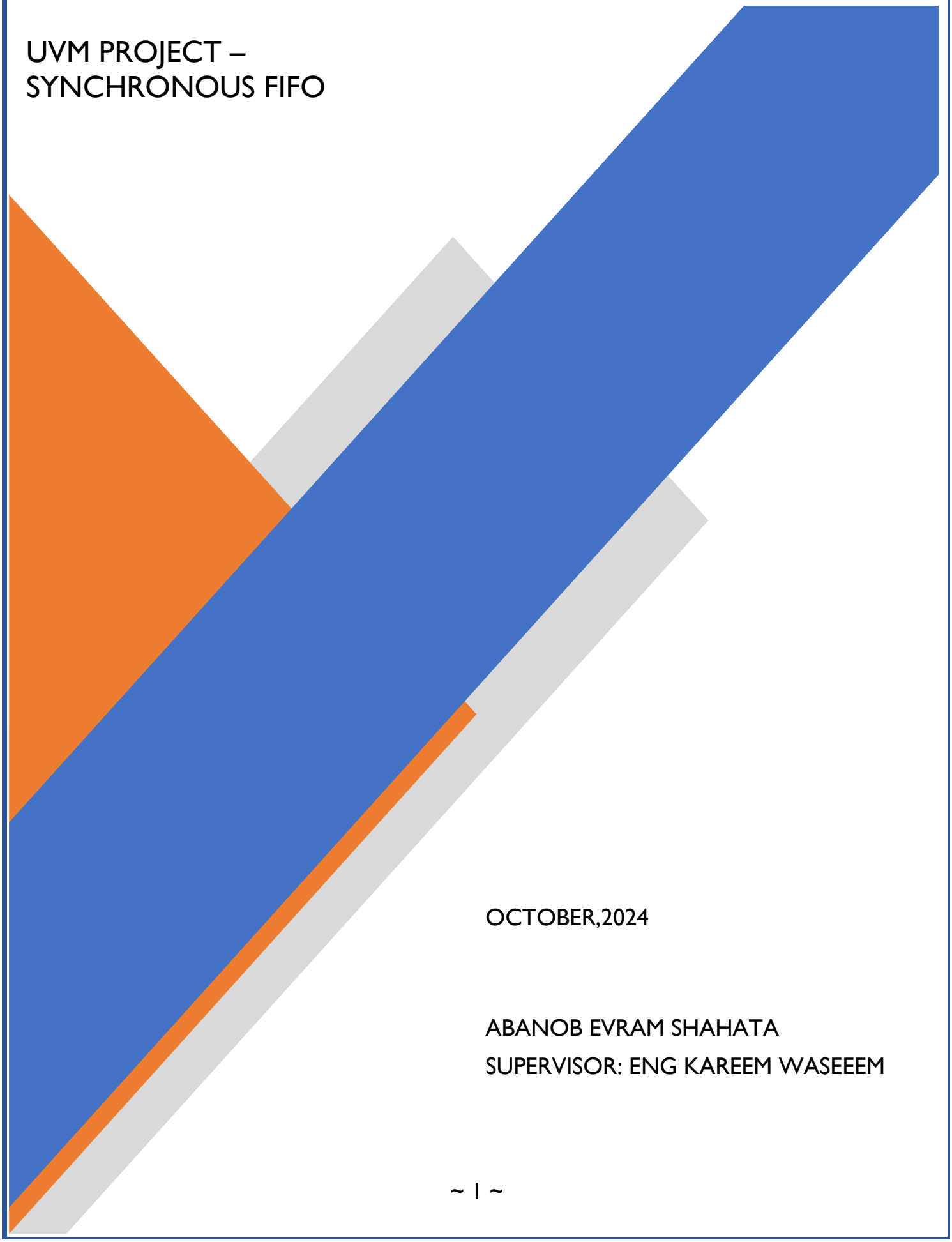
UVM PROJECT – SYNCHRONOUS FIFO



OCTOBER,2024

ABANOB EVRAM SHAHATA

SUPERVISOR: ENG KAREEM WASEEEM

TABLE OF CONTENTS

Verification plane

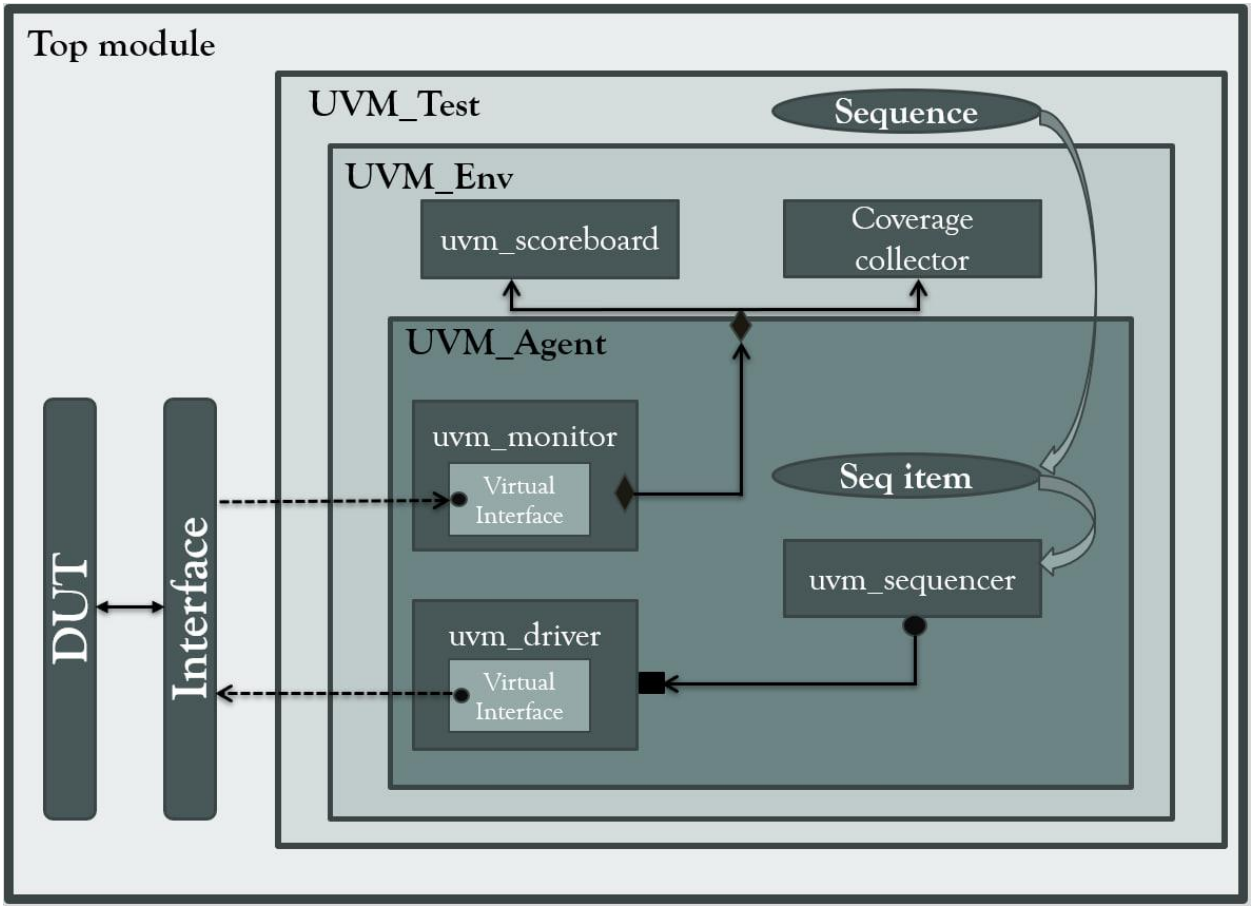
Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	Asserted reset at the start of the simulation	Directed at the start of the simulation	Cover the reset in coverpoint named rst_n_p	-
FIFO_2	When the reset is asserted, The sequential signals (wr_ack, overflow, underflow,rd_ptr,wr_ptr) should be zero	Directed at the start of the simulation	Cover this signals (wr_ack,overflow,underflow) in the covergroup	Immdiate assertion to check all seq signals when reset assert. A checker in the scoreboard to make sure the output is correct
FIFO_3	Write 10 iterations with randomized data. The 10 iterations will test writing to the FIFO when it is full.	Randomize under constraint (Write_only) that wr_en always high and rd_en low . also constriant (data_in) that be MAX 10% , low 10% and all values 80%	Cover all flags at the covergroup	Conncurrent assertions to check all flags and internal signals. A checker in the scoreboard to make sure the output is correct
FIFO_4	Read 10 iterations The 10 iterations will test reading from the FIFO when it is empty.	Randomize under constraint (Read_only) that rd_en always high and wr_en low .	Cover all flags at the covergroup	Conncurrent assertions to check all flags and internal signals. A checker in the scoreboard to make sure the output is correct
FIFO_5	Randomize all inputs 1000 iterations	Randomize under constraint (Write_Read) that wr_en be on 70% and rd_enbe on 30% of the simulation. Constriant (data_in) that be MAX10% , low 10% and all values 80%. Also constraint (Reset_prob) on reset to be off most of the time	Cover all flags , wr_en, rd_en and rst_n in the covergroup. Cross between the wr_en , rd_en and all flags	Conncurrent assertions to check all flags and internal signals. A checker in the scoreboard to make sure the output is correct
FIFO_6	Always write and read signal is high	Randomize under constraint (Write_Read_always) that wr_en and rd_en always high . also constriant (data_in) that be MAX 10% , low 10% and all values 80%	Cover all flags , wr_en, rd_en and rst_n in the covergroup. Cross between the wr_en , rd_en and all flags	Conncurrent assertions to check all flags and internal signals. A checker in the scoreboard to make sure the output is correct

