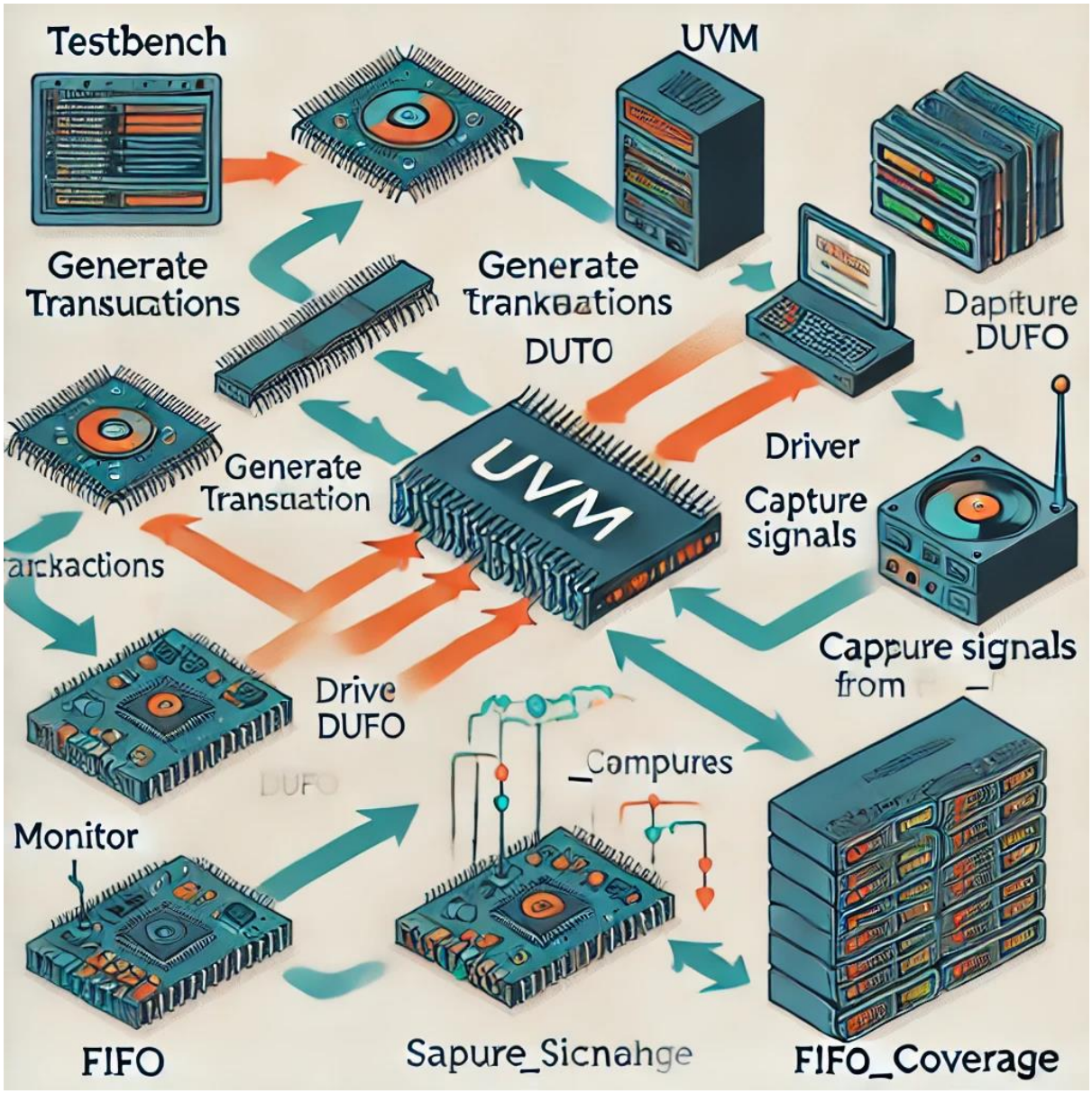
UVM-FIFO



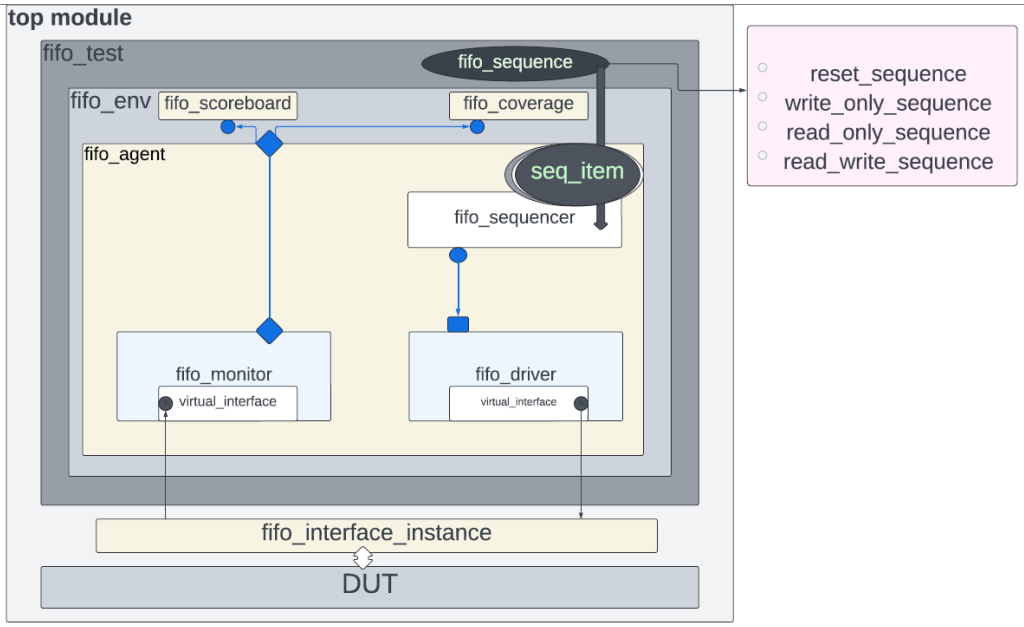
Abdelrahman Hatem Nasr

CONTENTS

Verification Plan

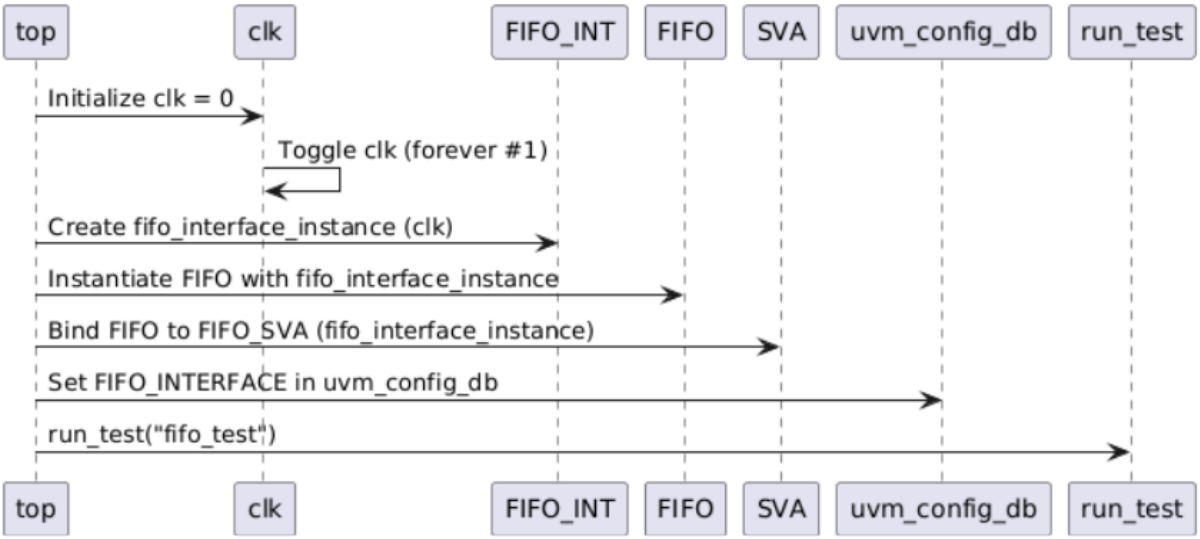
Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	when the reset is asserted all flags registers , write and read pointers should be low	directed at the start of simulation ,then randomized under constraints which make reset to be deasserted at most of time		immediate assertions to check for async reset functionallity
FIFO_2	when read enable is asserted and the fifo is not empty ,the fifo should be popped with correct data out	randomization under constraints on read enable to be high 30% most of the time	cover values of read enable,write enable and almost empty,underflow	golden model to check the output data out
FIFO_3	when write enable is asserted and the fifo is not full ,the fifo should be pushed with correct data in	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and almost full,overflow	golden model to check the output data out which reflect the good asserted data in
FIFO_4	wr_ack' should be asserted when write occurs and FIFO is not full	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and wr_ack	concuurent assertion to check the wr_ack
FIFO_5	full flag should be asserted when fifo is full	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and full flag	concuurent assertion to check the full flag
