            repeat (1000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.wr_en = 0;
                seq_item.rd_en= 1;
                seq_item.rst_n = $random();
                seq_item.data_in = $random();
                finish_item(seq_item);
            end
            // Write - Read
            repeat (1000) begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body

    endclass : FIFO_main_sequence
endpackage : FIFO_main_sequence_
```

FIFO reset sequence

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

    class FIFO_reset_seq extends uvm_sequence #(FIFO_seq_item);
    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        FIFO_seq_item seq_item;
```

```
    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_object_utils(FIFO_reset_seq)

    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_reset_seq");
            super.new(name);
        endfunction : new

        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 : body

    endclass : FIFO_reset_seq
endpackage : FIFO_reset_seq_
```

Sequencer:

```
package FIFO_sequencer_;
    import FIFO_seq_item_::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_sequencer extends  uvm_sequencer #(FIFO_seq_item);
    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_sequencer)

    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_sequencer", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new

    endclass : FIFO_sequencer
endpackage : FIFO_sequencer_
```

FIFO env:

```
package FIFO_env_;
import FIFO_driver_::*;
import uvm_pkg::*;
import FIFO_sequencer_::*;
import FIFO_coverage_::*;
import FIFO_score_board_::*;
import FIFO_agent_::*;
`include "uvm_macros.svh"

    class FIFO_env extends  uvm_env;
    /*-------------------------------------------------------------------------------
    -- Interface, port, fields
    -------------------------------------------------------------------------------*/
        FIFO_agent agt;
        FIFO_coverage cov;
        FIFO_score_board sb;
    /*-------------------------------------------------------------------------------
    -- UVM Factory register
    -------------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_env)

    /*-------------------------------------------------------------------------------
    -- Functions
    -------------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_env", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new

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

```
        // Connect phase
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agt.agt_ap.connect(cov.cov_ex);
            agt.agt_ap.connect(sb.sb_ex);
        endfunction
    endclass : FIFO_env

endpackage : FIFO_env_
```

FIFO Agent:

```
package FIFO_agent_;
    import FIFO_driver_::*;
    import uvm_pkg::*;
    import FIFO_sequencer_::*;
    import FIFO_monitor_::*;
    import FIFO_config_obj_::*;
    import FIFO_seq_item_::*;
    `include "uvm_macros.svh"

    class FIFO_agent extends uvm_agent;
    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        FIFO_sequencer seq;
        FIFO_config_obj co;
        FIFO_monitor mon;
        FIFO_driver drv;
        uvm_analysis_port #(FIFO_seq_item) agt_ap;
    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_agent)
    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
```

FIFO Driver:

```
package FIFO_driver_;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_config_obj_::*;
    import FIFO_seq_item_::*;

    class FIFO_driver extends uvm_driver #(FIFO_seq_item);
    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        virtual FIFO_if FIFO_driver_vif;
        FIFO_seq_item seq_item;
        FIFO_config_obj co;
    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_driver)
    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_driver", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new
        // Build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
        endfunction
        // Run phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                seq_item_port.get_next_item(seq_item);
                FIFO_driver_vif.rst_n = seq_item.rst_n;
                FIFO_driver_vif.data_in = seq_item.data_in;
                FIFO_driver_vif.rd_en = seq_item.rd_en;
                FIFO_driver_vif.wr_en = seq_item.wr_en;
                @(negedge FIFO_driver_vif.clk);
                seq_item_port.item_done();
            end
        endtask
```

FIFO Monitor:

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

    class FIFO_monitor extends  uvm_monitor;

    /*--------------------------------------------------------------------------------
    -- Interface, port, fields
    --------------------------------------------------------------------------------*/
        virtual FIFO_if FIFO_vif;
        FIFO_seq_item seq_item;
        uvm_analysis_port #(FIFO_seq_item) mon_ap;

    /*--------------------------------------------------------------------------------
    -- UVM Factory register
    --------------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_monitor)

    /*--------------------------------------------------------------------------------
    -- Functions
    --------------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_monitor", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new

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

        // Run phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                seq_item = FIFO_seq_item::type_id::create("seq_item");
                seq_item.rst_n = FIFO_vif.rst_n;
                seq_item.wr_en = FIFO_vif.wr_en;
                seq_item.rd_en = FIFO_vif.rd_en;
                seq_item.data_in = FIFO_vif.data_in;
```

```
                seq_item.wr_ack = FIFO_vif.wr_ack;
                seq_item.overflow = FIFO_vif.overflow;
                seq_item.underflow = FIFO_vif.underflow;
                seq_item.almostfull = FIFO_vif.almostfull;
                seq_item.almostempty = FIFO_vif.almostempty;
                seq_item.data_out = FIFO_vif.data_out;
                seq_item.full = FIFO_vif.full;
                seq_item.empty = FIFO_vif.empty;
                @(negedge FIFO_vif.clk);
                mon_ap.write(seq_item);
                `uvm_info("run phase",seq_item.convert2string(),UVM_HIGH)
            end
        endtask

    endclass : FIFO_monitor
endpackage : FIFO_monitor_
```