Table of assertions

Num	Feature	Assertion
1	Whenever the wr_en signal is high and count equals FIFO_DEPTH, the overflow signal is always high	@(posedge clk) disable iff (!rst_n) (count==FIFO_DEPTH&&wr_en) |=> overflow==1;
2	Whenever the w_en is low or the count not equal FIFO_DEPTH, the overflow must be low	@(posedge clk) disable iff (!rst_n) !(count==FIFO_DEPTH&&wr_en) |=> overflow==0;
3	When the rd_en signal is active and count is 0, the underflow signal is always set to high	@(posedge clk) disable iff (!rst_n) (count==0&&rd_en) |=> underflow==1;
4	When the rd_en signal is inactive or count is 0, the underflow signal is always set to low	@(posedge clk) disable iff (!rst_n) !(count==0&&rd_en) |=> underflow==0;
5	Whenever the wr_en signal is high and count is not equal to FIFO_DEPTH, the wr_ack signal is always high	@(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&wr_en) |=> wr_ack==1;
6	Whenever the wr_en signal is low or count is equal equal to FIFO_DEPTH, the wr_ack signal is always low	@(posedge clk) disable iff (!rst_n) !(count!=FIFO_DEPTH&&wr_en) |=> wr_ack==0;
7	If count equals FIFO_DEPTH, the full signal is always asserted high	@(posedge clk) count==FIFO_DEPTH |-> full==1;
8	If count is less than FIFO_DEPTH, the full signal is always low	@(posedge clk) count!=FIFO_DEPTH |-> full==0;
9	Whenever count equals FIFO_DEPTH-1, the almostfull signal is always high	@(posedge clk) count==FIFO_DEPTH-1 |-> almostfull==1;
10	Whenever count is not equal to FIFO_DEPTH-1, the almostfull signal is always low	@(posedge clk) count!=FIFO_DEPTH-1 |-> almostfull==0;
11	When count is 0, the empty signal is always high	@(posedge clk) count==0 |-> empty==1;
12	If count is greater than 0, the empty signal remains low	@(posedge clk) count!=0 |-> empty==0;
13	If count equals 1, the almostempty signal will always be high	@(posedge clk) count==1 |-> almostempty==1;

14	If count is not 1, the almostempty signal will always be low	@(posedge clk) count!=1 |-> almostempty==0;
15	If count is not at FIFO_DEPTH and the wr_en signal is active, the wr_ptr will always be the value of the previous wr_ptr incremented by 1	@(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&wr_en) |=> wr_ptr==$past(wr_ptr)+1'b1;
16	If count is greater than 0 and the rd_en signal is active, the rd_ptr will always be the previous rd_ptr incremented by 1	@(posedge clk) disable iff (!rst_n) (count!=0&&rd_en) |=> rd_ptr==$past(rd_ptr)+1'b1;
17	If count is not equal to FIFO_DEPTH, wr_en is active, and rd_en is inactive, the count increases by 1 compared to the previous cycle	@(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&wr_en&&!rd_en) |=> count==$past(count)+1'b1;
18	If count is not equal to 0, wr_en is inactive, and rd_en is active, the count decreases by 1 compared to the previous cycle	@(posedge clk) disable iff (!rst_n) (count!=0&&!wr_en&&rd_en) |=> count==$past(count)-1'b1;
19	If count is neither equal to FIFO_DEPTH nor 0, and both wr_en and rd_en are active, the count remains the same as the previous cycle	@(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&count!=0&&wr_en&&rd_en) |=> count==$past(count);
20	If count equals FIFO_DEPTH and both wr_en and rd_en are active, the count decreases by 1 compared to the previous cycle	@(posedge clk) disable iff (!rst_n) (count==FIFO_DEPTH&&wr_en&&rd_en) |=> count==$past(count)-1'b1;
21	If count equals 0 and both wr_en and rd_en are active, the count increases by 1 compared to the previous cycle	@(posedge clk) disable iff (!rst_n) (count==0&&wr_en&&rd_en) |=> count==$past(count)+1'b1;

UVM Structure



UVM_FLOW

1 Create shared_pkg:

- This package contains shared variables and parameters that will be use in files.

2 Create an Interface:

- Create an interface to connect the design with testbench components.

3 Create an Assertion File:

- Write the assertion file that will be used to verify the design behavior.

4 Create top File:

- Create the clock generated
- Pass data from the interface to both the DUT and the reference (golden) model.
- Bind the assertion file to the design.
- Set the virtual interface in the configuration database.

- Run a uvm_test_top.

5 Create a Configuration Class:

- This class will be used to get the virtual interface.

6 Create a sequence_item Class:

- Include the variables and random input signals.
- Create two methods: Print_data_out and Print_flags for displaying the data_out only and all signals.
- Add some constraint block.

7 Create 5 FIFO Sequences:

These sequences are used to verify the design, including:

- reset_sequence
- Write_only
- Read_only
- Write_Read
- Write_Read_Always

8 Create a Sequencer:

- This component manages sequences and drives the data.

9 Create driver File:

- Set up a virtual interface between the driver and the real interface.
- Make the driver a producer that retrieves data.
- Assign data from the sequence Item to the interface inputs.

10 Create monitor File:

- Set up a virtual interface between the monitor and the real interface.
- Assign data from the virtual interface to the monitor object using the sequence Item inputs and outputs.
- Create an analysis_port to send data to the agent.

11 Create agent File:

- Obtain the configuration object from the database and pass it to the monitor and driver.
- Establish a connection between the monitor and the agent to send data to the scoreboard and cover collector.
- Connect the sequencer and driver.

12 Create scoreboard File:

- Receive inputs from the monitor and reference model.
- Perform comparisons and log discrepancies or mismatches.
- Signal any test failures if the actual results deviate from the expected ones.

- Display report at the end of the simulation

13 Create Coverage collector File:

- Checking that all input combinations and branches of the design have been tested.
- Monitoring various signals and transitions to ensure each aspect of the design is covered.
- Providing a measure of test completeness, ensuring the testbench has thoroughly exercised the design.

14 Create env File:

- Instantiate components like the agent, scoreboard, and cover collector.
- Connect the agent with the scoreboard and cover collector.

15 Create test File:

- Get the virtual interface and pass it to the environment.
- Create objects for the sequences.
- Use the built-in function raise_objection to signal that the test has started.
- Run the sequence objects during the run phase using a specific sequence.
- Use the built-in function drop_objection to indicate that the test has finished.

Shared pkg

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Sharedpkg
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package shared_pkg;
    parameter FIFO_WIDTH    = 16;
    parameter FIFO_DEPTH    = 8;
    parameter MAX_FIFO_ADDR = $clog2(FIFO_DEPTH);
    //For constraints
    parameter RD_EN_ON_DIST=30;
    parameter WR_EN_ON_DIST=70;
    parameter RESET_ON_DIST=5;

    parameter ACTIVE=1;
    parameter INACTIVE=0;

    parameter ACTIVE_RESET=0;
    parameter INACRIVE_RESET=1;

    parameter MAX=(2**FIFO_WIDTH)-1;
    parameter ZERO=0;
endpackage : shared_pkg
```

Design

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
// Description: FIFO Design
////////////////////////////////////////////////////////////////////////////////
import shared_pkg::*;
module FIFO(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);

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;


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

logic [MAX_FIFO_ADDR-1:0] wr_ptr, rd_ptr;
logic [MAX_FIFO_ADDR:0] count;

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
        wr_ack<=0;//reset the wr_ack when rst_n is active
        overflow<=0;//reset the overflow when rst_n is active
    end
    else begin//Edit this condition and handle the overflow cases
        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;
        end
        if(wr_en&&count==FIFO_DEPTH)
            overflow<=1;
        else
            overflow<=0;
    end
end

always @(posedge clk or negedge rst_n) begin//Add the underflow flag because it is sequential
    if (!rst_n) begin
        rd_ptr <= 0;
        underflow<=0;//reset the underflow when rst_n is active
    end
    else begin//Edit this condition and handle the underflow cases
```

```
    else begin//Edit this condition and handle the underflow cases
        if (rd_en && count != 0) begin
            data_out <= mem[rd_ptr];
            rd_ptr <= rd_ptr + 1;
        end
        if(rd_en&&count==0)
            underflow<=1;
        else
            underflow<=0;
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin//Add two cases for the count
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
        else if ( ({wr_en, rd_en} == 2'b11) && full)//1
            count <= count -1;
        else if ( ({wr_en, rd_en} == 2'b11) && empty)//2
            count <= count + 1;
    end
end

assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //fix the condition of this flag
assign almostempty = (count == 1)? 1 : 0;

endmodule
```