FIFO_6	empty flag should be asserted when fifo is empty	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and empty flag	concuurent assertion to check the empty flag
FIFO_7	almostfull flag should be asserted when fifo count is fifo depth - 1	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,readd enable and almost full flag	concuurent assertion to check the almost full flag
FIFO_8	almostempty flag should be asserted when fifo count is 1	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and almost empty flag	concuurent assertion to check the almost empty flag
FIFO_9	overflow' flag if full and wr_en it should be assertd	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and overflow flag	concuurent assertion to check the overflow flag
FIFO_10	underflow flag if empty and rd_en it should be assertd	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and underflow flag	concuurent assertion to check the underflow flag
FIFO_11	count shouldn't be higher than fifo depth	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and full flag	concuurent assertion to check if count overflow
FIFO_12	write pointer shouldn't be higher than fifo depth	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and full flag	concuurent assertion to check if write pointer overflow
FIFO_13	read pointer shouldn't be higher than fifo depth	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and empty flag	concuurent assertion to check if read pointer underflow
FIFO_14	count' when wr_en is high and rd_en is low it should increased	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and wr_ack	concuurent assertion to check if count increased
FIFO_15	count' when wr_en is low and rd_en is high it should decreased	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and almostempty,empty and underflow flags	concuurent assertion to check if count decreased
FIFO_16	if rd_en is high and fifo is not empty rd_ptr should increased	randomization under constraints on read enable to be high 30% most of the time	cover values of write enable,read enable and almostempty,empty and underflow flags	concuurent assertion to check if rd_ptr increased
FIFO_17	if wr_en is high and fifo is not full wr_ptr should increased	randomization under constraints on write enable to be high 70% most of the time	cover values of write enable,read enable and wr_ack, overflow ,full and almost full flags	concuurent assertion to check if wr_ptr increased

UVM Structure



1.Top module

It generates the clock signal, Instantiate the interface and fifo design file ,then pass the clk to the interface and pass this interface to the dut ,the it binds the fifo and assertions file for check output functionality and then it sets the interface as a virtual interface into database to make it be able to accessed by another uvm components and run the test to start the testbench.