FIFO Scoreboard:

```
package FIFO_score_board_;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_seq_item_::*;
import shared_package::*;
```

```
    class FIFO_score_board extends uvm_scoreboard;
    /*------------------------------------------------------------------------------
    -- Interface, port, fields
    ------------------------------------------------------------------------------*/
        FIFO_seq_item seq_item;
        uvm_analysis_export #(FIFO_seq_item) sb_ex;
        uvm_tlm_analysis_fifo #(FIFO_seq_item) sb_fifo;

        // Outs to compare with
        logic [FIFO_WIDTH-1:0] data_out_ref;
        logic wr_ack_ref, overflow_ref;
        logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;
    /*------------------------------------------------------------------------------
    -- UVM Factory register
    ------------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_score_board)

    /*------------------------------------------------------------------------------
    -- Functions
    ------------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_score_board", uvm_component parent=null);
            super.new(name, parent);
        endfunction : new

        // Build phase
        function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        seq_item = FIFO_seq_item::type_id::create("seq_item");
        sb_ex = new("sb_ex",this);
        sb_fifo = new("sb_fifo",this);
        endfunction

        // Connect phase
        function void connect_phase(uvm_phase phase);
```

```
        super.connect_phase(phase);
        sb_ex.connect(sb_fifo.analysis_export);
    endfunction

    // Run phase
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            sb_fifo.get(seq_item);
            reference_model(seq_item);
            if (seq_item.data_out!=data_out_ref) begin
                `uvm_error("run_phase", $sformatf("The DUT gives data_out:%s
                    while the Golden model gives data_out : %b "
                ,seq_item.data_out,data_out_ref));
                error_count++;
            end else begin
                correct_count++;
            end
        end
    endtask

    function reference_model(FIFO_seq_item seq_item);

        logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
        logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
        logic [max_fifo_addr:0] count;

        if (!seq_item.rst_n) begin
            wr_ptr = 0;
            rd_ptr = 0;
            count = 0;
            underflow_ref = 0;
            overflow_ref = 0;
        end
        else begin
            /* Read operation before write operation to read the right value
            when read and write signals asserted together*/

            if (seq_item.rd_en && count != 0) begin
                data_out_ref = mem[rd_ptr];
                rd_ptr = rd_ptr + 1;
                count = count - 1;
            end else begin
```

```
                    if (empty_ref && seq_item.rd_en)
                        underflow_ref = 1;
                    else
                        underflow_ref = 0;
                end

                if (seq_item.wr_en && !full_ref) begin
                    mem[wr_ptr] = seq_item.data_in;
                    wr_ack_ref = 1;
                    wr_ptr = wr_ptr + 1;
                    count = count + 1;
                end
                else begin
                    wr_ack_ref = 0;
                    if (full_ref & seq_item.wr_en)
                        overflow_ref = 1;
                    else
                        overflow_ref = 0;
                end
            end


            if (count == FIFO_DEPTH)
                full_ref = 1;
            else
                full_ref = 0;

            if (count == 0)
                empty_ref = 1;
            else
                empty_ref = 0;

            if (count == FIFO_DEPTH-1)
                almostfull_ref = 1;
            else
                almostfull_ref = 0;

            if (count == 1)
                almostempty_ref = 1;
            else
                almostempty_ref = 0;
        endfunction

        endclass : FIFO_score_board
endpackage : FIFO_score_board
```

FIFO Coverage:

```
package FIFO_coverage_;
import uvm_pkg::*;
`include "uvm_macros.svh"
import FIFO_seq_item_::*;
import shared_package::*;

    class FIFO_coverage extends uvm_component;

    /*------------------------------------------------------------------------------
    -- Interface, port, fields
    ------------------------------------------------------------------------------*/
        FIFO_seq_item seq_item;
        uvm_analysis_export #(FIFO_seq_item) cov_ex;
        uvm_tlm_analysis_fifo #(FIFO_seq_item) cov_fifo;

    /*------------------------------------------------------------------------------
    -- UVM Factory register
    ------------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_component_utils(FIFO_coverage)

    /*------------------------------------------------------------------------------
    -- Functions
    ------------------------------------------------------------------------------*/

        // Cover groups
        covergroup cg;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.wr_ack;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.full;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.empty;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.almostfull;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.almostempty;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.overflow;
            cross seq_item.wr_en , seq_item.rd_en , seq_item.underflow;
        endgroup : cg

        // Constructor
        function new(string name = "FIFO_coverage", uvm_component parent=null);
            super.new(name, parent);
            cg = new;
        endfunction : new

        // Build phase
```

```
        // Build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            seq_item = FIFO_seq_item::type_id::create("seq_item");
            cov_ex = new("cov_ex",this);
            cov_fifo = new("cov_fifo",this);
        endfunction

        // Connect phase
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cov_ex.connect(cov_fifo.analysis_export);
        endfunction

        // Run phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item);
                cg.sample();
            end
        endtask

    endclass : FIFO_coverage
endpackage : FIFO_coverage_
```