Interface

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Interface_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
import shared_pkg::*;
interface fifo_if(input clk);
    //input signals
    logic [FIFO_WIDTH-1:0] data_in;
    logic rst_n, wr_en, rd_en;
    //output signals
    logic [FIFO_WIDTH-1:0] data_out;
    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;
endinterface : fifo_if
```

Assertion

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Assertions_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
import shared_pkg::*;
module fifo_sva(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out,  wr_ptr, rd_ptr, count);

input [FIFO_WIDTH-1:0] data_in;
input clk, rst_n, wr_en, rd_en;
input [FIFO_WIDTH-1:0] data_out;
input wr_ack, overflow, underflow;
input full, empty, almostfull, almostempty;
input [MAX_FIFO_ADDR-1:0] wr_ptr, rd_ptr;
input [MAX_FIFO_ADDR:0] count;


/*****************************************************************************************************/
//property for the sequential flags to check when it active
property overflow_p;
    @(posedge clk) disable iff (!rst_n) (count==FIFO_DEPTH&&wr_en) |=> overflow==1;
endproperty
property underflow_p;
    @(posedge clk) disable iff (!rst_n) (count==0&&rd_en) |=> underflow==1;
endproperty
property wr_ack_p;
    @(posedge clk) disable iff (!rst_n)  (count!=FIFO_DEPTH&&wr_en) |=> wr_ack==1;
endproperty
/*****************************************************************************************************/
//property for the sequential flags to check when it Inactive
property overflow_inac_p;
    @(posedge clk) disable iff (!rst_n) !(count==FIFO_DEPTH&&wr_en) |=> overflow==0;
endproperty
property underflow_inac_p;
    @(posedge clk) disable iff (!rst_n) !(count==0&&rd_en) |=> underflow==0;
endproperty
property wr_ack_inac_p;
    @(posedge clk) disable iff (!rst_n)  !(count!=FIFO_DEPTH&&wr_en) |=> wr_ack==0;
endproperty
/*****************************************************************************************************/
//property for the combinational flags to check when it active
property full_p;
    @(posedge clk)  count==FIFO_DEPTH |-> full==1;
endproperty
property almostfull_p;
    @(posedge clk)  count==FIFO_DEPTH-1 |-> almostfull==1;
endproperty
property empty_p;
    @(posedge clk)  count==0 |-> empty==1;
endproperty
property almostempty_p;
    @(posedge clk)  count==1 |-> almostempty==1;
endproperty
/*****************************************************************************************************/
```

```
//property for the combinational flags to check when it Inactive
property full_inac_p;
    @(posedge clk)  count!=FIFO_DEPTH |-> full==0;
endproperty
property almostfull_inac_p;
    @(posedge clk)  count!=FIFO_DEPTH-1 |-> almostfull==0;
endproperty
property empty_inac_p;
    @(posedge clk)  count!=0 |-> empty==0;
endproperty
property almostempty_inac_p;
    @(posedge clk)  count!=1 |-> almostempty==0;
endproperty
/*****************************************************************************************************/
//property for the pointers
property wr_ptr_p;
    @(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&wr_en) |=> wr_ptr==$past(wr_ptr)+1'b1;
endproperty
property rd_ptr_p;
    @(posedge clk) disable iff (!rst_n) (count!=0&&rd_en) |=> rd_ptr==$past(rd_ptr)+1'b1;
endproperty
/*****************************************************************************************************/
//property for the counter signal
property count_wr_p;
    @(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&wr_en&&!rd_en) |=> count==$past(count)+1'b1;
endproperty
property count_rd_p;
    @(posedge clk) disable iff (!rst_n) (count!=0&&!wr_en&&rd_en) |=> count==$past(count)-1'b1;
endproperty
property count_wr_rd_p;//this property for check if we read and write at the same clk cycle
    @(posedge clk) disable iff (!rst_n) (count!=FIFO_DEPTH&&count!=0&&wr_en&&rd_en) |=> count==$past(count);
endproperty
property countfull_wr_rd_p;//this property for check if we read and write is high but the FIFO is full
    @(posedge clk) disable iff (!rst_n) (count==FIFO_DEPTH&&wr_en&&rd_en) |=> count==$past(count)-1'b1;
endproperty
property countempty_wr_rd_p;//this property for check if we read and write is high but the FIFO is empty
    @(posedge clk) disable iff (!rst_n) (count==0&&wr_en&&rd_en) |=> count==$past(count)+1'b1;
endproperty
/*****************************************************************************************************/
```

```
93  /*****************************************************************************************************/
94  //assert for all properties
95  overflow_assertion          :assert property(overflow_p);
96  underflow_assertion         :assert property(underflow_p);
97  wr_ack_assertion            :assert property(wr_ack_p);
98
99  overflow_inac_assertion     :assert property(overflow_inac_p);
100 underflow_inac_assertion    :assert property(underflow_inac_p);
101 wr_ack_inac_assertion       :assert property(wr_ack_inac_p);
102
103 full_assertion              :assert property(full_p);
104 almostfull_assertion        :assert property(almostfull_p);
105 empty_assertion             :assert property(empty_p);
106 almostempty_assertion       :assert property(almostempty_p);
107
108 full_inac_assertion         :assert property(full_inac_p);
109 almostfull_inac_assertion   :assert property(almostfull_inac_p);
110 empty_inac_assertion        :assert property(empty_inac_p);
111 almostempty_inac_assertion  :assert property(almostempty_inac_p);
112
113 wr_ptr_assertion            :assert property(wr_ptr_p);
114 rd_ptr_assertion            :assert property(rd_ptr_p);
115
116 count_wr_assertion          :assert property(count_wr_p);
117 count_rd_assertion          :assert property(count_rd_p);
118 count_wr_rd_assertion       :assert property(count_wr_rd_p);
119 countfull_wr_rd_assertion   :assert property(countfull_wr_rd_p);
120 countempty_wr_rd_assertion  :assert property(countempty_wr_rd_p);
121 /*****************************************************************************************************/
122 //Cover for all properties
123 overflow_coverage           :cover property(overflow_p);
124 underflow_coverage          :cover property(underflow_p);
125 wr_ack_coverage             :cover property(wr_ack_p);
126
127 overflow_inac_coverage      :cover property(overflow_inac_p);
128 underflow_inac_coverage     :cover property(underflow_inac_p);
129 wr_ack_inac_coverage        :cover property(wr_ack_inac_p);
130
131 full_coverage               :cover property(full_p);
132 almostfull_coverage         :cover property(almostfull_p);
133 empty_coverage              :cover property(empty_p);
134 almostempty_coverage        :cover property(almostempty_p);
135
136 full_inac_coverage          :cover property(full_inac_p);
137 almostfull_inac_coverage    :cover property(almostfull_inac_p);
138 empty_inac_coverage         :cover property(empty_inac_p);
139 almostempty_inac_coverage   :cover property(almostempty_inac_p);
140
141 wr_ptr_coverage             :cover property(wr_ptr_p);
142 rd_ptr_coverage             :cover property(rd_ptr_p);
143
144 count_wr_coverage           :cover property(count_wr_p);
145 count_rd_coverage           :cover property(count_rd_p);
146 count_wr_rd_coverage        :cover property(count_wr_rd_p);
147 countfull_wr_rd_coverage    :cover property(countfull_wr_rd_p);
148 countempty_wr_rd_coverage   :cover property(countempty_wr_rd_p);
149 always_comb begin
150     if (!rst_n) begin
151         rst_overflow_assert   :assert final (overflow==0);
152         rst_underflow_assert  :assert final (underflow==0);
153         rst_wr_ack_assert     :assert final (wr_ack==0);
154         rst_wr_ptr_assert     :assert final (wr_ptr==0);
155         rst_rd_ptr_assert     :assert final (rd_ptr==0);
156         rst_count_assert      :assert final (count==0);
157
158         rst_overflow_coverage :cover final (overflow==0);
159         rst_underflow_coverage:cover final (underflow==0);
160         rst_wr_ack_coverage   :cover final (wr_ack==0);
161         rst_wr_ptr_coverage   :cover final (wr_ptr==0);
162         rst_rd_ptr_coverage   :cover final (rd_ptr==0);
163         rst_count_coverage    :cover final (count==0);
164     end
165 end
166
167 endmodule
```

Top

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: TOP_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
import fifo_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
module top();
 bit clk;
initial begin
    clk=0;
    forever
        #1 clk=~clk;
end

fifo_if f_if(clk);

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

bind FIFO fifo_sva SVA(
         .data_in(f_if.data_in),
         .wr_en(f_if.wr_en),
         .rd_en(f_if.rd_en),
         .clk(f_if.clk),
         .rst_n(f_if.rst_n),
         .full(f_if.full),
         .empty(f_if.empty),
         .almostfull(f_if.almostfull),
         .almostempty(f_if.almostempty),
         .wr_ack(f_if.wr_ack),
         .overflow(f_if.overflow),
         .underflow(f_if.underflow),
         .data_out(f_if.data_out),
         .wr_ptr(DUT.wr_ptr),
         .rd_ptr(DUT.rd_ptr),
         .count(DUT.count)
        );
```

```
initial begin
    uvm_config_db#(virtual fifo_if)::set(null, "uvm_test_top", "fifo_IF", f_if);
    run_test("fifo_test");
end

endmodule
```

Configuration

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Configration_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_config extends  uvm_object;
        `uvm_object_utils(fifo_config)

        virtual fifo_if fifo_vif;

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

    endclass : fifo_config
endpackage : fifo_config_pkg
```

Sequence Item

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Sequence_item_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_seq_item_pkg;
    import shared_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_seq_item extends  uvm_sequence_item;
        `uvm_object_utils(fifo_seq_item)

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

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

    function string Print_data_out();
        return $sformatf("data_in=%d,rst_n=%d,wr_en=%d,rd_en=%d,data_out=%d",
                         data_in,rst_n,wr_en,rd_en,data_out);
    endfunction : Print_data_out

    function string Print_flags();
        return $sformatf("%s,wr_ack=%d,overflow=%d,underflow=%d,full=%d,empty=%d,almostfull=%d,almostempty=%d",
                          Print_data_out(),wr_ack,overflow,underflow,full,empty,almostfull,almostempty);
    endfunction : Print_flags

/*****************************************************************************************************************/
/*****************************************Constraints*************************************************************/

    constraint Reset_prob {
        rst_n dist {ACTIVE_RESET:/RESET_ON_DIST,INACRIVE_RESET:/100-RESET_ON_DIST};
    }

    constraint Write_only {
        wr_en==ACTIVE;
        rd_en==INACTIVE;
        rst_n==INACRIVE_RESET;
    }

    constraint Read_only {
        wr_en==INACTIVE;
        rd_en==ACTIVE;
        rst_n==INACRIVE_RESET;
    }

    constraint Write_Read {
        wr_en dist {ACTIVE:/WR_EN_ON_DIST,INACTIVE:/100-WR_EN_ON_DIST};
        rd_en dist {ACTIVE:/RD_EN_ON_DIST,INACTIVE:/100-RD_EN_ON_DIST};
    }
```

