FIFO Verification

supervisor: Eng. Kareem Wassem
Name: Ahmed Essam Sayed Zayed

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) The design

Detected Bugs

- The almostfull was “depth-2” correct “depth-1”
  - Overflow and underflow were not have a value in case of reset.
  - Counter wasn’t covering all the cases of the wr_en and rd_en.
  - Underflow should be sequential not compinational.

Code without bugs

```
////////////////////////////////////////////////////////////////////////////////
//
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO 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;
    end
    else if (wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        wr_ack <= 1;
```

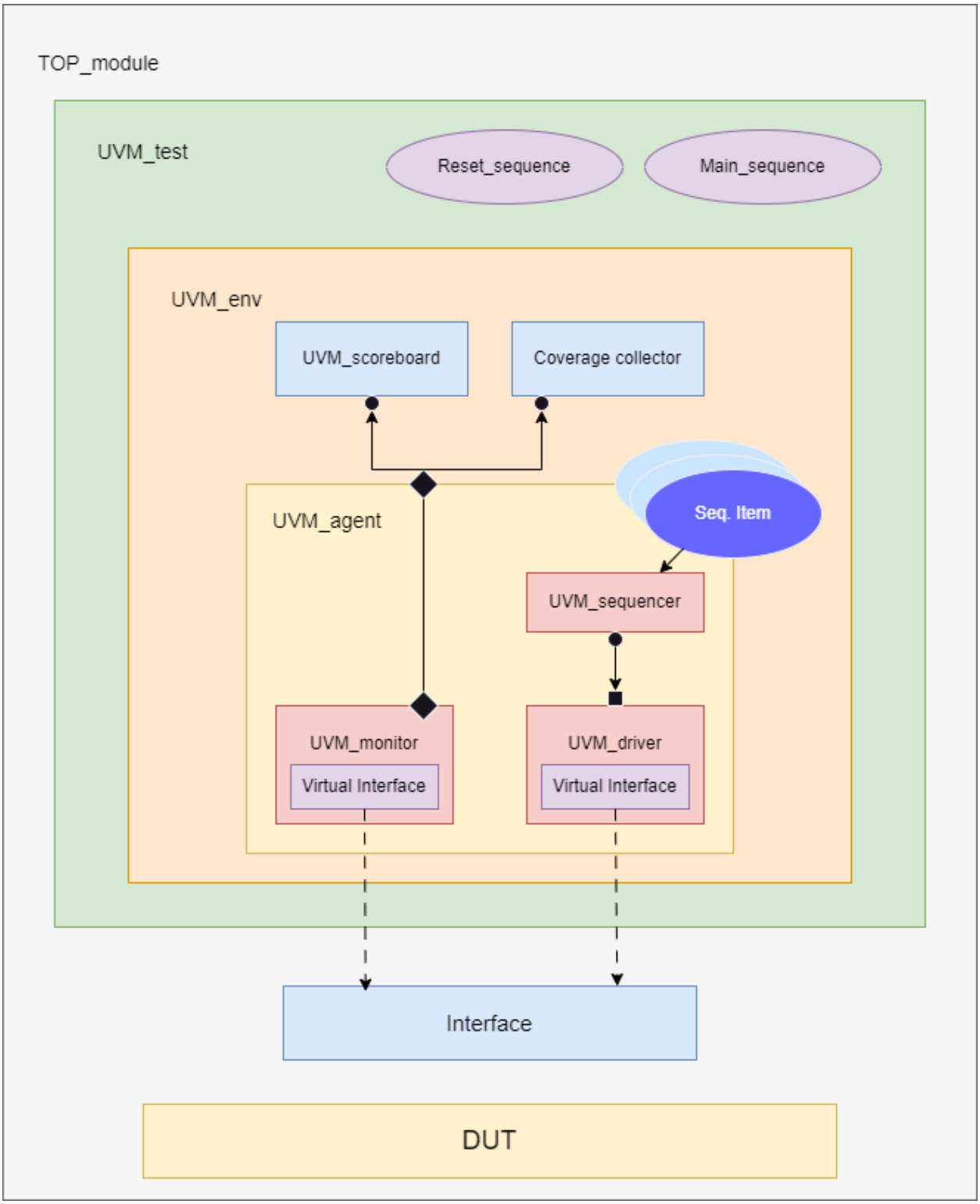
```
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        wr_ack <= 0;
        if (full && wr_en)  //was "&" correction is "&&"
            overflow <= 1;
        else
            overflow <= 0;
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rd_ptr <= 0;
    end
    else if (rd_en && count != 0) begin
        data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin                      //remove underflow from assert statements
because it is sequential output and add this else statement
        if (empty && rd_en) begin
            underflow <= 1;
        end
        else begin
            underflow <= 0;
        end
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
    //added this statement to cover all cases  ex: if wr_en and rd_en both
equal to 1 and the fifo is empty the code will do neither this "(({wr_en,
rd_en} == 2'b10) && !full)" nor this "(({wr_en, rd_en} == 2'b01) && !empty)"
however we can write in fifo and increase the counter normally
        if (({wr_en, rd_en} == 2'b11)) begin
            if (count==0) begin
                count<=count+1;
            end
            else if (count==FIFO_DEPTH) begin
                count<=count-1;
```

```
            end
        end
        else begin
            if  ( (wr_en == 1) && !full)
                count <= count + 1;
            else if ( (rd_en == 1) && !empty)
                count <= count - 1;
        end
    end
end

assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //was "FIFO_DEPTH-2"
correction "FIFO_DEPTH-1" only
assign almostempty = (count == 1)? 1 : 0;

endmodule
```

4) Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check	Feature	Assertion
FIFO_1	When the reset is asserted, overflow, underflow, and wr_ack should be low also the counter, wr_ptr, and rd_ptr return to zero.	Directed at the start of the sim, then randomized with constraint that drive the reset to be off most of the simulation time.	-	checking using immediate assertion in the SVA module.	whenever rst_n is asserted, empty=1, full, count, rd_ptr, and wr_ptr=0.	immediate if (!rst_n) assert final (!full && empty && (count==0) && (rd_ptr==0) && (wr_ptr==0));
FIFO_2	(write_only_sequence) because the depth is 8, after the 8th loop we can't write anymore and the output should be don't care throghout this repeat loop	assign the reset to be deasserted, and assign the wr_en to 1, and rd_en to 0, and randomize data_in.	Cover all the possible cross coverages when wr_en is on or off.	A checker in the FIFO_scoreboard_pack to make sure the output is correct, checking using some immediate and concurrent assertions in the SVA module to check on "count", "wr_ptr", "full", "almost_full", "wr_ack", and "overflow".	whenever (count==FIFO_DEPTH), full=1.	immediate if (count==(FIFO_DEPTH)) assert final (full);
					whenever (count==FIFO_DEPTH-1), almost_full=1.	immediate if (count==(FIFO_DEPTH-1)) assert final (almost_full);
					whenever rst_n is disabled ((count==FIFO_DEPTH)&&(wr_en)), overflow=1, in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) (((count==FIFO_DEPTH)&&(wr_en))|=> (overflow);
					whenever rst_n is disabled ((count!=FIFO_DEPTH)&&(wr_en)), wr_ack=1, in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) (((count!=FIFO_DEPTH)&&(wr_en))|=> (wr_ack));
					whenever rst_n is disabled (wr_en&&!rd_en&&!full), count should be incremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&!rd_en&&!full)|=> (count==$past(count)+1));
					whenever rst_n is disabled (wr_en&&!rd_en&&full), count should be stable in next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&!rd_en&&full)|=> ($stable(count)));
					whenever rst_n is disabled (wr_en&&!full&&(wr_ptr<7)), wr_ptr should be incremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&!full&&(wr_ptr<7))|=> (wr_ptr==$past(wr_ptr)+1))
					whenever rst_n is disabled (wr_en&&!full&&(wr_ptr==7)), wr_ptr should be 0.	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&!full&&(wr_ptr==7))|=> (wr_ptr==0));
					whenever rst_n is disabled (wr_en&&full), wr_en should be stable in next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&full)|=> ($stable(wr_ptr)));
FIFO_3	(read_only_sequence) because the depth is 8, the out will read the data we write previosly and after the 8th loop the out should be fixed to the last value	assign the reset to be deasserted, and assign the wr_en to 0, and rd_en to 1, and randomize data_in.	Cover all the possible cross coverages when rd_en is on or off.	A checker in the FIFO_scoreboard_pack to make sure the output is correct, checking using some immediate and concurrent assertions in the SVA module to check on "count", "rd_ptr", "empty", "almost_empty", and "underflow".	whenever (count==0), empty=1.	immediate if (count==(0)) assert final (empty);
					whenever (count==1), almost_empty=1.	immediate if (count==(1)) assert final (almost_empty);
					whenever rst_n is disabled ((count==0)&&(rd_en)), underflow=1, in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) (((count==0)&&(rd_en))|=> (underflow));
					whenever rst_n is disabled (!wr_en&&rd_en&&!empty), count should be decremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((!wr_en&&rd_en&&!empty)|=> (count==$past(count)-1));
					whenever rst_n is disabled (!wr_en&&rd_en&&empty), count should be stable in next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((!wr_en&&rd_en&&empty)|=> ($stable(count)));
					whenever rst_n is disabled (rd_en&&!empty&&(rd_ptr<7)), rd_ptr should be incremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((rd_en&&!empty&&(rd_ptr<7))|=> (rd_ptr==$past(rd_ptr)+1));
					whenever rst_n is disabled (rd_en&&!empty&&(rd_ptr==7)), rd_ptr should be 0.	Concurrent @(posedge clk) disable iff(!rst_n) ((rd_en&&!empty&&(rd_ptr==7))|=> (rd_ptr==0));
					whenever rst_n is disabled (rd_en&&empty), rd_en should be stable in next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((rd_en&&empty)|=> ($stable(rd_ptr)));
FIFO_4	(write_read_sequence) with every write the the out will read the write that preceded it except for the first read because from the previos loop the mem was empty	assign the reset to be deasserted, and assign the wr_en to 1, and rd_en to 1, and randomize data_in.	Cover all the possible cross coverages when rd_en and wr_en is on or off.	A checker in the FIFO_scoreboard_pack to make sure the output is correct, checking using some immediate and concurrent assertions in the SVA module to check on "count", "wr_ptr", "full", "almost_full", "wr_ack", "overflow", "rd_ptr", "empty", "almost_empty", and "underflow".	(in addition to all the previos write only and read only assertions) whenever rst_n is disabled (wr_en&&rd_en&&!full&&!empty), count should be stable in next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&!full&&!empty)|=> ($stable(count)));
					whenever rst_n is disabled (wr_en&&rd_en&&full&&!empty), count should be decremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&full&&!empty)|=> (count==$past(count)-1));
					whenever rst_n is disabled (wr_en&&rd_en&&!full&&empty), count should be incremented in the next clk	Concurrent @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&!full&&empty)|=> (count==$past(count)+1));
FIFO_5	(random_sequence) To check more cases.	Randomization under constraints on the wr_en signal to be on with the value of WR_EN_ON_DIST and off with value of 100-WR_EN_ON_DIST, and on the rd_en signal to be on with the value of RD_EN_ON_DIST and off with value of 100-RD_EN_ON_DIST.	Same as FIFO_4	Same as FIFO_4	All the previos assertins including the reset assertion	-