Configuration object:

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

    class FIFO_config_obj extends  uvm_object;

    /*-----------------------------------------------------------------------------
    -- Interface, port, fields
    -----------------------------------------------------------------------------*/
        virtual FIFO_if FIFO_config_vif;

    /*-----------------------------------------------------------------------------
    -- UVM Factory register
    -----------------------------------------------------------------------------*/
        // Provide implementations of virtual methods such as get_type_name and create
        `uvm_object_utils(FIFO_config_obj)

    /*-----------------------------------------------------------------------------
    -- Functions
    -----------------------------------------------------------------------------*/
        // Constructor
        function new(string name = "FIFO_config_obj");
            super.new(name);
        endfunction : new

    endclass : FIFO_config_obj

endpackage : FIFO_config_obj_
```

The wrapper code:

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

```
    bit clk;
    initial begin
        clk=0;
        forever begin
            #2 clk=~clk;
        end
    end

    FIFO_if F (clk);
    FIFO dut (F.data_in, F.wr_en, F.rd_en, F.clk, F.rst_n, F.full, F.empty,
F.almostfull,
        F.almostempty, F.wr_ack, F.overflow, F.underflow, F.data_out);

    initial begin
        uvm_config_db#(virtual FIFO_if)::set(null,"uvm_test_top","FIFO_if",F);
        run_test("FIFO_test");
    end
endmodule : FIFO_top
```

8) Do file

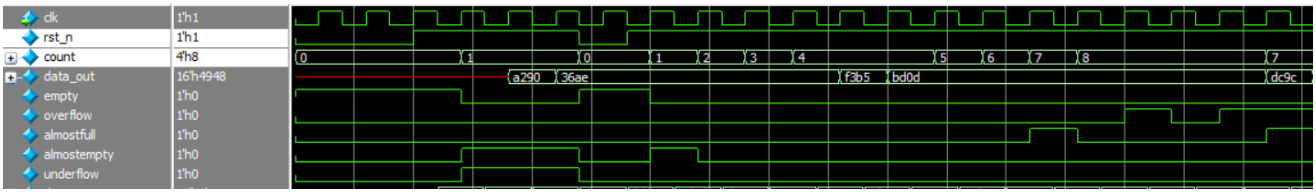
```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.FIFO_top -classdebug -uvmcontrol=all -cover
add wave /FIFO_top/F/*
coverage save FIFO_top.ucdb -onexit
run -all
```

The src_file.list:

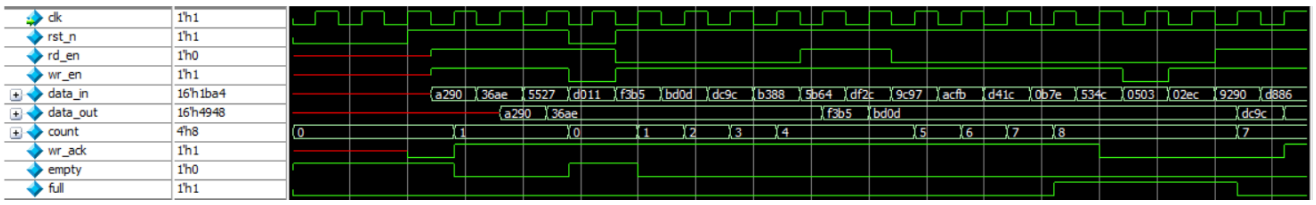
```
shared_package.sv
FIFO_if.sv
FIFO_dut.sv
FIFO_seq_item.sv
FIFO_coverage.sv
FIFO_score_board.sv
FIFO_sequencer.sv
FIFO_main_sequence.sv
FIFO_reset_seq.sv
FIFO_config_obj.sv
FIFO_driver.sv
FIFO_agent.sv
FIFO_env.sv
FIFO_test.sv
FIFO_top.sv
```

9) Simulation

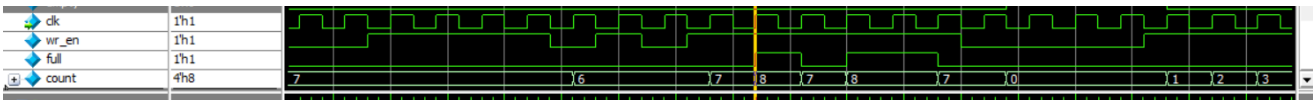
FIFO_1:



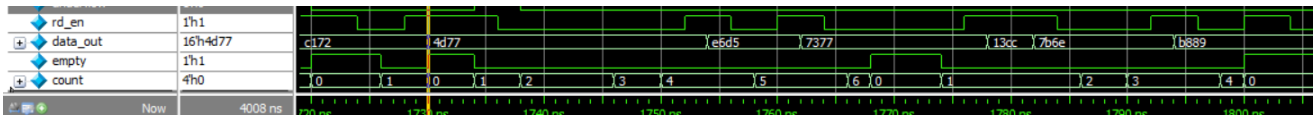
FIFO_2, FIFO_3:



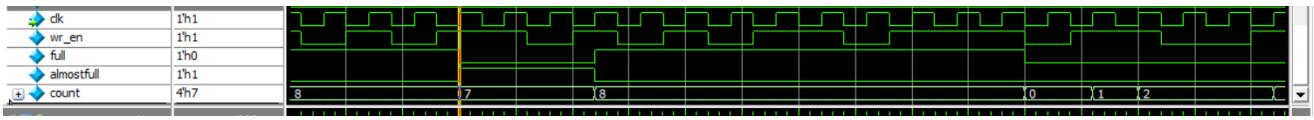
FIFO_4:



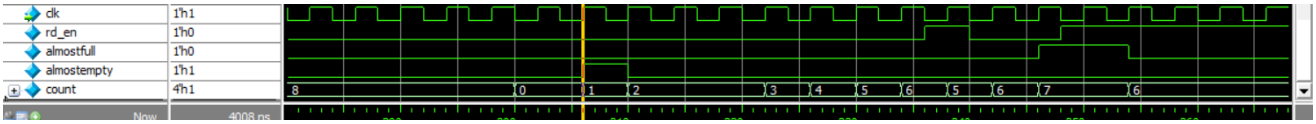
FIFO_5:



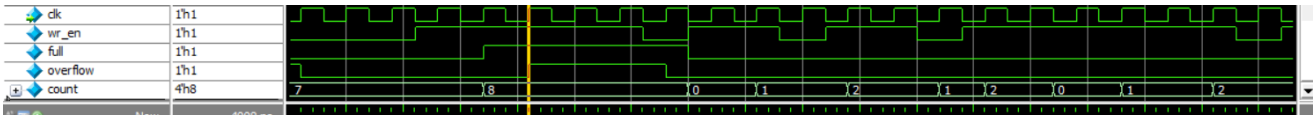
FIFO_6:



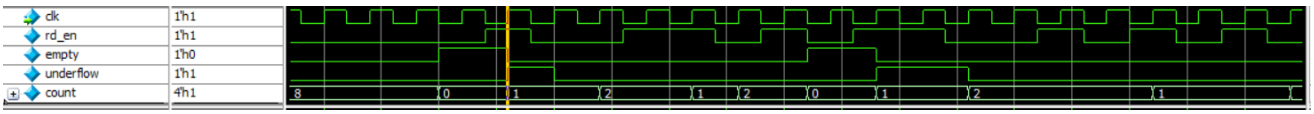
FIFO_7:



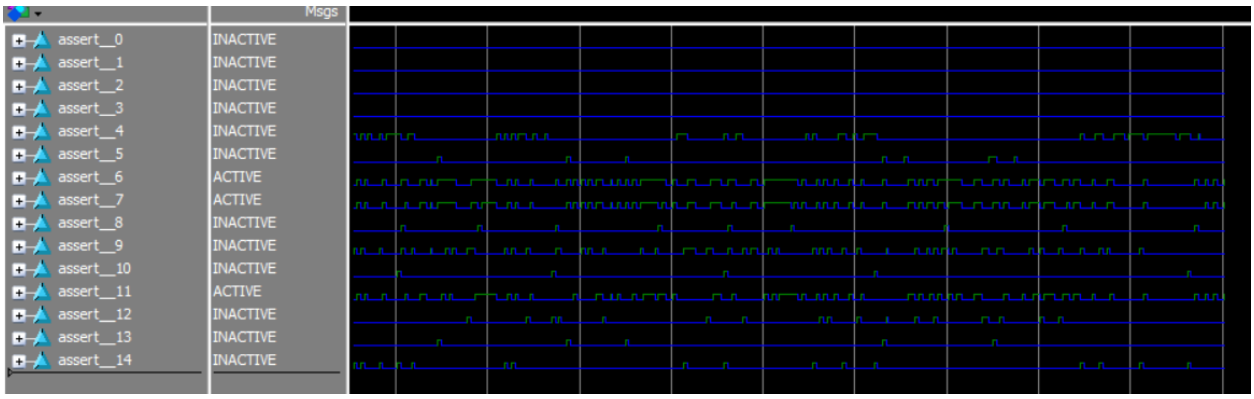
FIFO_8:



FIFO_9:



Assertions:



10) Coverage report

Assertions coverage