```
    constraint Write_Read_always {
        wr_en==ACTIVE;
        rd_en==ACTIVE;
        rst_n==INACRIVE_RESET;
    }

    constraint data_in_prob {
        data_in dist {MAX:/10,ZERO:/10,[ZERO:MAX]:/80};
    }

    endclass : fifo_seq_item
endpackage : fifo_seq_item_pkg
```

Sequences

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Sequences_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_sequence_pkg;
    import shared_pkg::*;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
/*****************************************************************************************************/
    class reset_assert extends  uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(reset_assert)

        fifo_seq_item seq_item;

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

        task body();
            seq_item=fifo_seq_item::type_id::create("seq_item");
            start_item(seq_item);
            seq_item.rd_en=0;
            seq_item.wr_en=0;
            seq_item.data_in=0;
            seq_item.rst_n=0;
            finish_item(seq_item);
        endtask : body
    endclass : reset_assert
/*****************************************************************************************************/
    class Write_only extends  uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(Write_only)

        fifo_seq_item seq_item;

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

        task body();
            repeat(10) begin
                seq_item=fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.constraint_mode(0);
                seq_item.Write_only.constraint_mode(1);//ON the Write_only constraint
                seq_item.data_in_prob.constraint_mode(1);//ON the data_in_prob constraint
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body
    endclass : Write_only
/*****************************************************************************************************/
    class Read_only extends  uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(Read_only)

        fifo_seq_item seq_item;

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

        task body();
            repeat(10) begin
                seq_item=fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.constraint_mode(0);
                seq_item.Read_only.constraint_mode(1);//ON the Read_only constraint
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body
    endclass : Read_only
/*****************************************************************************************************/
```

```
/*****************************************************************************************************/
    class Write_Read extends  uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(Write_Read)

        fifo_seq_item seq_item;

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

        task body();
            repeat(1000) begin
                seq_item=fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.constraint_mode(0);
                seq_item.Write_Read.constraint_mode(1);//ON the Write_Read constraint
                seq_item.data_in_prob.constraint_mode(1);//ON the data_in_prob constraint
                seq_item.Reset_prob.constraint_mode(1);//ON the Reset_prob constraint
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body
    endclass : Write_Read
/*****************************************************************************************************/
    class Write_Read_Always extends  uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(Write_Read_Always)

        fifo_seq_item seq_item;

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

        task body();
            repeat(10) begin
                seq_item=fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.constraint_mode(0);
                seq_item.Write_Read_always.constraint_mode(1);//ON the Write_Read_always constraint
                seq_item.data_in_prob.constraint_mode(1);//ON the data_in_prob constraint
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask : body
    endclass : Write_Read_Always
/*****************************************************************************************************/
endpackage : fifo_sequence_pkg
```

Sequencer

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Sequencer FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_sequencer_pkg;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_sequencer extends uvm_sequencer  #(fifo_seq_item);
        `uvm_component_utils(fifo_sequencer)

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

    endclass : fifo_sequencer
endpackage : fifo_sequencer_pkg
```

Driver

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Driver_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_driver_pkg;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_driver extends uvm_driver #(fifo_seq_item);
        `uvm_component_utils(fifo_driver)

        virtual fifo_if fifo_vif;
        fifo_seq_item stim_seq_item;

        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",this);
                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.data_in=stim_seq_item.data_in;
                fifo_vif.rst_n=stim_seq_item.rst_n;
                @(negedge fifo_vif.clk);
                seq_item_port.item_done();
            end
        endtask : run_phase

    endclass : fifo_driver
endpackage : fifo_driver_pkg
```

Monitor

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Monitor_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_monitor_pkg;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_monitor extends  uvm_monitor;
        `uvm_component_utils(fifo_monitor)

        virtual fifo_if fifo_vif;
        fifo_seq_item seq_item;
        uvm_analysis_port #(fifo_seq_item) mon_aport;

        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_aport=new("mon_aport",this);
        endfunction : build_phase
```

```
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                seq_item=fifo_seq_item::type_id::create("seq_item");
                @(negedge fifo_vif.clk);
                //Input
                seq_item.rd_en=fifo_vif.rd_en;
                seq_item.rst_n=fifo_vif.rst_n;
                seq_item.wr_en=fifo_vif.wr_en;
                seq_item.data_in=fifo_vif.data_in;
                //Output
                seq_item.full=fifo_vif.full;
                seq_item.empty=fifo_vif.empty;
                seq_item.wr_ack=fifo_vif.wr_ack;
                seq_item.overflow=fifo_vif.overflow;
                seq_item.underflow=fifo_vif.underflow;
                seq_item.almostfull=fifo_vif.almostfull;
                seq_item.almostempty=fifo_vif.almostempty;
                seq_item.data_out=fifo_vif.data_out;
                mon_aport.write(seq_item);
                `uvm_info("run_phase",seq_item.Print_flags(),UVM_HIGH);
            end
        endtask : run_phase

    endclass : fifo_monitor
endpackage : fifo_monitor_pkg
```

Agent

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Agent_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_agent_pkg;
    import fifo_monitor_pkg::*;
    import fifo_driver_pkg::*;
    import fifo_sequencer_pkg::*;
    import fifo_config_pkg::*;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class fifo_agent extends  uvm_agent;
        `uvm_component_utils(fifo_agent)

        fifo_monitor monitor;
        fifo_driver driver;
        fifo_sequencer sqr;
        fifo_config fifo_cfg;
        uvm_analysis_port #(fifo_seq_item) agt_aport;

        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))
                `uvm_fatal("build_phase","Agent - unable to get configuration object");

            monitor=fifo_monitor::type_id::create("monitor",this);
            driver=fifo_driver::type_id::create("driver",this);
            sqr=fifo_sequencer::type_id::create("sqr",this);

            agt_aport=new("agt_aport",this);
        endfunction : build_phase

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            driver.fifo_vif=fifo_cfg.fifo_vif;
            monitor.fifo_vif=fifo_cfg.fifo_vif;
            driver.seq_item_port.connect(sqr.seq_item_export);
            monitor.mon_aport.connect(agt_aport);
        endfunction : connect_phase

    endclass : fifo_agent
endpackage : fifo_agent_pkg
```

Scoreboard

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Scoeboard_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_scoreboard_pkg;
    import shared_pkg::*;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_scoreboard extends  uvm_scoreboard;
        `uvm_component_utils(fifo_scoreboard)

        Logic [FIFO_WIDTH-1:0] data_out_ref;
        Logic full_ref, empty_ref, almostfull_ref, almostempty_ref, overflow_ref, underflow_ref,wr_ack_ref;

        Logic [FIFO_WIDTH]mem_ref[FIFO_DEPTH];
        Logic [MAX_FIFO_ADDR-1:0] rd_ptr = 0,wr_ptr = 0;
        Logic [MAX_FIFO_ADDR:0] count =0;

        uvm_analysis_export     #(fifo_seq_item) sb_export;
        uvm_tlm_analysis_fifo   #(fifo_seq_item) sb_fifo;
        fifo_seq_item   seq_item_sb;

        int error_dataout_count     =0;
        int error_flags_count       =0;
        int correct_dataout_count   =0;
        int correct_flags_count     =0;

        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);
                reference_model(seq_item_sb);
                if(seq_item_sb.data_out!==data_out_ref) begin
                    `uvm_error("run_phase",$sformatf("Compair dataout faild:%s ,data_out_ref=%0d",seq_item_sb.Print_data_out(),data_out_ref));
                    error_dataout_count++;
                end
                else begin
                    `uvm_info("run_phase",$sformatf("Compair dataout correct:%s",seq_item_sb.Print_data_out()),UVM_HIGH);
                    correct_dataout_count++;
                end
                if({seq_item_sb.wr_ack, seq_item_sb.overflow, seq_item_sb.underflow, seq_item_sb.full, seq_item_sb.empty ,seq_item_sb.almostfull, seq_item_sb.almostempty}!=
                    {wr_ack_ref,overflow_ref,underflow_ref,full_ref,empty_ref,almostfull_ref,almostempty_ref}) begin
                    `uvm_error("run_phase",$sformatf("Compair dataout faild:%s",seq_item_sb.Print_flags()));
                    error_flags_count++;
                end
                else begin
                    `uvm_info("run_phase",$sformatf("Compair dataout correct:%s",seq_item_sb.Print_flags()),UVM_HIGH);
                    correct_flags_count++;
                end
            end
        endtask : run_phase

        function void reference_model(fifo_seq_item seq_chk);
        if(!seq_chk.rst_n) begin
            rd_ptr=0;wr_ptr=0;count=0;wr_ack_ref=0;overflow_ref=0;underflow_ref=0;
        end
```

```
        else begin
            //Condition for Write seq and  wr_ack flag
            if(seq_chk.wr_en&&count!=FIFO_DEPTH)begin
                mem_ref[wr_ptr]=seq_chk.data_in;
                wr_ack_ref=1;
                wr_ptr++;
            end
            else begin
                wr_ack_ref=0;
            end
            //Condition for overflow flag
            if (seq_chk.wr_en&&count==FIFO_DEPTH) begin
                overflow_ref=1;
            end
            else begin
                overflow_ref=0;
            end
            //Condition for underflow flag
            if (seq_chk.rd_en&&count==0) begin
                underflow_ref=1;
            end
            else begin
                underflow_ref=0;
            end
            //Condition for read seq
            if (seq_chk.rd_en&&count!=0) begin
                data_out_ref=mem_ref[rd_ptr];
                rd_ptr++;
            end
            //for counter
            if (seq_chk.wr_en&&!seq_chk.rd_en&&count!=FIFO_DEPTH) begin
                count++;
            end
            else if (!seq_chk.wr_en&&seq_chk.rd_en&&count!=0) begin
                count--;
            end
            else if(seq_chk.wr_en&&seq_chk.rd_en&&count==FIFO_DEPTH)
                count--;
            else if(seq_chk.wr_en&&seq_chk.rd_en&&count==0)
                count++;

        end
        //assigns for flags
            full_ref = (count == FIFO_DEPTH)? 1 : 0;
            empty_ref = (count == 0)? 1 : 0;
            almostfull_ref = (count == FIFO_DEPTH-1)? 1 : 0;
            almostempty_ref = (count == 1)? 1 : 0;
    endfunction
```