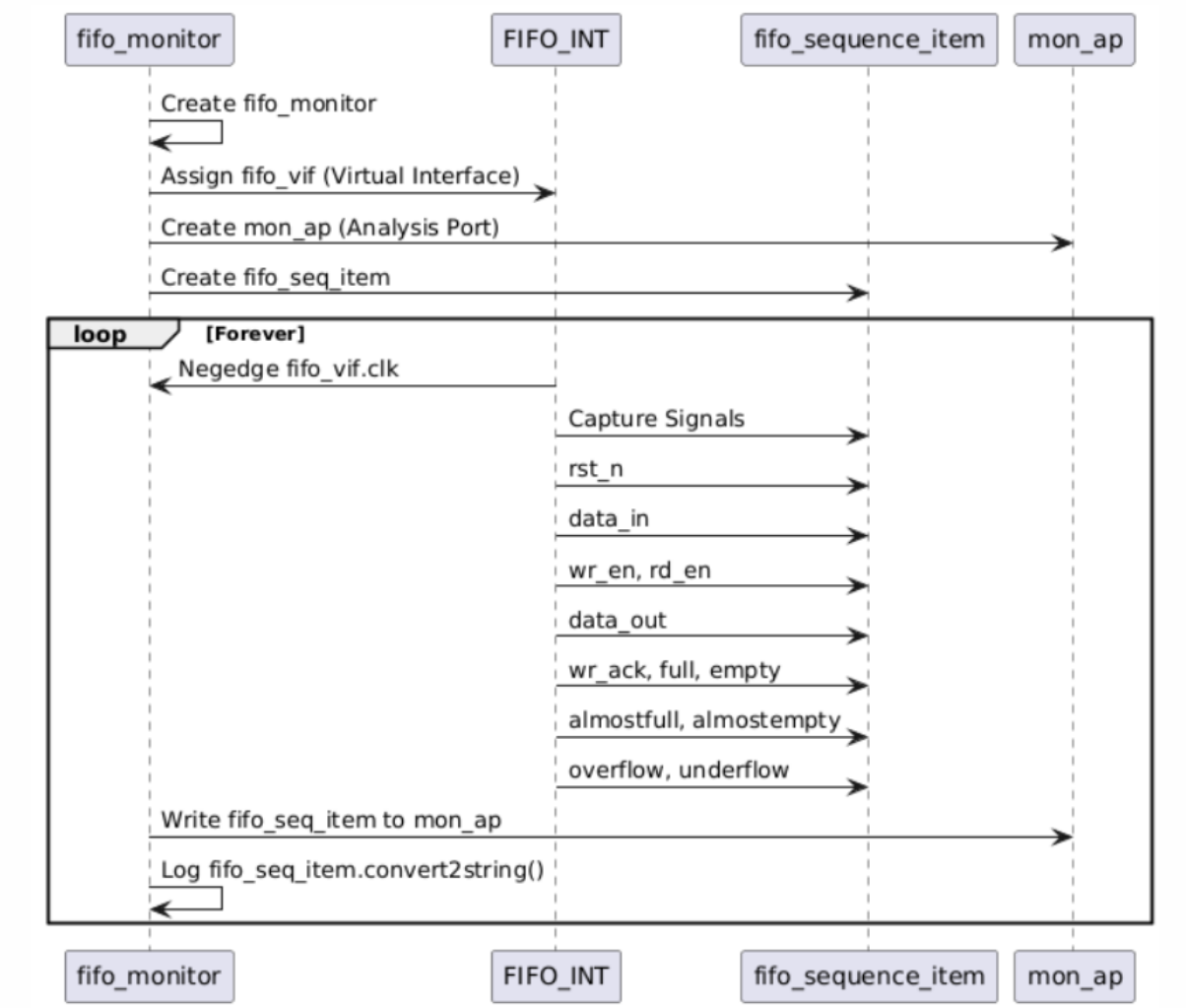
2.Interface

It is the intelligent bundle of wires ,it connect the design to testbench blocks monitor and driver the fifo interface define the signals and has a clock as a separate port as it is generated in top module.