```
Assertion Coverage:
    Assertions                      15        15         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/FIFO_top/dut/assert__14
                     FIFO_dut.sv(122)                    0          1
/FIFO_top/dut/assert__13
                     FIFO_dut.sv(120)                    0          1
/FIFO_top/dut/assert__12
                     FIFO_dut.sv(118)                    0          1
/FIFO_top/dut/assert__11
                     FIFO_dut.sv(116)                    0          1
/FIFO_top/dut/assert__10
                     FIFO_dut.sv(113)                    0          1
/FIFO_top/dut/assert__9
                     FIFO_dut.sv(110)                    0          1
/FIFO_top/dut/assert__8
                     FIFO_dut.sv(107)                    0          1
/FIFO_top/dut/assert__7
                     FIFO_dut.sv(104)                    0          1
/FIFO_top/dut/assert__6
                     FIFO_dut.sv(100)                    0          1
/FIFO_top/dut/assert__5
                     FIFO_dut.sv(97)                     0          1
/FIFO_top/dut/assert__4
                     FIFO_dut.sv(94)                     0          1
/FIFO_top/dut/assert__3
                     FIFO_dut.sv(91)                     0          1
/FIFO_top/dut/assert__2
                     FIFO_dut.sv(88)                     0          1
/FIFO_top/dut/assert__1
                     FIFO_dut.sv(85)                     0          1
/FIFO_top/dut/assert__0
                     FIFO_dut.sv(82)                     0          1
```

Branch coverage

```
Branch Coverage:
    Enabled Coverage                Bins      Hits    Misses  Coverage
    ----------------                ----      ----    ------  --------
    Branches                          25        25         0   100.00%

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

Branch Coverage for instance /FIFO_top/dut

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File FIFO_dut.sv
-----------------------------------IF Branch-----------------------------------
    25                                    3552     Count coming in to IF
    25             1                      1576                if (!rst_n) begin

    29             1                       950                else if (wr_en && count < FIFO_DEPTH) begin

    34             1                      1026                else begin

Branch totals: 3 hits of 3 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    36                                    1026     Count coming in to IF
    36             1                       229                        if (full & wr_en)

    38             1                       797                        else

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    44                                    2709     Count coming in to IF
    44             1                      1104                if (!rst_n) begin

    48             1                       249                else if (rd_en && count != 0) begin

    51             1                      1356                end else begin

Branch totals: 3 hits of 3 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    52                                    1356     Count coming in to IF
    52             1                       401                        if (empty && rd_en)
```

```
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    60                                    2678     Count coming in to IF
    60              1                     1104                 if (!rst_n) begin

    63              1                     1574                 else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    64                                    1574     Count coming in to IF
    64              1                      836                          if       ( ({wr_en, rd_en} == 2'b10) && !full)

    66              1                       78                          else if ( ({wr_en, rd_en} == 2'b01) && !empty)

    68              1                       13                          else if (({wr_en, rd_en} == 2'b11) && empty)

    70              1                       70                          else if (({wr_en, rd_en} == 2'b11) && full)

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

------------------------------------IF Branch------------------------------------
    75                                    1297     Count coming in to IF
    75              1                      106        assign full = (count == FIFO_DEPTH)? 1 : 0;

    75              2                     1191        assign full = (count == FIFO_DEPTH)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    76                                    1297     Count coming in to IF
    76              1                      308        assign empty = (count == 0)? 1 : 0;

    76              2                      989        assign empty = (count == 0)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    77                                    1297     Count coming in to IF
    77              1                      135        assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
```

```
    77              2                       1162        assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    78                                      1297     Count coming in to IF
    78              1                        313        assign almostempty = (count == 1)? 1 : 0;

    78              2                        984        assign almostempty = (count == 1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%
```

Directive coverage