5) 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 some 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 covergroup to check the functionality of the design.

Note:

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 because it is the one which send.

Every “analysis port” or “single port” should be connected in the topper pkg in the “connect_phase”

6) Interface

```
interface FIFO_inter (clk);

    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_inter
```

7) SVA

```
module FIFO_SVA (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 clk;
    input [FIFO_WIDTH-1:0] data_in;
    input rst_n, wr_en, rd_en;
    input [FIFO_WIDTH-1:0] data_out;
    input wr_ack, overflow;
    input full, empty, almostfull, almostempty, underflow;

    localparam max_fifo_addr = $clog2(FIFO_DEPTH);
    logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    logic [max_fifo_addr:0] count;

//for write and read pointers and count only
    always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
        wr_ptr <= 0;
        rd_ptr <= 0;
    end
    else begin
        if (({wr_en, rd_en} == 2'b11)) begin
            if (count==0) begin
                count<=count+1;
                wr_ptr<=wr_ptr+1;
            end
            else if (count==FIFO_DEPTH) begin
                count<=count-1;
                rd_ptr<=rd_ptr+1;
            end
            else begin
                wr_ptr<=wr_ptr+1;
                rd_ptr<=rd_ptr+1;
            end
        end
        else begin
            if  ( (wr_en == 1) && !full)  begin
                count <= count + 1;
                wr_ptr<=wr_ptr+1;
```

```
            end
            else if ( (rd_en == 1) && !empty) begin
                count <= count - 1;
                rd_ptr<=rd_ptr+1;
            end
        end
    end
end

//Assertions
always_comb begin
    if (count==FIFO_DEPTH) begin
        assert_full: assert final (full);
        full_c: cover final (full);
    end
    if (count==(FIFO_DEPTH-1)) begin
        assert_almostfull: assert final (almostfull);
        almostfull_c: cover final (almostfull);
    end
    if (count==0) begin
        assert_empty: assert final (empty);
        empty_c: cover final (empty);
    end
    if (count==1) begin
        assert_almostempty: assert final (almostempty);
        almostempty_c: cover final (almostempty);
    end
    if (!rst_n) begin
        assert_reset: assert final (!full && empty && (count==0) && (rd_ptr==0)
&& (wr_ptr==0));
        reset_c: cover final (!full && empty && (count==0) && (rd_ptr==0) &&
(wr_ptr==0));
    end
end

property assert_overflow;
    @(posedge clk) disable iff(!rst_n) (((count==FIFO_DEPTH)&&(wr_en))|=>
(overflow));  //sequential output signal
endproperty
property assert_underflow;
    @(posedge clk) disable iff(!rst_n) (((count==0)&&(rd_en))|=>
(underflow));  //sequential output signal
endproperty
property assert_wr_ack;
```

```
    @(posedge clk) disable iff(!rst_n) (((count!=FIFO_DEPTH)&&(wr_en))|=>
(wr_ack));    //sequential output signal
endproperty
property assert_count_not_max;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&!full&&!empty)|=>
($stable(count)));
endproperty
property assert_count_max;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&full&&!empty)|=>
(count==$past(count)-1));
endproperty
property assert_count_ZERO;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&rd_en&&!full&&empty)|=>
(count==$past(count)+1));
endproperty
property assert_count_write_not_max;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&!rd_en&&!full)|=>
(count==$past(count)+1));
endproperty
property assert_count_write_max;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&!rd_en&&full)|=>
($stable(count)));
endproperty
property assert_count_read_not_max;
    @(posedge clk) disable iff(!rst_n) ((!wr_en&&rd_en&&!empty)|=>
(count==$past(count)-1));
endproperty
property assert_count_read_max;
    @(posedge clk) disable iff(!rst_n) ((!wr_en&&rd_en&&empty)|=>
($stable(count)));
endproperty
property assert_wr_ptr_not_full;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&!full&&(wr_ptr<7))|=>
(wr_ptr==$past(wr_ptr)+1));
endproperty
property assert_wr_ptr_not_full_max;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&!full&&(wr_ptr==7))|=>
(wr_ptr==0));
endproperty
property assert_wr_ptr_full;
    @(posedge clk) disable iff(!rst_n) ((wr_en&&full)|=> ($stable(wr_ptr)));
endproperty
property assert_rd_ptr_not_full;
    @(posedge clk) disable iff(!rst_n) ((rd_en&&!empty&&(rd_ptr<7))|=>
(rd_ptr==$past(rd_ptr)+1));
```

```
endproperty
property assert_rd_ptr_not_full_max;
    @(posedge clk) disable iff(!rst_n) ((rd_en&&!empty&&(rd_ptr==7))|=>
(rd_ptr==0));
endproperty
property assert_rd_ptr_full;
    @(posedge clk) disable iff(!rst_n) ((rd_en&&empty)|=> ($stable(rd_ptr)));
endproperty

assert property (assert_overflow);
cover property (assert_overflow);

assert property (assert_underflow);
cover property (assert_underflow);

assert property (assert_wr_ack);
cover property (assert_wr_ack);

assert property (assert_count_not_max);
cover property (assert_count_not_max);

assert property (assert_count_max);
cover property (assert_count_max);

assert property (assert_count_ZERO);
cover property (assert_count_ZERO);

assert property (assert_count_write_not_max);
cover property (assert_count_write_not_max);

assert property (assert_count_write_max);
cover property (assert_count_write_max);

assert property (assert_count_read_not_max);
cover property (assert_count_read_not_max);

assert property (assert_count_read_max);
cover property (assert_count_read_max);

assert property (assert_wr_ptr_not_full);
cover property (assert_wr_ptr_not_full);

assert property (assert_wr_ptr_not_full_max);
cover property (assert_wr_ptr_not_full_max);
```

```
assert property (assert_wr_ptr_full);
cover property (assert_wr_ptr_full);

assert property (assert_rd_ptr_not_full);
cover property (assert_rd_ptr_not_full);

assert property (assert_rd_ptr_not_full_max);
cover property (assert_rd_ptr_not_full_max);

assert property (assert_rd_ptr_full);
cover property (assert_rd_ptr_full);

endmodule : FIFO_SVA
```