```
    function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase",$sformatf("Total successful output: %0d",correct_dataout_count),UVM_LOW);
            `uvm_info("report_phase",$sformatf("Total faild output: %0d",error_dataout_count),UVM_LOW);
            `uvm_info("report_phase",$sformatf("Total successful flags: %0d",correct_flags_count),UVM_LOW);
            `uvm_info("report_phase",$sformatf("Total faild flags: %0d",error_flags_count),UVM_LOW);
        endfunction : report_phase

    endclass : fifo_scoreboard
endpackage : fifo_scoreboard_pkg
```

Coverage collector

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Coverage_collector_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_coverage_pkg;
    import fifo_seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_coverage extends  uvm_component;
        `uvm_component_utils(fifo_coverage)

        uvm_analysis_export     #(fifo_seq_item) cov_export;
        uvm_tlm_analysis_fifo   #(fifo_seq_item) cov_fifo;
        fifo_seq_item seq_item;

        covergroup covcode;
            rst_n_p      :coverpoint seq_item.rst_n {
                bins rst_n[] = {0,1};
            }
            wr_en_p      :coverpoint seq_item.wr_en{
                bins wr_en[] = {0,1};
            }
            rd_en_p      :coverpoint seq_item.rd_en{
                bins rd_en[] = {0,1};
            }
            wr_ack_p     :coverpoint seq_item.wr_ack{
                bins wr_ack[] = {0,1};
            }
```

```
            }
            overflow_p   :coverpoint seq_item.overflow{
                bins overflow[] = {0,1};
            }
            full_p       :coverpoint seq_item.full{
                bins full[] = {0,1};
            }
            empty_p      :coverpoint seq_item.empty{
                bins empty[] = {0,1};
            }
            almostfull_p :coverpoint seq_item.almostfull{
                bins almostfull[] = {0,1};
            }
            almostempty_p:coverpoint seq_item.almostempty{
                bins almostempty[] = {0,1};
            }
            underflow_p  :coverpoint seq_item.underflow{
                bins underflow[] = {0,1};
            }

            cross wr_en_p,rd_en_p,wr_ack_p{
                illegal_bins wr_dis_ack_en = binsof(wr_en_p) intersect{0} &&binsof(wr_ack_p) intersect{1};
            }
            cross wr_en_p,rd_en_p,overflow_p{
                illegal_bins wr_dis_over_en = binsof(wr_en_p) intersect{0} &&binsof(overflow_p) intersect{1};
            }
            cross wr_en_p,rd_en_p,full_p{
                illegal_bins wr_full_en=binsof(rd_en_p)intersect{1}&&binsof(full_p)intersect{1};
            }
            cross wr_en_p,rd_en_p,empty_p;
            cross wr_en_p,rd_en_p,almostfull_p;
            cross wr_en_p,rd_en_p,almostempty_p;
            cross wr_en_p,rd_en_p,underflow_p{
                illegal_bins rd_dis_uflo_en=binsof(rd_en_p)intersect{0}&&binsof(underflow_p)intersect{1};
            }
        endgroup : covcode


        function  new(string name = "fifo_coverage" , uvm_component parent = null);
            super.new(name,parent);
            covcode=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);
                covcode.sample();
            end
        endtask : run_phase

    endclass : fifo_coverage
endpackage : fifo_coverage_pkg
```

Environment

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Enviroment_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_env_pkg;
    import fifo_scoreboard_pkg::*;
    import fifo_coverage_pkg::*;
    import fifo_agent_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_env extends  uvm_env;
        `uvm_component_utils(fifo_env)

        fifo_scoreboard sb;
        fifo_coverage cov;
        fifo_agent agt;

        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_aport.connect(sb.sb_export);
            agt.agt_aport.connect(cov.cov_export);
        endfunction : connect_phase
    endclass : fifo_env
endpackage : fifo_env_pkg
```

Test

```
////////////////////////////////////////////////////////////////////////////////
// Author: Abanob Evram
// Description: Test_FIFO
// Date: October, 2024
////////////////////////////////////////////////////////////////////////////////
package fifo_test_pkg;
    import fifo_env_pkg::*;
    import fifo_config_pkg::*;
    import fifo_sequence_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh";

    class fifo_test extends  uvm_test;
        `uvm_component_utils(fifo_test)

        fifo_env env;
        fifo_config fifo_cfg;

        reset_assert reset_seq;
        Write_only Write_seq;
        Read_only Read_seq;
        Write_Read Write_Read_seq;
        Write_Read_Always Write_Read_Always_Seq;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env          =fifo_env::type_id::create("env",this);
            fifo_cfg     =fifo_config::type_id::create("fifo_cfg",this);
            reset_seq=reset_assert::type_id::create("reset_seq",this);
            Write_seq=Write_only::type_id::create("Write_seq",this);
            Read_seq=Read_only::type_id::create("Read_seq",this);
            Write_Read_seq=Write_Read::type_id::create("Write_Read_seq",this);
            Write_Read_Always_Seq=Write_Read_Always::type_id::create("Write_Read_Always_Seq",this);

            if(!uvm_config_db#(virtual fifo_if)::get(this, "", "fifo_IF",fifo_cfg.fifo_vif))
                `uvm_fatal("build_phase","Test - unable to get the virtual interface of the shift_reg from the uvm_config_db");
            uvm_config_db#(fifo_config)::set(this, "*", "CFG",fifo_cfg);
        endfunction : build_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","Write only sequence Started",UVM_LOW)
            Write_seq.start(env.agt.sqr);

            `uvm_info("run_phase","Read only sequence Started",UVM_LOW)
            Read_seq.start(env.agt.sqr);

            `uvm_info("run_phase","Write Read sequence Started",UVM_LOW)
            Write_Read_seq.start(env.agt.sqr);

            `uvm_info("run_phase","Write Read Always sequence Started",UVM_LOW)
            Write_Read_Always_Seq.start(env.agt.sqr);

            `uvm_info("run_phase","Test - End stimulus Generation",UVM_LOW);
            phase.drop_objection(this);
        endtask : run_phase

    endclass : fifo_test

endpackage : fifo_test_pkg
```

List file

```
../Codes/Shared_pkg/shared_pkg.sv
../Codes/Design/FIFO.sv
../Codes/Assertions/fifo_sva.sv
../Codes/Interface/fifo_if.sv

../Codes/Objects/Sequence/fifo_seq_item_pkg.sv
../Codes/Objects/Sequence/fifo_sequence_pkg.sv
../Codes/Objects/Configrstion/fifo_config_pkg.sv

../Codes/Test/Enviroment/Agent/Sequencer/fifo_sequencer_pkg.sv
../Codes/Test/Enviroment/Agent/Monitor/fifo_monitor_pkg.sv
../Codes/Test/Enviroment/Agent/Driver/fifo_driver_pkg.sv
../Codes/Test/Enviroment/Agent/fifo_agent_pkg.sv

../Codes/Test/Enviroment/Coverage_Collector/fifo_coverage_pkg.sv
../Codes/Test/Enviroment/Scoreboard/fifo_scoreboard_pkg.sv
../Codes/Test/Enviroment/fifo_env_pkg.sv

../Codes/Test/fifo_test.sv
../Codes/Top/fifo_top.sv
```

Do file

```
vlib work
vlog -f list.list +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave -position insertpoint sim:/top/f_if/*
run -all
```

Bug report

BUG_1: The signal wr_ack is sequential so it must be reset to zero when reset is activate

FIX: -

```
21    always @(posedge clk or negedge rst_n) begin
22        if (!rst_n) begin
23            wr_ptr <= 0;
24            wr_ack<=0;//reset the wr_ack when rst_n is active
```

BUG_2: The signal overflow is sequential so it must be reset to zero when reset is activated

FIX: -

```
21    always @(posedge clk or negedge rst_n) begin
22        if (!rst_n) begin
23            wr_ptr <= 0;
24            wr_ack<=0;//reset the wr_ack when rst_n is active
25            overflow<=0;//reset the overflow when rst_n is active
26        end
```

BUG_3: The overflow always equal zero expect FIFO is full && operation is write.

FIX: -

```
27        else begin//Edit this condition and handle the overflow cases
28            if (wr_en && count < FIFO_DEPTH) begin
29                mem[wr_ptr] <= data_in;
30                wr_ack <= 1;
31                wr_ptr <= wr_ptr + 1;
32            end
33            else begin
34                wr_ack <= 0;
35            end
36            if(wr_en&&count==FIFO_DEPTH)
37                overflow<=1;
38            else
39                overflow<=0;
40        end
41    end
```

BUG_4: The underflow is sequential so it must be reset to zero when reset is activate .

FIX: -

```
43   always @(posedge clk or negedge rst_n) begin//Add the underflow flag because it is sequential
44       if (!rst_n) begin
45           rd_ptr <= 0;
46           underflow<=0;//reset the underflow when rst_n is active
```

BUG_5: Handle the underflow cases in the read always block

FIX: -

```
43   always @(posedge clk or negedge rst_n) begin//Add the underflow flag because it is sequential
44       if (!rst_n) begin
45           rd_ptr <= 0;
46           underflow<=0;//reset the underflow when rst_n is active
47       end
48       else begin//Edit this condition and handle the underflow cases
49           if (rd_en && count != 0) begin
50               data_out <= mem[rd_ptr];
51               rd_ptr <= rd_ptr + 1;
52           end
53           if(rd_en&&count==0)
54               underflow<=1;
55           else
56               underflow<=0;
57       end
58   end
```

BUG_6: Counter is not handle the case of wr_en and rd_en both are high if operation is write or read operation.

FIX: -