```
Directive Coverage:
    Directives                      15         15          0    100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------
Name                                Design Design   Lang File(Line)      Hits Status
                                    Unit   UnitType
--------------------------------------------------------------------------------------
/FIFO_top/dut/cover__14             FIFO   Verilog  SVA  FIFO_dut.sv(123)  63 Covered
/FIFO_top/dut/cover__13             FIFO   Verilog  SVA  FIFO_dut.sv(121)  13 Covered
/FIFO_top/dut/cover__12             FIFO   Verilog  SVA  FIFO_dut.sv(119)  71 Covered
/FIFO_top/dut/cover__11             FIFO   Verilog  SVA  FIFO_dut.sv(117) 568 Covered
/FIFO_top/dut/cover__10             FIFO   Verilog  SVA  FIFO_dut.sv(114)  15 Covered
/FIFO_top/dut/cover__9              FIFO   Verilog  SVA  FIFO_dut.sv(111) 216 Covered
/FIFO_top/dut/cover__8              FIFO   Verilog  SVA  FIFO_dut.sv(108)  41 Covered
/FIFO_top/dut/cover__7              FIFO   Verilog  SVA  FIFO_dut.sv(105) 637 Covered
/FIFO_top/dut/cover__6              FIFO   Verilog  SVA  FIFO_dut.sv(101) 678 Covered
/FIFO_top/dut/cover__5              FIFO   Verilog  SVA  FIFO_dut.sv(98)  289 Covered
/FIFO_top/dut/cover__4              FIFO   Verilog  SVA  FIFO_dut.sv(95)  215 Covered
/FIFO_top/dut/cover__3              FIFO   Verilog  SVA  FIFO_dut.sv(92)  197 Covered
/FIFO_top/dut/cover__2              FIFO   Verilog  SVA  FIFO_dut.sv(89)  194 Covered
/FIFO_top/dut/cover__1              FIFO   Verilog  SVA  FIFO_dut.sv(86) 1871 Covered
/FIFO_top/dut/cover__0              FIFO   Verilog  SVA  FIFO_dut.sv(83)  326 Covered
```

Statement coverage

```
  Enabled Coverage                Bins      Hits    Misses  Coverage
    ----------------                ----      ----    ------  --------
    Statements                        26        26         0   100.00%

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

Statement Coverage for instance /FIFO_top/dut --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO_dut.sv
    8                                              module FIFO (data_in, wr_en, rd_en, clk, rst_n, full, empty,
almostfull, almostempty, wr_ack, overflow, underflow, data_out);

    9                                                  parameter FIFO_WIDTH = 16;

    10                                                 parameter FIFO_DEPTH = 8;

    11                                                 input [FIFO_WIDTH-1:0] data_in;

    12                                                 input clk, rst_n, wr_en, rd_en;

    13                                                 output reg [FIFO_WIDTH-1:0] data_out;

    14                                                 output reg wr_ack, overflow, underflow;

    15                                                 output full, empty, almostfull, almostempty;

    16

    17                                                 localparam max_fifo_addr = $clog2(FIFO_DEPTH);

    18

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

    20

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

    22                                                 reg [max_fifo_addr:0] count;
```

```
26              1                       1576                            wr_ptr <= 0;
27              1                       1576                            overflow <= 0;
28                                                              end
29                                                              else if (wr_en && count < FIFO_DEPTH) begin
30              1                        950                            mem[wr_ptr] <= data_in;
31              1                        950                            wr_ptr <= wr_ptr + 1;
32              1                        950                            wr_ack <= 1;
33                                                              end
34                                                              else begin
35              1                       1026                            wr_ack <= 0;
36                                                                      if (full & wr_en)
37              1                        229                                    overflow <= 1;
38                                                                      else
39              1                        797                                    overflow <= 0;
40                                                              end
41                                                      end
42
43              1                       2709            always @(posedge clk or negedge rst_n) begin
44                                                              if (!rst_n) begin
45              1                       1104                            rd_ptr <= 0;
46              1                       1104                            underflow <= 0;
47                                                              end
48                                                              else if (rd_en && count != 0) begin
49              1                        249                            data_out <= mem[rd_ptr];
```

```
  50             1                       249                           rd_ptr <= rd_ptr + 1;
   51                                                           end else begin
   52                                                                   if (empty && rd_en)
   53             1                       401                                   underflow <= 1;
   54                                                                   else
   55             1                       955                                   underflow <= 0;
   56                                                           end
   57                                                   end
   58
   59             1                       2678        always @(posedge clk or negedge rst_n) begin
   60                                                           if (!rst_n) begin
   61             1                       1104                  count <= 0;
   62                                                           end
   63                                                           else begin
   64                                                                   if      ( ({wr_en, rd_en} == 2'b10) &&
!full)
   65             1                       836                           count <= count + 1;
   66                                                                   else if ( ({wr_en, rd_en} == 2'b01) &&
!empty)
   67             1                       78                            count <= count - 1;
   68                                                                   else if (({wr_en, rd_en} == 2'b11) &&
empty)
   69             1                       13                            count <= count + 1;
   70                                                                   else if (({wr_en, rd_en} == 2'b11) && full)
   71             1                       70                            count <= count - 1;
```