8) Packages code

FIFO_config_pkg

```
package FIFO_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_config extends  uvm_object;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_object_utils(FIFO_config)

        virtual FIFO_inter FIFO_vif;

        // Constructor
        function new(string name = "FIFO_config");
            super.new(name);
        endfunction : new

    endclass : FIFO_config
endpackage : FIFO_config_pkg
```

FIFO_sequence_item_pkg

```
package FIFO_sequence_item_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class FIFO_seq_item extends  uvm_sequence_item;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_object_utils(FIFO_seq_item)

        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;

        logic clk;

        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;

        integer RD_EN_ON_DIST, WR_EN_ON_DIST;

        constraint reset_c {
            rst_n dist {1:/95, 0:/5};
        }
        constraint wr_c {
            wr_en dist {1:/WR_EN_ON_DIST, 0:/(100-WR_EN_ON_DIST)};
        }
        constraint rd_c {
            rd_en dist {1:/RD_EN_ON_DIST, 0:/(100-RD_EN_ON_DIST)};
        }

        // Constructor
        function new(string name = "FIFO_seq_item", integer wr_in_dist=70,
integer rd_in_dist=30);
            super.new(name);
            RD_EN_ON_DIST=rd_in_dist;
            WR_EN_ON_DIST=wr_in_dist;
        endfunction : new

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

```
                rst_n, wr_en, rd_en, data_in, wr_ack, overflow, full, empty,
almostfull, almostempty, underflow, data_out);
        endfunction : convert2string

    endclass : FIFO_seq_item
endpackage : FIFO_sequence_item_pkg
```

FIFO_main_sequence_pkg

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

    class FIFO_main_sequence extends  uvm_sequence #(FIFO_seq_item);
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_object_utils(FIFO_main_sequence)

        FIFO_seq_item seq_item;

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

        task body();
//FIFO_2(write_only_sequence)
            repeat (10) begin
                seq_item=FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n=1;
                seq_item.wr_en=1;
                seq_item.rd_en=0;
                seq_item.data_in=$random();
                finish_item(seq_item);
            end
            //after the 8th loop we can't write anymore and the output should
be don't care throghout this repeat loop

//FIFO_3(read_only_sequence)
            repeat (10) begin
                seq_item=FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n=1;
                seq_item.wr_en=0;
                seq_item.rd_en=1;
                seq_item.data_in=$random();
                finish_item(seq_item);
            end
            //the out will read the data we write previosly and after the 8th
loop the out will be fixed to the last value
```

```
//FIFO_4(write_read_sequence)
            repeat (10) begin
                seq_item=FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n=1;
                seq_item.wr_en=1;
                seq_item.rd_en=1;
                seq_item.data_in=$random();
                finish_item(seq_item);
            end
            //with every write the the out will read the write that preceded it
except for the first read because from the previos loop the mem was empty

//FIFO_5(random_sequence)
                seq_item=FIFO_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                rst_n=0;
                finish_item(seq_item);

            repeat (10000) 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_pkg
```

FIFO_rst_sequence_pkg

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

    class FIFO_rst_sequence extends  uvm_sequence #(FIFO_seq_item);
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_object_utils(FIFO_rst_sequence)

        FIFO_seq_item seq_item;

        // Constructor
        function new(string name = "FIFO_rst_sequence");
            super.new(name);
        endfunction : new

//FIFO_1(reset)
        task body();
            seq_item=FIFO_seq_item::type_id::create("seq_item");
            start_item(seq_item);
            seq_item.rst_n=0;
            seq_item.rd_en=1;
            seq_item.wr_en=1;
            seq_item.data_in=16'hFFFF;
            finish_item(seq_item);
        endtask : body

    endclass : FIFO_rst_sequence
endpackage : FIFO_rst_sequence_pkg
```

FIFO_sequencer_pkg

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

    class FIFO_sequencer extends uvm_sequencer #(FIFO_seq_item);
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_sequencer)
        // Constructor
        function new(string name = "FIFO_sequencer", uvm_component
parent=null);
            super.new(name, parent);
        endfunction : new

    endclass : FIFO_sequencer
endpackage : FIFO_sequencer_pkg
```

FIFO_monitor_pkg

```
package FIFO_monitor_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_sequence_item_pkg::*;
    class FIFO_monitor extends uvm_monitor;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_monitor)

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

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

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

        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.rd_en=FIFO_vif.rd_en;
                rsp_seq_item.wr_en=FIFO_vif.wr_en;
                rsp_seq_item.rst_n=FIFO_vif.rst_n;
                rsp_seq_item.data_in=FIFO_vif.data_in;
                rsp_seq_item.full=FIFO_vif.full;
                rsp_seq_item.empty=FIFO_vif.empty;
                rsp_seq_item.wr_ack=FIFO_vif.wr_ack;
                rsp_seq_item.almostfull=FIFO_vif.almostfull;
                rsp_seq_item.almostempty=FIFO_vif.almostempty;
                rsp_seq_item.overflow=FIFO_vif.overflow;
                rsp_seq_item.underflow=FIFO_vif.underflow;
                rsp_seq_item.data_out=FIFO_vif.data_out;
                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)
            end
```

```
        endtask : run_phase

    endclass : FIFO_monitor
endpackage : FIFO_monitor_pkg
```

FIFO_coverage_pkg

```
package FIFO_coverage_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_sequence_item_pkg::*;
    class FIFO_coverage extends uvm_component;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_coverage)

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

        covergroup CovGp ();
            wr_full: cross seq_item_cov.wr_en, seq_item_cov.full;
            wr_rd_almostfull: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.almostfull;
            wr_rd_empty: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.empty;
            wr_rd_almostempty: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.almostempty;
            wr_rd_overflow: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.overflow;
            wr_rd_underflow: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.underflow;
            wr_rd_wr_ack: cross seq_item_cov.wr_en, seq_item_cov.rd_en,
seq_item_cov.wr_ack;
         endgroup : CovGp

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

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

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

```
        endfunction : connect_phase

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

    endclass : FIFO_coverage
endpackage : FIFO_coverage_pkg
```

FIFO_driver_pkg

```
package FIFO_driver_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_sequence_item_pkg::*;
    class FIFO_driver extends uvm_driver #(FIFO_seq_item);
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_driver)

        virtual FIFO_inter FIFO_vif;
        FIFO_seq_item stim_seq_item;

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

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

    endclass : FIFO_driver
endpackage : FIFO_driver_pkg
```

FIFO_agent_pkg

```
package FIFO_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_driver_pkg::*;
    import FIFO_monitor_pkg::*;
    import FIFO_sequencer_pkg::*;
    import FIFO_config_pkg::*;
    import FIFO_sequence_item_pkg::*;
    class FIFO_agent extends uvm_agent;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_agent)

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

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

        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", "agent- unable to get the db block");
            end

            sqr=FIFO_sequencer::type_id::create("sqr",this);
            drv=FIFO_driver::type_id::create("drv",this);
            mon=FIFO_monitor::type_id::create("mon",this);
            agt_ap=new("agt_ap",this);
        endfunction : build_phase

        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 : connect_phase
```

```
    endclass : FIFO_agent

endpackage : FIFO_agent_pkg
```

FIFO_scoreboard_pkg

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

    class FIFO_scoreboard extends uvm_scoreboard;
        // Provide implementations of virtual methods such as get_type_name and
create
        `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;

        static logic [15:0] data_out_ex;

        int error_count=0;
        int correct_count=0;

        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;

        localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

        logic [max_fifo_addr-1:0] wr_count, rd_count;
        logic [max_fifo_addr:0] count;

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

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

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

```
        endfunction : connect_phase

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                ref_model(seq_item_sb);
                if (seq_item_sb.data_out != data_out_ex) begin  //compare
between the design and the golden model
                    `uvm_error("run_phase", "Comparsion failed")
                    error_count++;
                end
                else begin
                    correct_count++;
                end
            end
        endtask : run_phase