```
60   always @(posedge clk or negedge rst_n) begin
61       if (!rst_n) begin
62           count <= 0;
63       end
64       else begin//Add two cases for the count
65           if  ( ({wr_en, rd_en} == 2'b10) && !full)
66               count <= count + 1;
67           else if ( ({wr_en, rd_en} == 2'b01) && !empty)
68               count <= count - 1;
69           else if ( ({wr_en, rd_en} == 2'b11) && full)//1
70               count <= count -1;
71           else if ( ({wr_en, rd_en} == 2'b11) && empty)//2
72               count <= count + 1;
73       end
74   end
```

BUG_6: Almostfull is high when internal signal "count" is less than FIFO_DEPTH by one

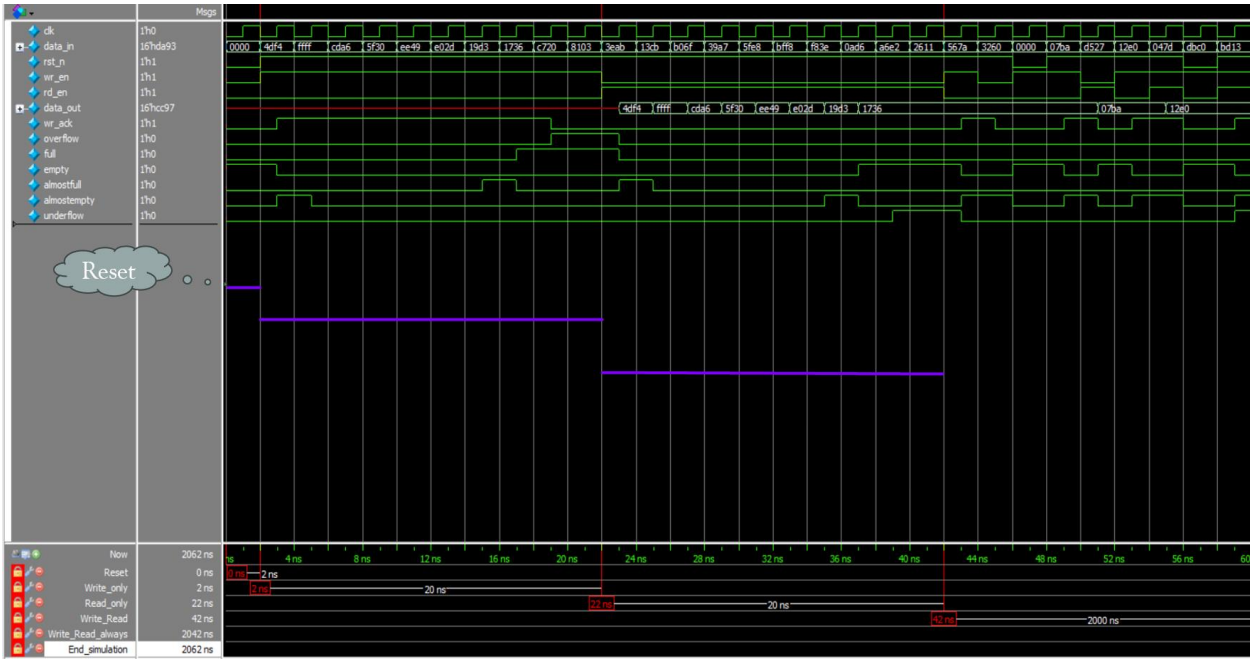
FIX: -

```
assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //fix the condition of this flag
```

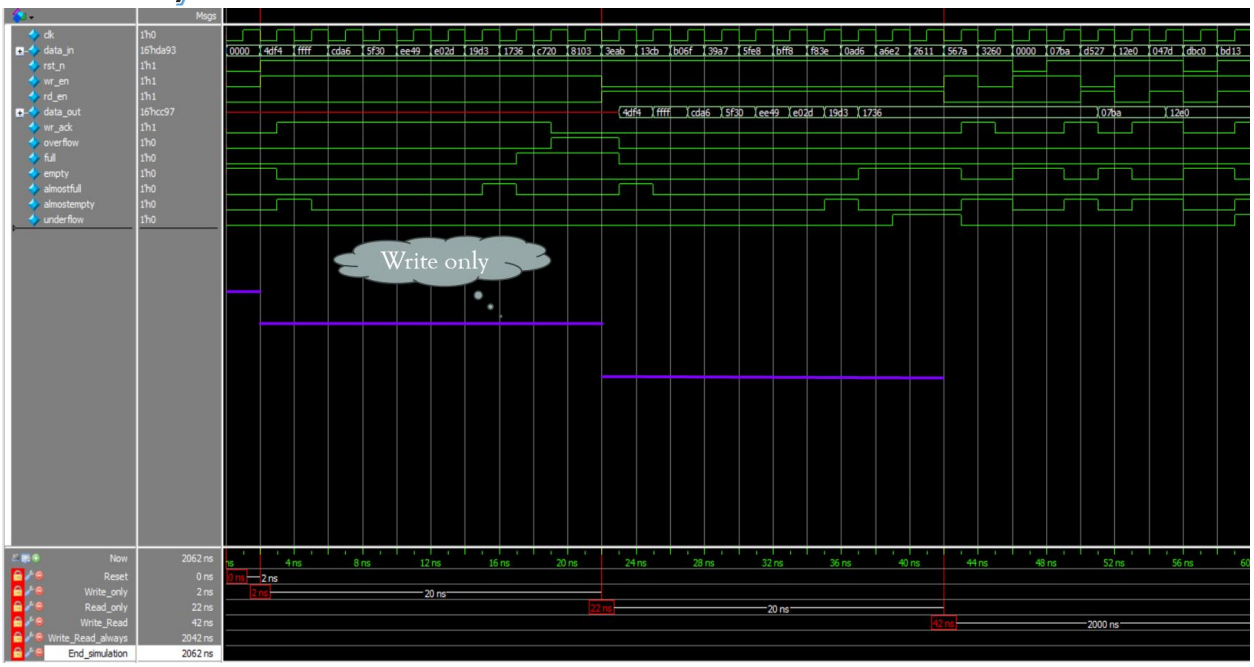
QuestaSim snippets