```
72                                                      end
73                                                  end
74
75              1                       1298        assign full = (count == FIFO_DEPTH)? 1 : 0;
76              1                       1298        assign empty = (count == 0)? 1 : 0;
77              1                       1298        assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
78              1                       1298        assign almostempty = (count == 1)? 1 : 0;
```

Toggle coverage

```
Toggle Coverage:
    Enabled Coverage               Bins      Hits    Misses  Coverage
    ----------------               ----      ----    ------  --------
    Toggles                         106       106         0   100.00%

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

Toggle Coverage for instance /FIFO_top/dut --

                                        Node      1H->0L      0L->1H  "Coverage"
                                        ----------------------------------------
                                  almostempty           1           1     100.00
                                   almostfull           1           1     100.00
                                          clk           1           1     100.00
                                   count[3-0]           1           1     100.00
                                 data_in[0-15]          1           1     100.00
                                data_out[15-0]          1           1     100.00
                                         empty          1           1     100.00
                                          full          1           1     100.00
                                      overflow          1           1     100.00
                                         rd_en          1           1     100.00
                                   rd_ptr[2-0]          1           1     100.00
                                         rst_n          1           1     100.00
                                     underflow          1           1     100.00
                                        wr_ack          1           1     100.00
                                         wr_en          1           1     100.00
                                   wr_ptr[2-0]          1           1     100.00

Total Node Count     =         53
Toggled Node Count   =         53
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (106 of 106 bins)
```

Covergroup coverage

```
Covergroup Coverage:
    Covergroups                     1        na        na      100%
        Coverpoints/Crosses        28        na        na        na
            Covergroup Bins        92        92         0      100%
----------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status


----------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_/FIFO_coverage/cg              97.32%        100          -    Uncovered
    covered/total bins:                                 92         98          -
    missing/total bins:                                  6         98          -
    % Hit:                                          93.87%        100          -
    Coverpoint #seq_item.wr_en__0#                 100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    1291          1          -    Covered
        bin auto[1]                                    1708          1          -    Covered
    Coverpoint #seq_item.rd_en__1#                 100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    1712          1          -    Covered
        bin auto[1]                                    1287          1          -    Covered
    Coverpoint #seq_item.underflow__2#             100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    2454          1          -    Covered
        bin auto[1]                                     546          1          -    Covered
    Coverpoint #seq_item.wr_en__3#                 100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    1291          1          -    Covered
        bin auto[1]                                    1708          1          -    Covered
    Coverpoint #seq_item.rd_en__4#                 100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
```

% Hit:	100.00%	100	-	
bin auto[0]	1712	1	-	Covered
bin auto[1]	1287	1	-	Covered
Coverpoint #seq_item.overflow__5#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	2723	1	-	Covered
bin auto[1]	277	1	-	Covered
Coverpoint #seq_item.wr_en__6#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1291	1	-	Covered
bin auto[1]	1708	1	-	Covered
Coverpoint #seq_item.rd_en__7#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1712	1	-	Covered
bin auto[1]	1287	1	-	Covered
Coverpoint #seq_item.almostempty__8#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	2660	1	-	Covered
bin auto[1]	340	1	-	Covered
Coverpoint #seq_item.wr_en__9#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1291	1	-	Covered
bin auto[1]	1708	1	-	Covered
Coverpoint #seq_item.rd_en__10#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1712	1	-	Covered
bin auto[1]	1287	1	-	Covered