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("correct :%0d",
correct_count),UVM_MEDIUM);
            `uvm_info("report_phase", $sformatf("Error :%0d",
error_count),UVM_MEDIUM);
        endfunction : report_phase

        task ref_model(FIFO_seq_item seq_item_sb);
            if (!seq_item_sb.rst_n) begin   //the rst_n doesn't reset the
output
                wr_count=0;
                rd_count=0;
                count=0;
            end
            else if (seq_item_sb.wr_en && seq_item_sb.rd_en) begin
                if (count==0) begin
                    mem[wr_count]=seq_item_sb.data_in;
                    wr_count++;
                    count++;
                end
                else if (count==FIFO_DEPTH) begin
                    data_out_ex=mem[rd_count];
                    rd_count++;
                    count--;
                end
                else begin
```

```
                    data_out_ex=mem[rd_count];  //read first because if wr was
first and "wr_count++"= "rd_count" "data_out_ex" will take the new value
                    rd_count++;
                    mem[wr_count]=seq_item_sb.data_in;
                    wr_count++;
                end
            end
            else begin
                if (seq_item_sb.wr_en && (count!=FIFO_DEPTH)) begin
                    mem[wr_count]=seq_item_sb.data_in;
                    wr_count++;
                    count++;
                end
                else if (seq_item_sb.rd_en && (count!=0)) begin
                    data_out_ex=mem[rd_count];
                    rd_count++;
                    count--;
                end
            end
        endtask : ref_model

    endclass : FIFO_scoreboard
endpackage : FIFO_scoreboard_pkg
```

FIFO_env_pkg

```
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;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_env)

        FIFO_agent agt;
        FIFO_scoreboard sb;
        FIFO_coverage cov;

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

        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 : build_phase

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agt.agt_ap.connect(sb.sb_export);
            agt.agt_ap.connect(cov.cov_export);
        endfunction : connect_phase

    endclass : FIFO_env
endpackage : FIFO_env_pkg
```

FIFO_test_pkg

```
package FIFO_test_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import FIFO_config_pkg::*;
    import FIFO_main_sequence_pkg::*;
    import FIFO_rst_sequence_pkg::*;
    import FIFO_env_pkg::*;

    class FIFO_test extends uvm_test;
        // Provide implementations of virtual methods such as get_type_name and
create
        `uvm_component_utils(FIFO_test)

        virtual FIFO_inter FIFO_vif;
        FIFO_config FIFO_cfg;
        FIFO_env env;
        FIFO_rst_sequence reset_seq;
        FIFO_main_sequence main_seq;

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

        // Build_phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            FIFO_cfg=FIFO_config::type_id::create("FIFO_cfg");
            env=FIFO_env::type_id::create("env", this);
            reset_seq=FIFO_rst_sequence::type_id::create("reset_seq");
            main_seq=FIFO_main_sequence::type_id::create("main_seq");

            if (!uvm_config_db#(virtual FIFO_inter)::get(this, "", "FIFO_IF",
FIFO_cfg.FIFO_vif)) begin
                `uvm_fatal("build_phase", "TEST- unable to get the virtual
interface");
            end
            uvm_config_db#(FIFO_config)::set(this, "*", "CFG", FIFO_cfg);
        endfunction : build_phase

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

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

            `uvm_info("run_phase", "main begin", UVM_LOW)
            main_seq.start(env.agt.sqr);
            `uvm_info("run_phase", "main end", UVM_LOW)
            phase.drop_objection(this);
        endtask : run_phase

    endclass : FIFO_test
endpackage : FIFO_test_pkg
```

9) TOP_module

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

module FIFO_top ();
    bit clk;

    initial begin
        clk=0;
        forever
        #1 clk=~clk;
    end

    FIFO_inter inter (clk);
    FIFO DUT (
        .clk        (clk),
        .rd_en      (inter.rd_en),
        .wr_en      (inter.wr_en),
        .rst_n      (inter.rst_n),
        .data_in    (inter.data_in),
        .full       (inter.full),
        .empty      (inter.empty),
        .wr_ack     (inter.wr_ack),
        .data_out   (inter.data_out),
        .overflow   (inter.overflow),
        .underflow  (inter.underflow),
        .almostfull (inter.almostfull),
        .almostempty(inter.almostempty)
        );

    bind FIFO FIFO_SVA SVA (
        .clk        (clk),
        .rd_en      (inter.rd_en),
        .wr_en      (inter.wr_en),
        .rst_n      (inter.rst_n),
        .data_in    (inter.data_in),
        .full       (inter.full),
        .empty      (inter.empty),
        .wr_ack     (inter.wr_ack),
        .data_out   (inter.data_out),
        .overflow   (inter.overflow),
        .underflow  (inter.underflow),
        .almostfull (inter.almostfull),
```

```
        .almostempty(inter.almostempty)
        );

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

endmodule : FIFO_top
```