```
# UVM_INFO ../Codes/Test/fifo_test.sv(48) @ 0: uvm_test_top [run_phase] Reset asserted
# UVM_INFO ../Codes/Test/fifo_test.sv(51) @ 2: uvm_test_top [run_phase] Write only sequence Started
# UVM_INFO ../Codes/Test/fifo_test.sv(54) @ 22: uvm_test_top [run_phase] Read only sequence Started
# UVM_INFO ../Codes/Test/fifo_test.sv(57) @ 42: uvm_test_top [run_phase] Write Read sequence Started
# UVM_INFO ../Codes/Test/fifo_test.sv(60) @ 2042: uvm_test_top [run_phase] Write Read Always sequence Started
# UVM_INFO ../Codes/Test/fifo_test.sv(63) @ 2062: uvm_test_top [run_phase] Test - End stimulus Generation
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 2062: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO ../Codes/Test/Enviroment/Scoreboard/fifo_scoreboard_pkg.sv(125) @ 2062: uvm_test_top.env.sb [report_phase] Total successful output: 1031
# UVM_INFO ../Codes/Test/Enviroment/Scoreboard/fifo_scoreboard_pkg.sv(126) @ 2062: uvm_test_top.env.sb [report_phase] Total faild output: 0
# UVM_INFO ../Codes/Test/Enviroment/Scoreboard/fifo_scoreboard_pkg.sv(127) @ 2062: uvm_test_top.env.sb [report_phase] Total successful flags: 1031
# UVM_INFO ../Codes/Test/Enviroment/Scoreboard/fifo_scoreboard_pkg.sv(128) @ 2062: uvm_test_top.env.sb [report_phase] Total faild flags: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   14
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     4
# [run_phase]     6
```

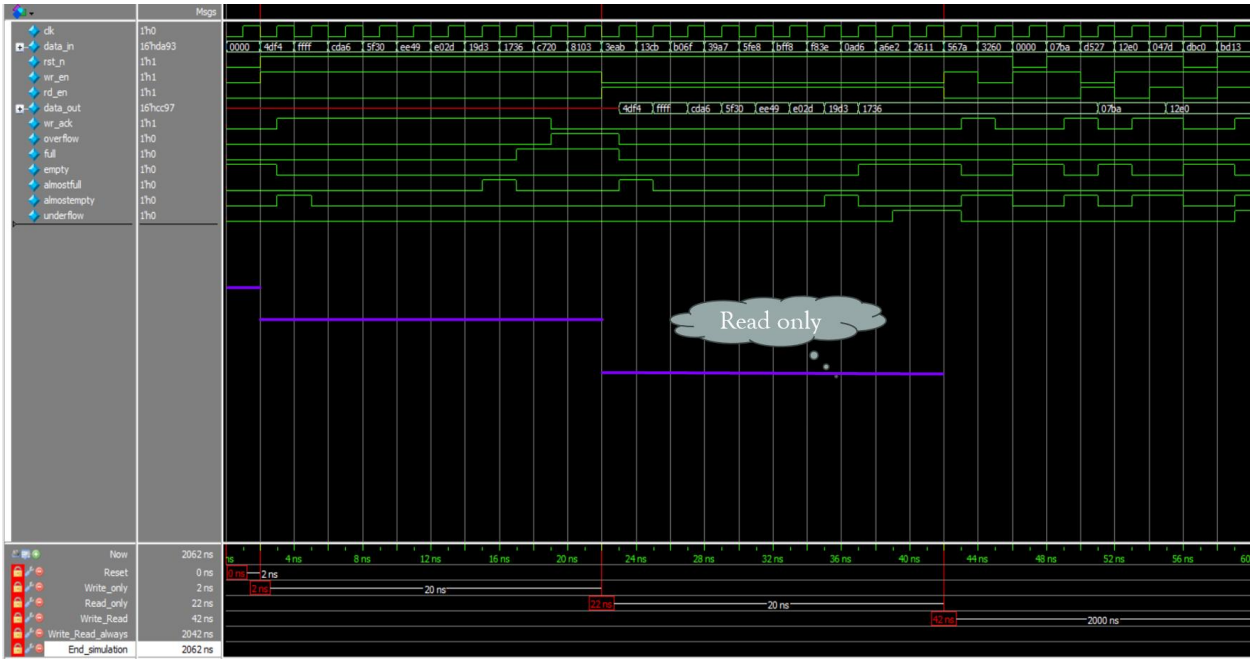
1. **Activate reset:** To reset all input siganls at the start of the simulation



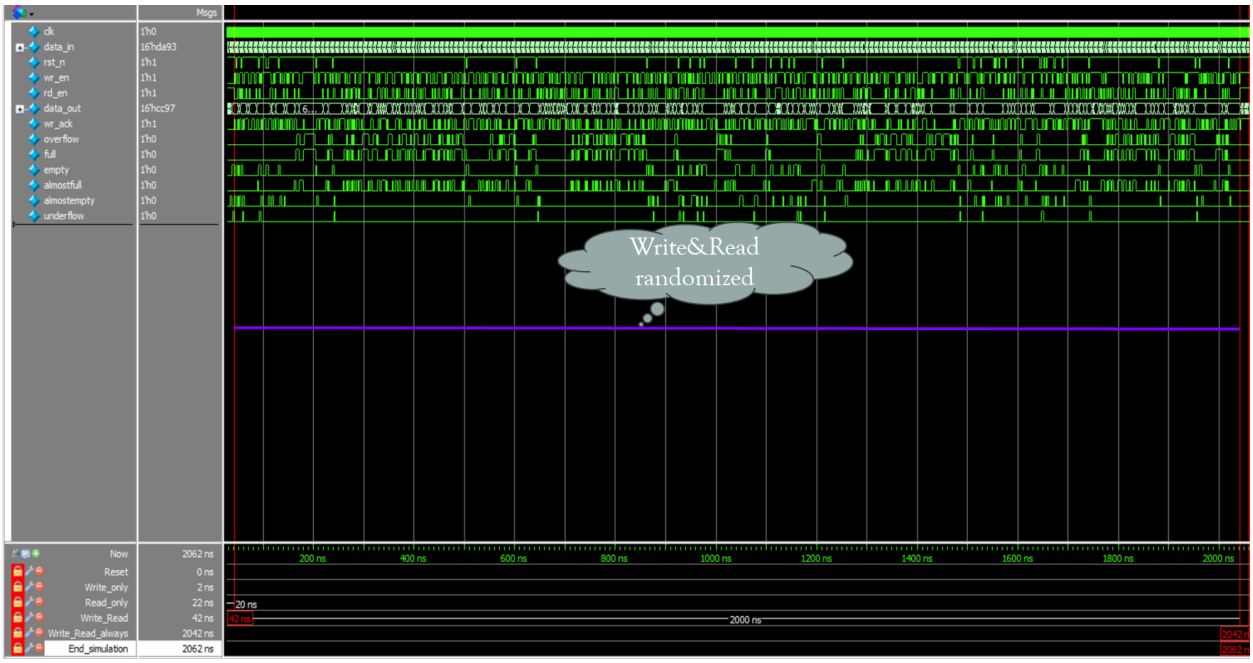
2. **Write only:** Write 10 iterations to make the FIFO full



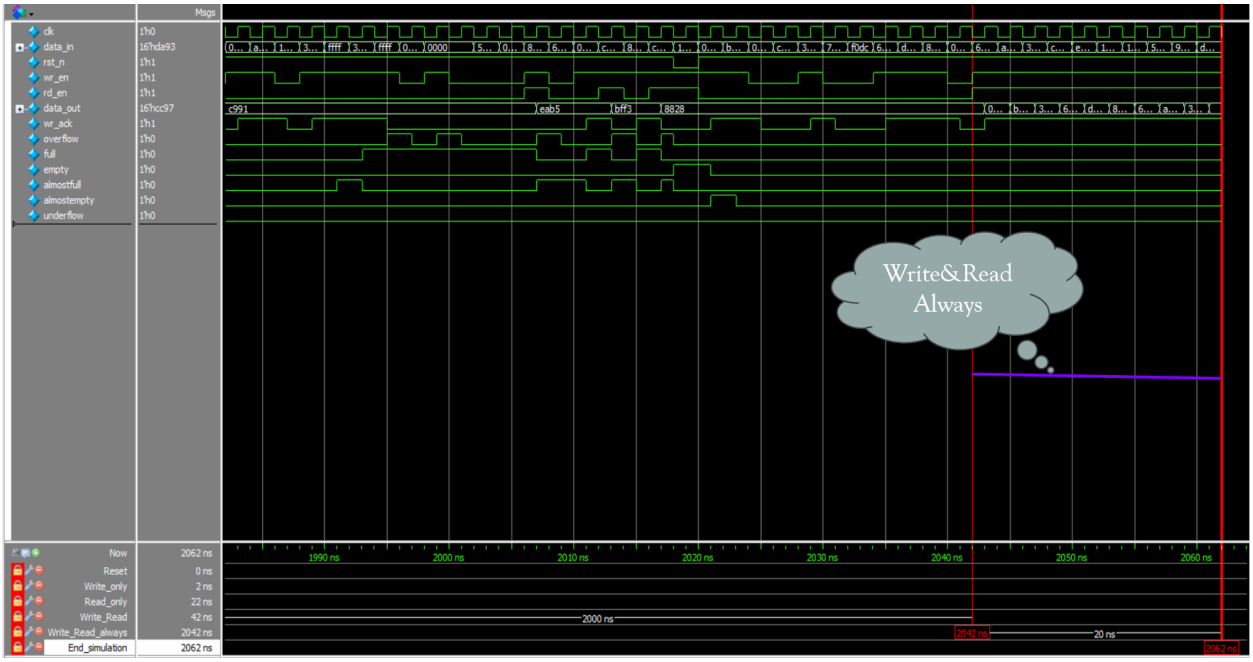
3. **Read only:** Read 10 iterations to make the FIFO empty



4. **Write Read seq:** Randomize all inputs 1000 iteration to check the functionality of the design



5. **Write Read always:** Activate the Write and read sequences for 10 iterations



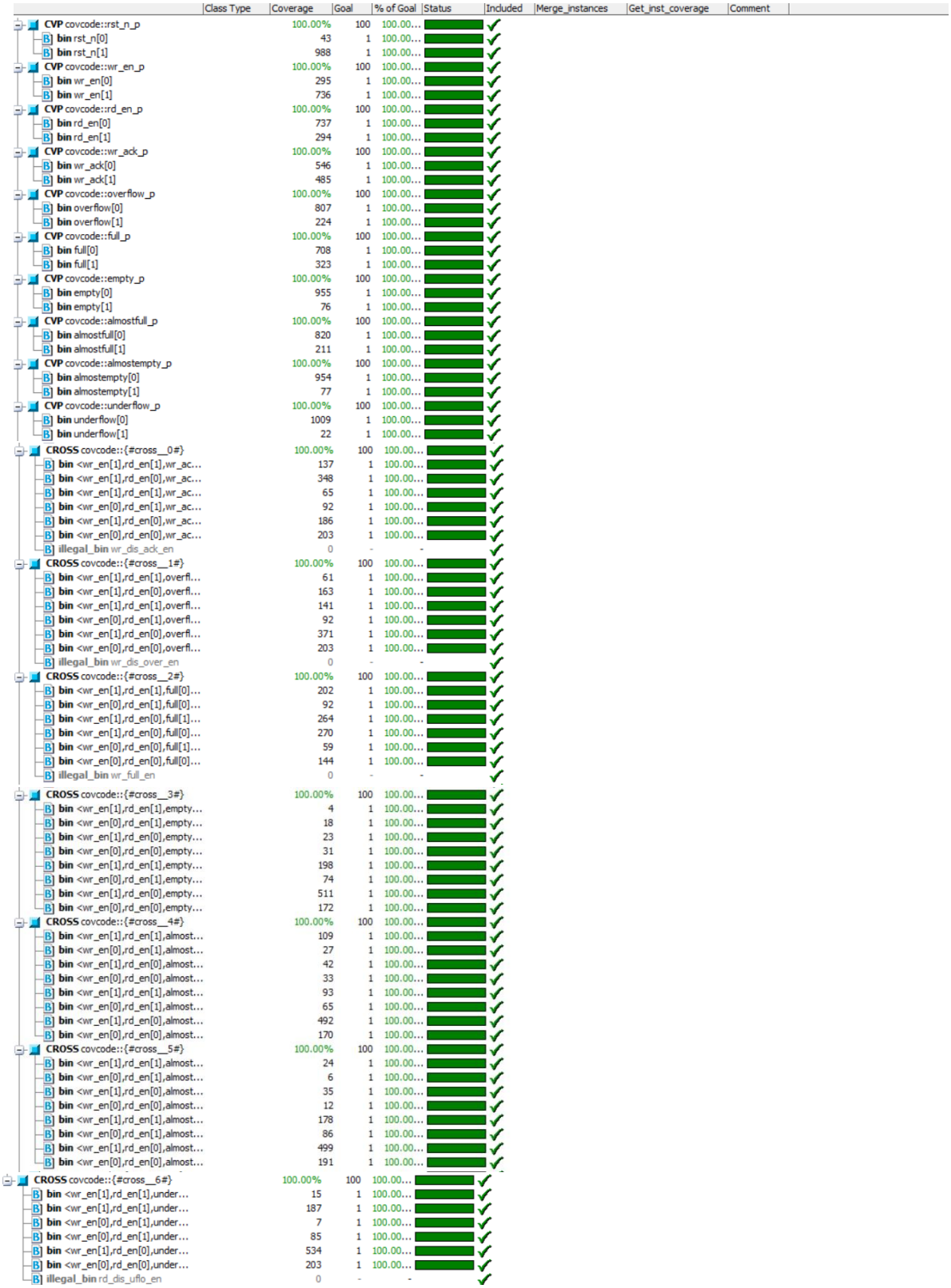
Sequential Domain Coverage report

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/SVA/overflow_coverage	SVA	✓	Off	214	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/underflow_coverage	SVA	✓	Off	21	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/wr_ack_coverage	SVA	✓	Off	462	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/overflow_inac_coverage	SVA	✓	Off	732	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/underflow_inac_coverage	SVA	✓	Off	925	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/wr_ack_inac_coverage	SVA	✓	Off	484	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/full_coverage	SVA	✓	Off	310	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/almostfull_coverage	SVA	✓	Off	203	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/empty_coverage	SVA	✓	Off	116	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/almostempty_coverage	SVA	✓	Off	72	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/full_inac_coverage	SVA	✓	Off	721	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/almostfull_inac_coverage	SVA	✓	Off	828	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/empty_inac_coverage	SVA	✓	Off	915	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/almostempty_inac_coverage	SVA	✓	Off	959	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/wr_ptr_coverage	SVA	✓	Off	462	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rd_ptr_coverage	SVA	✓	Off	255	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/count_wr_coverage	SVA	✓	Off	333	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/count_rd_coverage	SVA	✓	Off	84	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/count_wr_rd_coverage	SVA	✓	Off	114	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/countfull_wr_rd_coverage	SVA	✓	Off	57	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/countempty_wr_rd_coverage	SVA	✓	Off	15	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_overflow_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_underflow_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_wr_ack_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_wr_ptr_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_rd_ptr_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0
/top/DUT/SVA/rst_count_coverage	SVA	✓	Off	42	1	Unlimit...	1	100%		✓	0	0	0 ns	0

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Includ
/top/DUT/SVA/full_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count!=8)|...	✓
/top/DUT/SVA/rst_rd_ptr_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (rd_ptr==0)	✓
/fifo_sequence_pkg::Write_only::body/#ublk#40571367#42/immed__48	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/fifo_sequence_pkg::Read_only::body/#ublk#40571367#64/immed__69	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/fifo_sequence_pkg::Write_Read::body/#ublk#40571367#85/immed__92	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/fifo_sequence_pkg::Write_Read_Always::body/#ublk#40571367#108/immed__...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/DUT/SVA/overflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/underflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/wr_ack_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/overflow_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/underflow_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/wr_ack_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/full_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==8)...	✓
/top/DUT/SVA/almostfull_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==7)...	✓
/top/DUT/SVA/empty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==0)...	✓
/top/DUT/SVA/almostempty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count==1)...	✓
/top/DUT/SVA/rst_count_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (count==0)	✓
/top/DUT/SVA/almostfull_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count!=7)|...	✓
/top/DUT/SVA/empty_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count!=0)|...	✓
/top/DUT/SVA/almostempty_inac_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (count!=1)|...	✓
/top/DUT/SVA/wr_ptr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/rd_ptr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/count_wr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/count_rd_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/count_wr_rd_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/countfull_wr_rd_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/countempty_wr_rd_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/top/DUT/SVA/rst_overflow_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~overflow)	✓