```
    Coverpoint #seq_item.almostfull__11#          100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    2794          1          -    Covered
    bin auto[1]                                     206          1          -    Covered
Coverpoint #seq_item.wr_en__12#                 100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    1291          1          -    Covered
    bin auto[1]                                    1708          1          -    Covered
Coverpoint #seq_item.rd_en__13#                 100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    1712          1          -    Covered
    bin auto[1]                                    1287          1          -    Covered
Coverpoint #seq_item.empty__14#                 100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    1429          1          -    Covered
    bin auto[1]                                    1571          1          -    Covered
Coverpoint #seq_item.wr_en__15#                 100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    1291          1          -    Covered
    bin auto[1]                                    1708          1          -    Covered
Coverpoint #seq_item.rd_en__16#                 100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
    bin auto[0]                                    1712          1          -    Covered
    bin auto[1]                                    1287          1          -    Covered
Coverpoint #seq_item.full__17#                  100.00%        100          -    Covered
    covered/total bins:                             2          2          -
    missing/total bins:                             0          2          -
    % Hit:                                      100.00%        100          -
```

Coverpoint #seq_item.wr_en__18#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1291	1	-	Covered
bin auto[1]	1708	1	-	Covered
Coverpoint #seq_item.rd_en__19#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1712	1	-	Covered
bin auto[1]	1287	1	-	Covered
Coverpoint #seq_item.wr_ack__20#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	1540	1	-	Covered
bin auto[1]	1458	1	-	Covered
Cross #cross__0#	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]>	119	1	-	Covered
bin <auto[0],auto[1],auto[1]>	2	1	-	Covered
bin <auto[1],auto[0],auto[1]>	1334	1	-	Covered
bin <auto[0],auto[0],auto[1]>	3	1	-	Covered
bin <auto[1],auto[1],auto[0]>	77	1	-	Covered
bin <auto[0],auto[1],auto[0]>	1089	1	-	Covered
bin <auto[1],auto[0],auto[0]>	177	1	-	Covered
bin <auto[0],auto[0],auto[0]>	197	1	-	Covered
Cross #cross__1#	100.00%	100	-	Covered
covered/total bins:	6	8	-	
missing/total bins:	2	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[0],auto[1]>	265	1	-	Covered
bin <auto[0],auto[0],auto[1]>	71	1	-	Covered
bin <auto[1],auto[1],auto[0]>	196	1	-	Covered

```
        bin <auto[0],auto[1],auto[0]>              1091          1          -    Covered
        bin <auto[1],auto[0],auto[0]>              1247          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               129          1          -    Covered
        bin <*,auto[1],auto[1]>                       0          1          2    ZERO
Cross #cross__2#                                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]>               12          1          -    Covered
        bin <auto[0],auto[1],auto[1]>              1021          1          -    Covered
        bin <auto[1],auto[0],auto[1]>               518          1          -    Covered
        bin <auto[0],auto[0],auto[1]>                19          1          -    Covered
        bin <auto[1],auto[1],auto[0]>               184          1          -    Covered
        bin <auto[0],auto[1],auto[0]>                70          1          -    Covered
        bin <auto[1],auto[0],auto[0]>               994          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               181          1          -    Covered
Cross #cross__3#                                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]>               96          1          -    Covered
        bin <auto[0],auto[1],auto[1]>               26          1          -    Covered
        bin <auto[1],auto[0],auto[1]>               39          1          -    Covered
        bin <auto[0],auto[0],auto[1]>               45          1          -    Covered
        bin <auto[1],auto[1],auto[0]>              100          1          -    Covered
        bin <auto[0],auto[1],auto[0]>              1065          1          -    Covered
        bin <auto[1],auto[0],auto[0]>              1473          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               155          1          -    Covered
Cross #cross__4#                                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]>               24          1          -    Covered
        bin <auto[0],auto[1],auto[1]>                5          1          -    Covered
        bin <auto[1],auto[0],auto[1]>              294          1          -    Covered
        bin <auto[0],auto[0],auto[1]>               17          1          -    Covered
        bin <auto[1],auto[1],auto[0]>              172          1          -    Covered
        bin <auto[0],auto[1],auto[0]>              1086          1          -    Covered
        bin <auto[1],auto[0],auto[0]>              1218          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               183          1          -    Covered
Cross #cross__5#                                  100%        100          -    Covered
```

```
    covered/total bins:                              6          8          -
    missing/total bins:                              2          8          -
    % Hit:                                        100%        100          -    Covered
    Auto, Default and User Defined Bins:
        bin <auto[1],auto[1],auto[1]>               82          1          -    Covered
        bin <auto[1],auto[0],auto[1]>               195         1          -    Covered
        bin <auto[1],auto[1],auto[0]>               114         1          -    Covered
        bin <auto[0],auto[1],auto[0]>               1091        1          -    Covered
        bin <auto[1],auto[0],auto[0]>               1317        1          -    Covered
        bin <auto[0],auto[0],auto[0]>               200         1          -    Covered
        bin <auto[0],*,auto[1]>                     0           1          2
Cross #cross__6#                                    100%        100          -
    covered/total bins:                              6          8          -
    missing/total bins:                              2          8          -
    % Hit:                                        100%        100          -
    Auto, Default and User Defined Bins:
        bin <auto[1],auto[1],auto[1]>               17          1          -    Covered
        bin <auto[0],auto[1],auto[1]>               529         1          -    Covered
        bin <auto[1],auto[1],auto[0]>               179         1          -    Covered
        bin <auto[0],auto[1],auto[0]>               562         1          -    Covered
        bin <auto[1],auto[0],auto[0]>               1512        1          -    Covered
        bin <auto[0],auto[0],auto[0]>               200         1          -    Covered
```