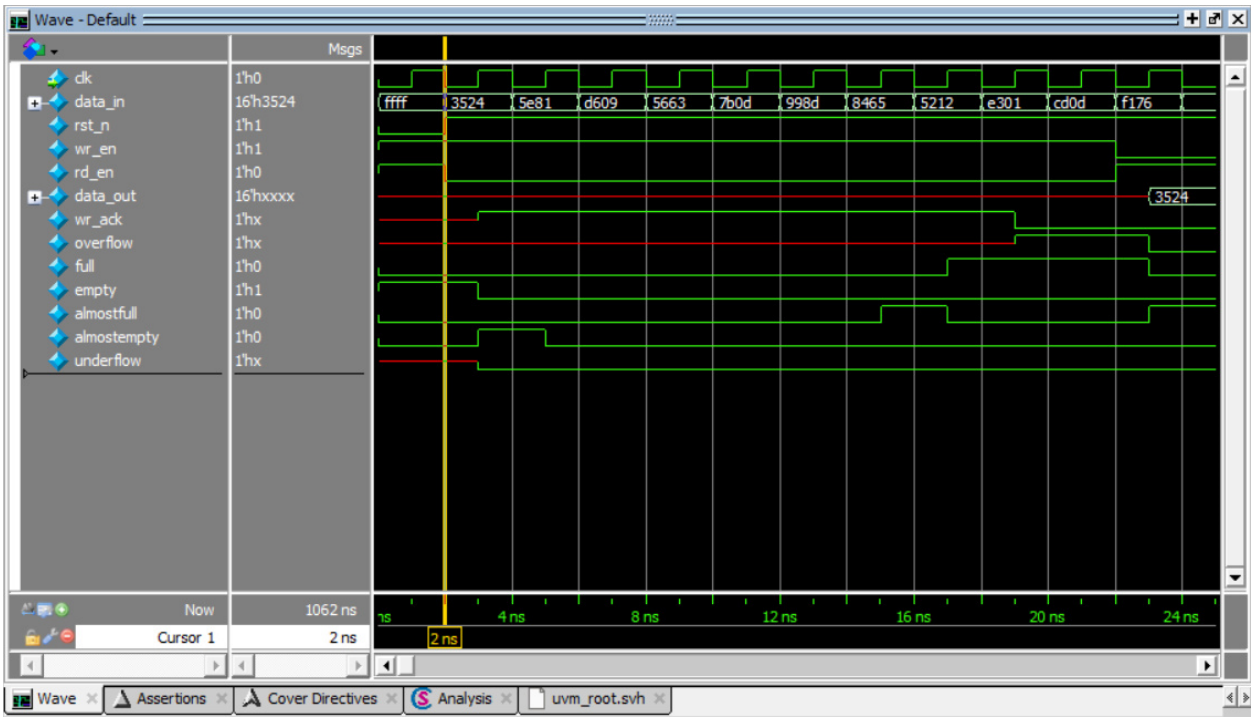
10) Do file

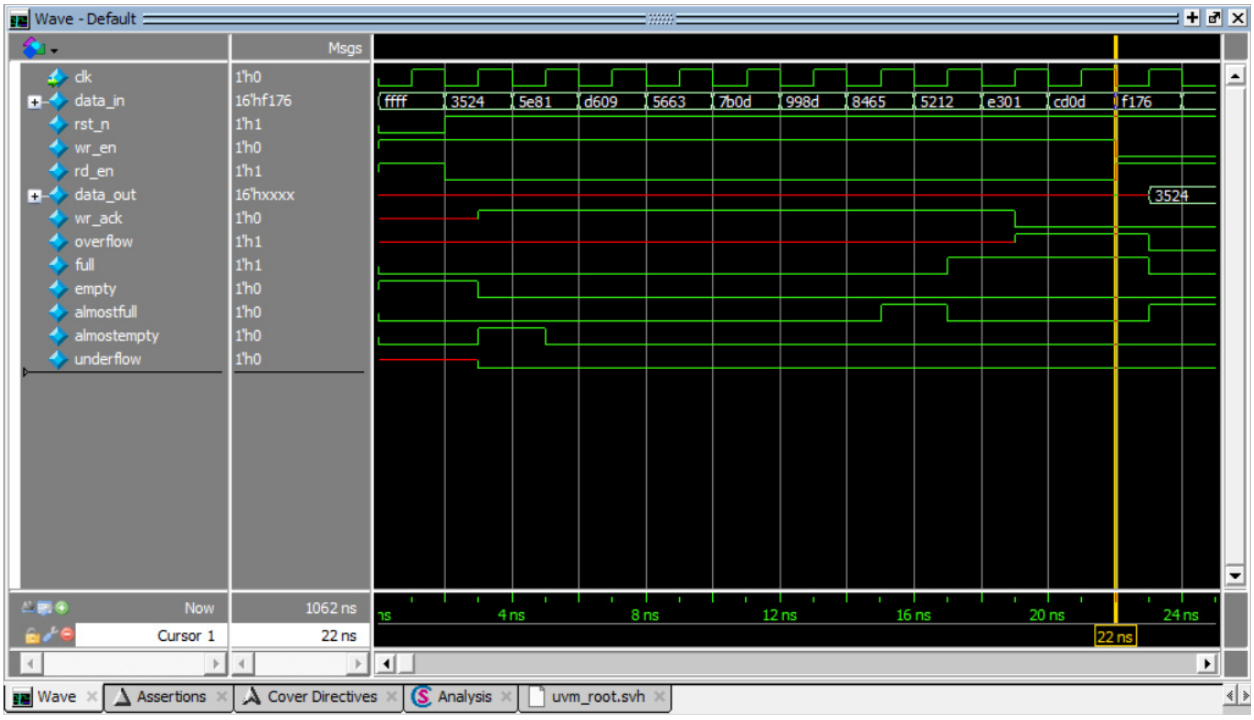
```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.FIFO_top -classdebug -uvmcontrol=all -cover
add wave -position insertpoint sim:/FIFO_top/inter/*
coverage save FIFO_top.ucdb -onexit
run -all
```

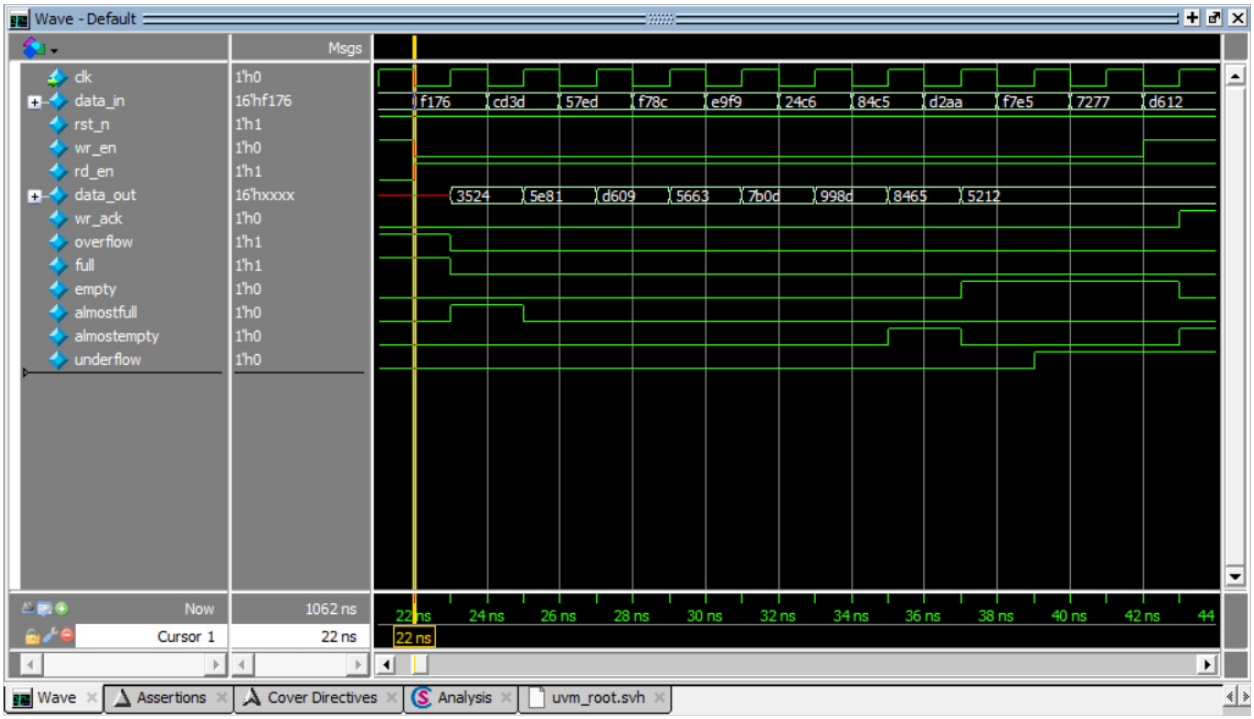
11) Src_list File

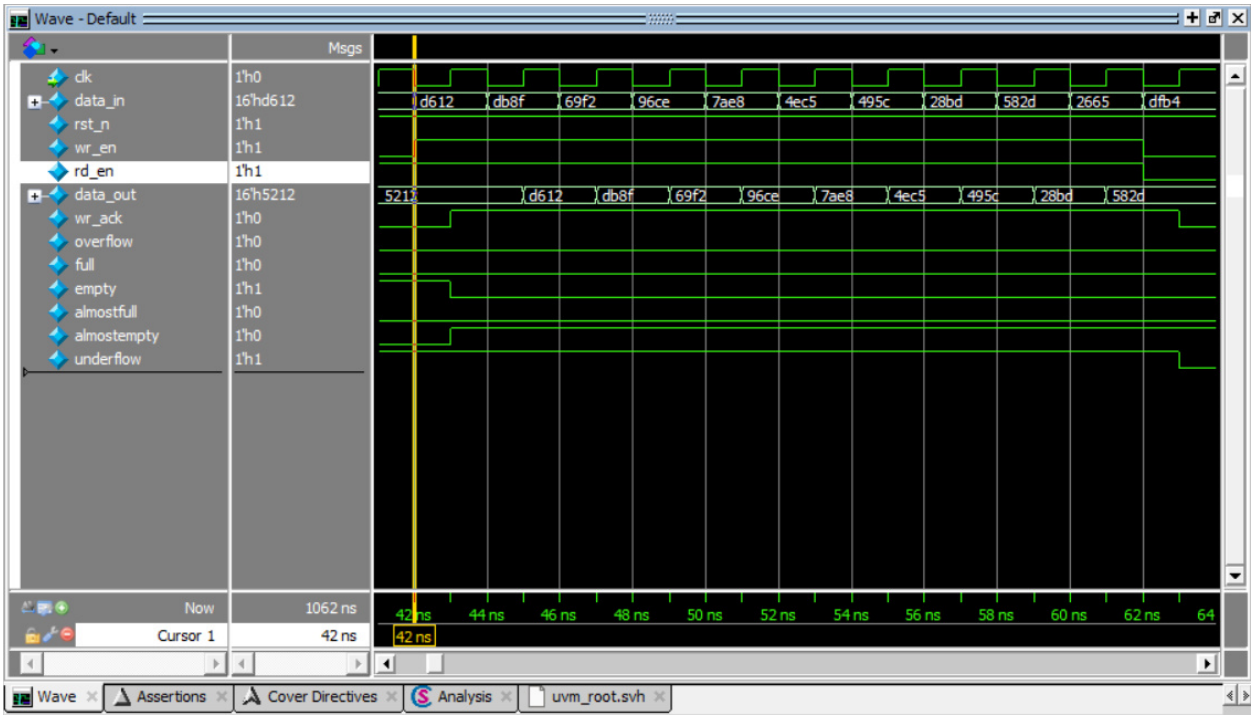
```
FIFO.v
FIFO_SVA.sv
FIFO_config_pkg.sv
FIFO_sequence_item_pkg.sv
FIFO_inter.sv
FIFO_main_sequence_pkg.sv
FIFO_rst_sequence_pkg.sv
FIFO_sequencer_pkg.sv
FIFO_monitor_pkg.sv
FIFO_coverage_pkg.sv
FIFO_driver_pkg.sv
FIFO_agent_pkg.sv
FIFO_scoreboard_pkg.sv
FIFO_env_pkg.sv
FIFO_test_pkg.sv
FIFO_top.sv
```