/top/DUT/SVA/rst_underflow_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~underflow)	✓
/top/DUT/SVA/rst_wr_ack_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~wr_ack)	✓
/top/DUT/SVA/rst_wr_ptr_assert	Immediate	SVA	on	0	1	-	-	-	-		off	assert (wr_ptr==0)	✓

Coverage groups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/fifo_coverage_pkg/fifo_coverage		100.00%							
TYPE covcode		100.00%	100	100.00...		✓		auto(1)	
CVP covcode::rst_n_p		100.00%	100	100.00...		✓			
CVP covcode::wr_en_p		100.00%	100	100.00...		✓			
CVP covcode::rd_en_p		100.00%	100	100.00...		✓			
CVP covcode::wr_ack_p		100.00%	100	100.00...		✓			
CVP covcode::overflow_p		100.00%	100	100.00...		✓			
CVP covcode::full_p		100.00%	100	100.00...		✓			
CVP covcode::empty_p		100.00%	100	100.00...		✓			
CVP covcode::almostfull_p		100.00%	100	100.00...		✓			
CVP covcode::almostempty_p		100.00%	100	100.00...		✓			
CVP covcode::underflow_p		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__0#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__1#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__2#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__3#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__4#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__5#}		100.00%	100	100.00...		✓			
CROSS covcode::{#cross__6#}		100.00%	100	100.00...		✓			



	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
CVP covcode::rst_n_p		100.00%	100	100.00...		✓			
bin rst_n[0]		43	1	100.00...		✓			
bin rst_n[1]		988	1	100.00...		✓			
CVP covcode::wr_en_p		100.00%	100	100.00...		✓			
bin wr_en[0]		295	1	100.00...		✓			
bin wr_en[1]		736	1	100.00...		✓			
CVP covcode::rd_en_p		100.00%	100	100.00...		✓			
bin rd_en[0]		737	1	100.00...		✓			
bin rd_en[1]		294	1	100.00...		✓			
CVP covcode::wr_ack_p		100.00%	100	100.00...		✓			
bin wr_ack[0]		546	1	100.00...		✓			
bin wr_ack[1]		485	1	100.00...		✓			
CVP covcode::overflow_p		100.00%	100	100.00...		✓			
bin overflow[0]		807	1	100.00...		✓			
bin overflow[1]		224	1	100.00...		✓			
CVP covcode::full_p		100.00%	100	100.00...		✓			
bin full[0]		708	1	100.00...		✓			
bin full[1]		323	1	100.00...		✓			
CVP covcode::empty_p		100.00%	100	100.00...		✓			
bin empty[0]		955	1	100.00...		✓			
bin empty[1]		76	1	100.00...		✓			
CVP covcode::almostfull_p		100.00%	100	100.00...		✓			
bin almostfull[0]		820	1	100.00...		✓			
bin almostfull[1]		211	1	100.00...		✓			
CVP covcode::almostempty_p		100.00%	100	100.00...		✓			
bin almostempty[0]		954	1	100.00...		✓			
bin almostempty[1]		77	1	100.00...		✓			
CVP covcode::underflow_p		100.00%	100	100.00...		✓			
bin underflow[0]		1009	1	100.00...		✓			
bin underflow[1]		22	1	100.00...		✓			
CROSS covcode::{#cross__0#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],wr_ac...		137	1	100.00...		✓			
bin <wr_en[1],rd_en[0],wr_ac...		348	1	100.00...		✓			
bin <wr_en[1],rd_en[1],wr_ac...		65	1	100.00...		✓			
bin <wr_en[0],rd_en[1],wr_ac...		92	1	100.00...		✓			
bin <wr_en[1],rd_en[0],wr_ac...		186	1	100.00...		✓			
bin <wr_en[0],rd_en[0],wr_ac...		203	1	100.00...		✓			
illegal_bin wr_dis_ack_en		0	-	-		✓			
CROSS covcode::{#cross__1#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],overfl...		61	1	100.00...		✓			
bin <wr_en[1],rd_en[0],overfl...		163	1	100.00...		✓			
bin <wr_en[1],rd_en[1],overfl...		141	1	100.00...		✓			
bin <wr_en[0],rd_en[1],overfl...		92	1	100.00...		✓			
bin <wr_en[1],rd_en[0],overfl...		371	1	100.00...		✓			
bin <wr_en[0],rd_en[0],overfl...		203	1	100.00...		✓			
illegal_bin wr_dis_over_en		0	-	-		✓			
CROSS covcode::{#cross__2#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],full[0]...		202	1	100.00...		✓			
bin <wr_en[0],rd_en[1],full[0]...		92	1	100.00...		✓			
bin <wr_en[1],rd_en[0],full[1]...		264	1	100.00...		✓			
bin <wr_en[1],rd_en[0],full[0]...		270	1	100.00...		✓			
bin <wr_en[0],rd_en[0],full[1]...		59	1	100.00...		✓			
bin <wr_en[0],rd_en[0],full[0]...		144	1	100.00...		✓			
illegal_bin wr_full_en		0	-	-		✓			
CROSS covcode::{#cross__3#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],empty...		4	1	100.00...		✓			
bin <wr_en[0],rd_en[1],empty...		18	1	100.00...		✓			
bin <wr_en[1],rd_en[0],empty...		23	1	100.00...		✓			
bin <wr_en[0],rd_en[0],empty...		31	1	100.00...		✓			
bin <wr_en[1],rd_en[1],empty...		198	1	100.00...		✓			
bin <wr_en[0],rd_en[1],empty...		74	1	100.00...		✓			
bin <wr_en[1],rd_en[0],empty...		511	1	100.00...		✓			
bin <wr_en[0],rd_en[0],empty...		172	1	100.00...		✓			
CROSS covcode::{#cross__4#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],almost...		109	1	100.00...		✓			
bin <wr_en[0],rd_en[1],almost...		27	1	100.00...		✓			
bin <wr_en[1],rd_en[0],almost...		42	1	100.00...		✓			
bin <wr_en[0],rd_en[0],almost...		33	1	100.00...		✓			
bin <wr_en[1],rd_en[1],almost...		93	1	100.00...		✓			
bin <wr_en[0],rd_en[1],almost...		65	1	100.00...		✓			
bin <wr_en[1],rd_en[0],almost...		492	1	100.00...		✓			
bin <wr_en[0],rd_en[0],almost...		170	1	100.00...		✓			
CROSS covcode::{#cross__5#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],almost...		24	1	100.00...		✓			
bin <wr_en[0],rd_en[1],almost...		6	1	100.00...		✓			
bin <wr_en[1],rd_en[0],almost...		35	1	100.00...		✓			
bin <wr_en[0],rd_en[0],almost...		12	1	100.00...		✓			
bin <wr_en[1],rd_en[1],almost...		178	1	100.00...		✓			
bin <wr_en[0],rd_en[1],almost...		86	1	100.00...		✓			
bin <wr_en[1],rd_en[0],almost...		499	1	100.00...		✓			
bin <wr_en[0],rd_en[0],almost...		191	1	100.00...		✓			
CROSS covcode::{#cross__6#}		100.00%	100	100.00...		✓			
bin <wr_en[1],rd_en[1],under...		15	1	100.00...		✓			
bin <wr_en[1],rd_en[1],under...		187	1	100.00...		✓			
bin <wr_en[0],rd_en[1],under...		7	1	100.00...		✓			
bin <wr_en[0],rd_en[1],under...		85	1	100.00...		✓			
bin <wr_en[1],rd_en[0],under...		534	1	100.00...		✓			
bin <wr_en[0],rd_en[0],under...		203	1	100.00...		✓			
illegal_bin rd_dis_uflo_en		0	-	-		✓			

Code Coverage

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


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

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

Condition Coverage for instance /\top#DUT  --


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

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

Statement Coverage for instance /\top#DUT  --
    --                _                       - --
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        106       106         0   100.00%

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

Toggle Coverage for instance /\top#DUT  --

                                           Node      1H->0L      0L->1H  "Coverage"
```