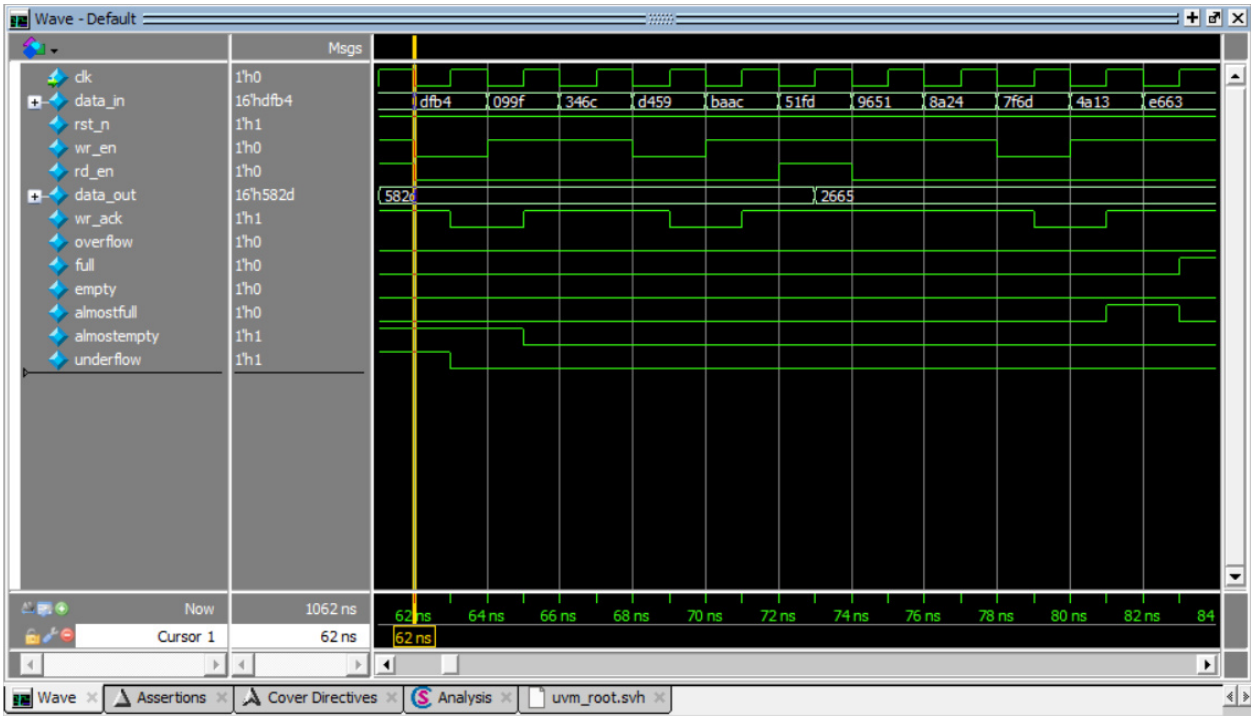
12) Waveform


Wave - Default
Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
1'h0
16'h3524
1'h1
1'h1
1'h0
16'hxxxx
1'hx
1'hx
1'h0
1'h1
1'h0
1'h0
1'hx
ffff
3524
5e81
d609
5663
7b0d
998d
8465
5212
e301
cd0d
f176
3524
Now
1062 ns
Cursor 1
2 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
Wave
Assertions
Cover Directives
Analysis
uvm_root.svh


Wave - Default
Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
1'h0
16'hf176
1'h1
1'h0
1'h1
16'hxxxx
1'h0
1'h1
1'h1
1'h0
1'h0
1'h0
1'h0
ffff
3524
5e81
d609
5663
7b0d
998d
8465
5212
e301
cd0d
f176
3524
Now
1062 ns
Cursor 1
22 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
22 ns
Wave
Assertions
Cover Directives
Analysis
uvm_root.svh


Wave - Default
Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
1'h0
16'hf176
1'h1
1'h0
1'h1
16'hxxxx
1'h0
1'h1
1'h1
1'h0
1'h0
1'h0
f176
cd3d
57ed
f78c
e9f9
24c6
84c5
d2aa
f7e5
7277
d612
3524
5e81
d609
5663
7b0d
998d
8465
5212
Now
1062 ns
Cursor 1
22 ns
22 ns
24 ns
26 ns
28 ns
30 ns
32 ns
34 ns
36 ns
38 ns
40 ns
42 ns
Wave
Assertions
Cover Directives
Analysis
uvm_root.svh


Wave - Default
Msgs
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
1'h0
16'hd612
1'h1
1'h1
1'h1
16'h5212
1'h0
1'h0
1'h0
1'h1
1'h0
1'h0
1'h1
d612
db8f
69f2
96ce
7ae8
4ec5
495c
28bd
582d
2665
dfb4
d612
db8f
69f2
96ce
7ae8
4ec5
495c
28bd
582d
Now
1062 ns
Cursor 1
42 ns
42 ns
44 ns
46 ns
48 ns
50 ns
52 ns
54 ns
56 ns
58 ns
60 ns
62 ns
Wave
Assertions
Cover Directives
Analysis
uvm_root.svh



Labels:

FIFO_1	0ms -> 2ms
FIFO_2	2ms -> 22ms
FIFO_3	22ms -> 42ms
FIFO_4	42ms -> 62ms
FIFO_5	62ms -> stop

13) Transcript

```
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test FIFO_test...
# UVM_INFO FIFO_test_pkg.sv(42) @ 0: uvm_test_top [run_phase] Reset asserted
# ****************************************************************************
# * Questa UVM Transaction Recording Turned ON.                              *
# * recording_detail has been set.                                           *
# *  To turn off, set 'recording_detail' to off:                             *
# * uvm_config_db#(int)            ::set(null, "", "recording_detail", 0); *
# * uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ****************************************************************************
# UVM_INFO FIFO_test_pkg.sv(44) @ 2: uvm_test_top [run_phase] Reset deasserted
# UVM_INFO FIFO_test_pkg.sv(46) @ 2: uvm_test_top [run_phase] main begin
# UVM_INFO FIFO_test_pkg.sv(48) @ 20062: uvm_test_top [run_phase] main end
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20062: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_scoreboard_pkg.sv(62) @ 20062: uvm_test_top.env.sb [report_phase] correct :10031
# UVM_INFO FIFO_scoreboard_pkg.sv(63) @ 20062: uvm_test_top.env.sb [report_phase] Error :0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   10
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
# [run_phase]     4
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 20062 ns  Iteration: 61  Instance: /FIFO_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430

VSIM 15>
```

14) Coverage report

Assertions coverage

```
Assertion Coverage:
    Assertions                      21        21         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/FIFO_top/DUT/SVA/assert__assert_rd_ptr_full
                     FIFO_SVA.sv(170)                    0          1
/FIFO_top/DUT/SVA/assert__assert_rd_ptr_not_full_max
                     FIFO_SVA.sv(167)                    0          1
/FIFO_top/DUT/SVA/assert__assert_rd_ptr_not_full
                     FIFO_SVA.sv(164)                    0          1
/FIFO_top/DUT/SVA/assert__assert_wr_ptr_full
                     FIFO_SVA.sv(161)                    0          1
/FIFO_top/DUT/SVA/assert__assert_wr_ptr_not_full_max
                     FIFO_SVA.sv(158)                    0          1
/FIFO_top/DUT/SVA/assert__assert_wr_ptr_not_full
                     FIFO_SVA.sv(155)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_read_max
                     FIFO_SVA.sv(152)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_read_not_max
                     FIFO_SVA.sv(149)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_write_max
                     FIFO_SVA.sv(146)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_write_not_max
                     FIFO_SVA.sv(143)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_ZERO
                     FIFO_SVA.sv(140)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_max
                     FIFO_SVA.sv(137)                    0          1
/FIFO_top/DUT/SVA/assert__assert_count_not_max
                     FIFO_SVA.sv(134)                    0          1
/FIFO_top/DUT/SVA/assert__assert_wr_ack
                     FIFO_SVA.sv(131)                    0          1
/FIFO_top/DUT/SVA/assert__assert_underflow
                     FIFO_SVA.sv(128)                    0          1
/FIFO_top/DUT/SVA/assert__assert_overflow
                     FIFO_SVA.sv(125)                    0          1
/FIFO_top/DUT/SVA/assert_full
                     FIFO_SVA.sv(55)                     0          1
/FIFO_top/DUT/SVA/assert_almostfull
```

```
                    FIFO_SVA.sv(59)                    0          1
/FIFO_top/DUT/SVA/assert_empty
                    FIFO_SVA.sv(63)                    0          1
/FIFO_top/DUT/SVA/assert_almostempty
                    FIFO_SVA.sv(67)                    0          1
/FIFO_top/DUT/SVA/assert_reset
                    FIFO_SVA.sv(71)                    0          1
```

Directive coverage

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
-------------
Name                                     Design Design   Lang
File(Line)      Hits Status
                                         Unit   UnitType

--------------------------------------------------------------------------------
-------------
/FIFO_top/DUT/SVA/cover__assert_rd_ptr_full
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(171) 209 Covered
/FIFO_top/DUT/SVA/cover__assert_rd_ptr_not_full_max
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(168) 167 Covered
/FIFO_top/DUT/SVA/cover__assert_rd_ptr_not_full
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(165)2342 Covered
/FIFO_top/DUT/SVA/cover__assert_wr_ptr_full
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(162)1641 Covered
/FIFO_top/DUT/SVA/cover__assert_wr_ptr_not_full_max
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(159) 407 Covered
/FIFO_top/DUT/SVA/cover__assert_wr_ptr_not_full
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(156)4350 Covered
/FIFO_top/DUT/SVA/cover__assert_count_read_max
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(153)  50 Covered
/FIFO_top/DUT/SVA/cover__assert_count_read_not_max
                                         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(150) 713 Covered
/FIFO_top/DUT/SVA/cover__assert_count_write_max
```

```
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(147)1139 Covered
/FIFO_top/DUT/SVA/cover__assert_count_write_not_max
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(144)3304 Covered
/FIFO_top/DUT/SVA/cover__assert_count_ZERO
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(141) 159 Covered
/FIFO_top/DUT/SVA/cover__assert_count_max
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(138) 502 Covered
/FIFO_top/DUT/SVA/cover__assert_count_not_max
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(135)1294 Covered
/FIFO_top/DUT/SVA/cover__assert_wr_ack   FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(132)4757 Covered
/FIFO_top/DUT/SVA/cover__assert_underflow
                                        FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(129) 209 Covered
/FIFO_top/DUT/SVA/cover__assert_overflow FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(126)1641 Covered
/FIFO_top/DUT/SVA/full_c                FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(56)  856 Covered
/FIFO_top/DUT/SVA/almostfull_c          FIFO_SVA Verilog  SVA  FIFO_SVA.sv(60)
1521 Covered
/FIFO_top/DUT/SVA/empty_c               FIFO_SVA Verilog  SVA  FIFO_SVA.sv(64)
1027 Covered
/FIFO_top/DUT/SVA/almostempty_c         FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(68)  806 Covered
/FIFO_top/DUT/SVA/reset_c               FIFO_SVA
Verilog  SVA  FIFO_SVA.sv(72)  468 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 21
```

Branch coverage

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        28        28         0   100.00%

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

Branch Coverage for instance /FIFO_top/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.v
-----------------------------------IF Branch------------------------------------
--
    25                                     10499     Count coming in to IF
    25             1                         970        if (!rst_n) begin

    28             1                        5003        else if (wr_en && count
< FIFO_DEPTH) begin

    33             1                        4526        else begin

Branch totals: 3 hits of 3 branches = 100.00%

-----------------------------------IF Branch------------------------------------
--
    35                                     4526     Count coming in to IF
    35             1                        1721            if (full &&
wr_en)  //was "&" correction is "&&"

    37             1                        2805            else

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
--
    43                                     8388     Count coming in to IF
    43             1                         952        if (!rst_n) begin

    46             1                        2636        else if (rd_en && count
!= 0) begin
```

```
    50              1                       4800        else
begin                     //remove underflow from assert statements because it
is sequential output and add this else statement

Branch totals: 3 hits of 3 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
--
    51                                      4800     Count coming in to IF
    51              1                        219             if (empty && rd_en)
begin

    54              1                       4581             else begin

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
--
    61                                      9257     Count coming in to IF
    61              1                        960        if (!rst_n) begin

    64              1                       8297        else begin

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
--
    66                                      8297     Count coming in to IF
    66              1                       1777             if (({wr_en, rd_en}
== 2'b11)) begin

    74              1                       6520             else begin

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
--
    67                                      1777     Count coming in to IF
    67              1                        169                 if (count==0)
begin

    70              1                        523                 else if
(count==FIFO_DEPTH) begin

                                            1085     All False Count
```

```
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch-------------------------------------
--
    75                                    6520     Count coming in to IF
    75              1                     3478                 if  ( (wr_en ==
1) && !full)

    77              1                      757                 else if (
(rd_en == 1) && !empty)

                                          2285     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch-------------------------------------
--
    83                                    5377     Count coming in to IF
    83              1                      856     assign full = (count ==
FIFO_DEPTH)? 1 : 0;

    83              2                     4521     assign full = (count ==
FIFO_DEPTH)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch-------------------------------------
--
    84                                    5377     Count coming in to IF
    84              1                      541     assign empty = (count ==
0)? 1 : 0;

    84              2                     4836     assign empty = (count ==
0)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch-------------------------------------
--
    85                                    5377     Count coming in to IF
    85              1                     1102     assign almostfull = (count
== FIFO_DEPTH-1)? 1 : 0;    //was "FIFO_DEPTH-2" correction "FIFO_DEPTH-1" only

    85              2                     4275     assign almostfull = (count
== FIFO_DEPTH-1)? 1 : 0;    //was "FIFO_DEPTH-2" correction "FIFO_DEPTH-1" only
```

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

-----------------------------------IF Branch-----------------------------------
--
    86                                      5377     Count coming in to IF
    86              1                        624     assign almostempty =
(count == 1)? 1 : 0;

    86              2                       4753     assign almostempty =
(count == 1)? 1 : 0;

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

Condition coverage

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      20        20         0   100.00%

================================Condition
Details================================

Condition Coverage for instance /FIFO_top/DUT --

  File FIFO.v
----------------Focused Condition View-------------------
Line       28 Item    1  (wr_en && (count < 8))
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       35 Item    1  (full && wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

```
----------------Focused Condition View-------------------
Line       46 Item    1  (rd_en && (count != 0))
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       51 Item    1  (empty && rd_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       66 Item    1  (rd_en & wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  rd_en_0              wr_en
  Row   2:          1  rd_en_1              wr_en
```

```
  Row   3:          1  wr_en_0              rd_en
  Row   4:          1  wr_en_1              rd_en

----------------Focused Condition View-------------------
Line       67 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       70 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       75 Item    1  (wr_en && ~full)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

```
----------------Focused Condition View-------------------
Line       77 Item    1  (rd_en && ~empty)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       83 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       84 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       85 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       86 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        -
```

Statement coverage

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      24        24         0   100.00%

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

Statement Coverage for instance /FIFO_top/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.v
    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;

    23

    24              1                   10499   always @(posedge clk or
negedge rst_n) begin

    25                                             if (!rst_n) begin

    26              1                     970          wr_ptr <= 0;

    27                                             end

    28                                             else if (wr_en && count
< FIFO_DEPTH) begin

    29              1                    5003          mem[wr_ptr] <=
data_in;

    30              1                    5003          wr_ack <= 1;

    31              1                    5003          wr_ptr <= wr_ptr +
1;

    32                                             end

    33                                             else begin

    34              1                    4526          wr_ack <= 0;

    35                                             if (full &&
wr_en)  //was "&" correction is "&&"

    36              1                    1721              overflow <= 1;

    37                                             else
```

```
    38              1                       2805                overflow <= 0;
    39                                                      end
    40                                                  end
    41
    42              1                       8388        always @(posedge clk or
negedge rst_n) begin
    43                                                      if (!rst_n) begin
    44              1                        952            rd_ptr <= 0;
    45                                                      end
    46                                                      else if (rd_en && count
!= 0) begin
    47              1                       2636            data_out <=
mem[rd_ptr];
    48              1                       2636            rd_ptr <= rd_ptr +
1;
    49                                                      end
    50                                                      else
begin                       //remove underflow from assert statements because it
is sequential output and add this else statement
    51                                                          if (empty && rd_en)
begin
    52              1                        219                underflow <= 1;
    53                                                      end
    54                                                      else begin
    55              1                       4581                underflow <= 0;
    56                                                      end
```

```
    57                                              end

    58                                          end

    59

    60              1                       9257    always @(posedge clk or
negedge rst_n) begin

    61                                              if (!rst_n) begin

    62              1                        960                count <= 0;

    63                                              end

    64                                              else begin

    65                                              //added this statement
to cover all cases  ex: if wr_en and rd_en both equal to 1 and the fifo is
empty the code will do neither this "(({wr_en, rd_en} == 2'b10) && !full)" nor
this "(({wr_en, rd_en} == 2'b01) && !empty)" however we can write in fifo and
increase the counter normally

    66                                                  if (({wr_en, rd_en}
== 2'b11)) begin

    67                                                      if (count==0)
begin

    68              1                        169                    count<=coun
t+1;

    69                                                      end

    70                                                      else if
(count==FIFO_DEPTH) begin

    71              1                        523                    count<=coun
t-1;

    72                                                      end

    73                                                  end
```

```
    74                                                  else begin

    75                                                      if  ( (wr_en ==
1) && !full)

    76              1                       3478                    count <=
count + 1;

    77                                                      else if (
(rd_en == 1) && !empty)

    78              1                        757                    count <=
count - 1;

    79                                                  end

    80                                              end

    81                                          end

    82

    83              1                       5378     assign full = (count ==
FIFO_DEPTH)? 1 : 0;

    84              1                       5378     assign empty = (count ==
0)? 1 : 0;

    85              1                       5378     assign almostfull = (count
== FIFO_DEPTH-1)? 1 : 0;    //was "FIFO_DEPTH-2" correction "FIFO_DEPTH-1" only

    86              1                       5378     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)
```

Function coverage

```
Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        27        na        na        na
            Covergroup Bins        92        92         0   100.00%
--------------------------------------------------------------------------------
--------------------------
Covergroup                                          Metric       Goal
Bins    Status


--------------------------------------------------------------------------------
--------------------------
 TYPE
/FIFO_coverage_pkg/FIFO_coverage/CovGp          100.00%        100          -
 Covered
    covered/total
bins:                                  92         92          -

    missing/total
bins:                                   0         92          -

    %
Hit:                                          100.00%        100          -

    Coverpoint
#seq_item_cov.wr_en__0#                 100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                        100.00%        100          -

        bin
auto[0]                                     2947          1          -    Cove
red
        bin
auto[1]                                     7084          1          -    Cove
red
```

```
    Coverpoint
#seq_item_cov.rd_en__1#                100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                 7030          1          -    Cove
red
        bin
auto[1]                                 3001          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_ack__2#               100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                 4767          1          -    Cove
red
        bin
auto[1]                                 5263          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_en__3#                100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -
```

```
        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  2947          1          -    Cove
red
        bin
auto[1]                                  7084          1          -    Cove
red
    Coverpoint
#seq_item_cov.rd_en__4#                 100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  7030          1          -    Cove
red
        bin
auto[1]                                  3001          1          -    Cove
red
    Coverpoint
#seq_item_cov.underflow__5#             100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  9731          1          -    Cove
red
        bin
auto[1]                                   299          1          -    Cove
red
```

```
    Coverpoint
#seq_item_cov.wr_en__6#                100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                2947          1          -    Cove
red
        bin
auto[1]                                7084          1          -    Cove
red
    Coverpoint
#seq_item_cov.rd_en__7#                100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                7030          1          -    Cove
red
        bin
auto[1]                                3001          1          -    Cove
red
    Coverpoint
#seq_item_cov.overflow__8#             100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -
```

```
        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  7492          1          -    Cove
red
        bin
auto[1]                                  2530          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_en__9#                 100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  2947          1          -    Cove
red
        bin
auto[1]                                  7084          1          -    Cove
red
    Coverpoint
#seq_item_cov.rd_en__10#                100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  7030          1          -    Cove
red
        bin
auto[1]                                  3001          1          -    Cove
red
```

```
    Coverpoint
#seq_item_cov.almostempty__11#         100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                9027          1          -    Cove
red
        bin
auto[1]                                1004          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_en__12#               100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                 100.00%        100          -

        bin
auto[0]                                2947          1          -    Cove
red
        bin
auto[1]                                7084          1          -    Cove
red
    Coverpoint
#seq_item_cov.rd_en__13#               100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -
```

```
        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  7030          1          -    Cove
red
        bin
auto[1]                                  3001          1          -    Cove
red
    Coverpoint
#seq_item_cov.empty__14#                100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  9222          1          -    Cove
red
        bin
auto[1]                                   809          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_en__15#                100.00%        100          -    Covered

        covered/total
bins:                                   2          2          -

        missing/total
bins:                                   0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  2947          1          -    Cove
red
        bin
auto[1]                                  7084          1          -    Cove
red
```

```
    Coverpoint
#seq_item_cov.rd_en__16#               100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                        100.00%        100          -

        bin
auto[0]                                     7030          1          -    Cove
red
        bin
auto[1]                                     3001          1          -    Cove
red
    Coverpoint
#seq_item_cov.almostfull__17#          100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -

        %
Hit:                                        100.00%        100          -

        bin
auto[0]                                     8106          1          -    Cove
red
        bin
auto[1]                                     1925          1          -    Cove
red
    Coverpoint
#seq_item_cov.wr_en__18#               100.00%        100          -    Covered

        covered/total
bins:                                  2          2          -

        missing/total
bins:                                  0          2          -
```

```
        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  2947          1          -    Cove
red
        bin
auto[1]                                  7084          1          -    Cove
red
    Coverpoint
#seq_item_cov.full__19#                100.00%        100          -    Covered

        covered/total
bins:                                 2          2          -

        missing/total
bins:                                 0          2          -

        %
Hit:                                    100.00%        100          -

        bin
auto[0]                                  7490          1          -    Cove
red
        bin
auto[1]                                  2541          1          -    Cove
red
    Cross
wr_full                                 100.00%        100          -    Co
vered
        covered/total
bins:                                 4          4          -

        missing/total
bins:                                 0          4          -

        %
Hit:                                    100.00%        100          -

        Auto, Default and User Defined Bins:
            bin
<auto[1],auto[1]>                       2054          1          -    Covered
```

```
            bin
<auto[0],auto[1]>                        487          1          -    Covered

            bin
<auto[1],auto[0]>                       5030          1          -    Covered

            bin
<auto[0],auto[0]>                       2460          1          -    Covered

    Cross
wr_rd_almostfull                          100.00%       100          -    Co
vered
        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]>                1214          1          -    Covered

            bin
<auto[0],auto[1],auto[1]>                 634          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>                 368          1          -    Covered

            bin
<auto[0],auto[0],auto[1]>                 404          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>                1214          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                 634          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>                4560          1          -    Covered
```

```
            bin
<auto[0],auto[0],auto[0]>                1698          1          -    Covered

    Cross
wr_rd_empty                                100.00%        100          -    Co
vered
        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]>                 108          1          -    Covered

            bin
<auto[0],auto[1],auto[1]>                 179          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>                 252          1          -    Covered

            bin
<auto[0],auto[0],auto[1]>                 270          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>                2048          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                 666          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>                4676          1          -    Covered

            bin
<auto[0],auto[0],auto[0]>                1832          1          -    Covered

    Cross
wr_rd_almostempty                          100.00%        100          -    Co
vered
```

```
        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]>                  351          1          -    Covered

            bin
<auto[0],auto[1],auto[1]>                   83          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>                  372          1          -    Covered

            bin
<auto[0],auto[0],auto[1]>                  198          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>                 1805          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                  762          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>                 4556          1          -    Covered

            bin
<auto[0],auto[0],auto[0]>                 1904          1          -    Covered

    Cross
wr_rd_overflow                              100.00%        100          -    Co
vered
        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]>                762          1          -    Covered

            bin
<auto[0],auto[1],auto[1]>                  8          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>               1733          1          -    Covered

            bin
<auto[0],auto[0],auto[1]>                 27          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>               1393          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                837          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>               3187          1          -    Covered

            bin
<auto[0],auto[0],auto[0]>               2075          1          -    Covered

    Cross
wr_rd_underflow                         100.00%        100          -    Co
vered
        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]>                209          1          -    Covered
```

```
            bin
<auto[0],auto[1],auto[1]>                  81          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>                   7          1          -    Covered

            bin
<auto[0],auto[0],auto[1]>                   2          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>                1946          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                 764          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>                4921          1          -    Covered

            bin
<auto[0],auto[0],auto[0]>                2100          1          -    Covered

    Cross
wr_rd_wr_ack                              100.00%        100          -    Co
vered
        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]>                1576          1          -    Covered

            bin
<auto[0],auto[1],auto[1]>                  16          1          -    Covered

            bin
<auto[1],auto[0],auto[1]>                3618          1          -    Covered
```

```
            bin
<auto[0],auto[0],auto[1]>                 53          1          -    Covered

            bin
<auto[1],auto[1],auto[0]>                579          1          -    Covered

            bin
<auto[0],auto[1],auto[0]>                829          1          -    Covered

            bin
<auto[1],auto[0],auto[0]>               1310          1          -    Covered

            bin
<auto[0],auto[0],auto[0]>               2049          1          -    Covered
```