3.Monitoring

a.Fifo monitor:

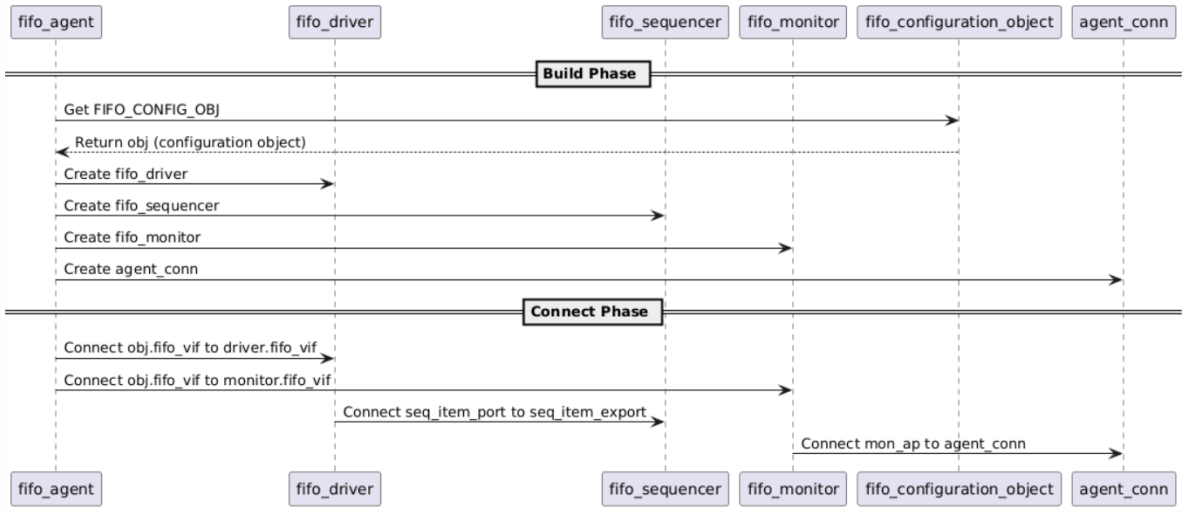
fifo monitor monitors the activity of fifo interface ,and check the functionality/coverage it translates the pin-level signals into sequence items to broadcast them to fifo scoreboard to check the functionality of the design and fifo coverage through analysis port connects analysis exports.

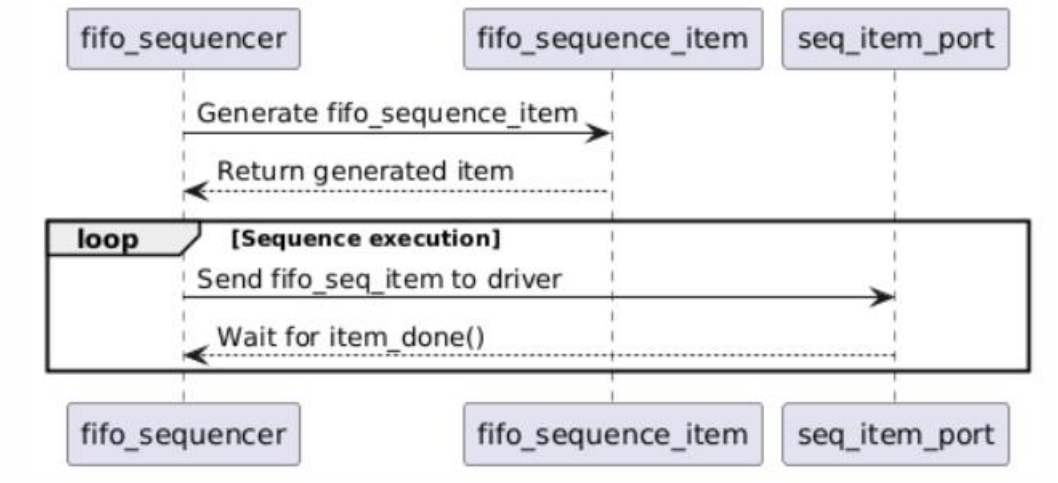


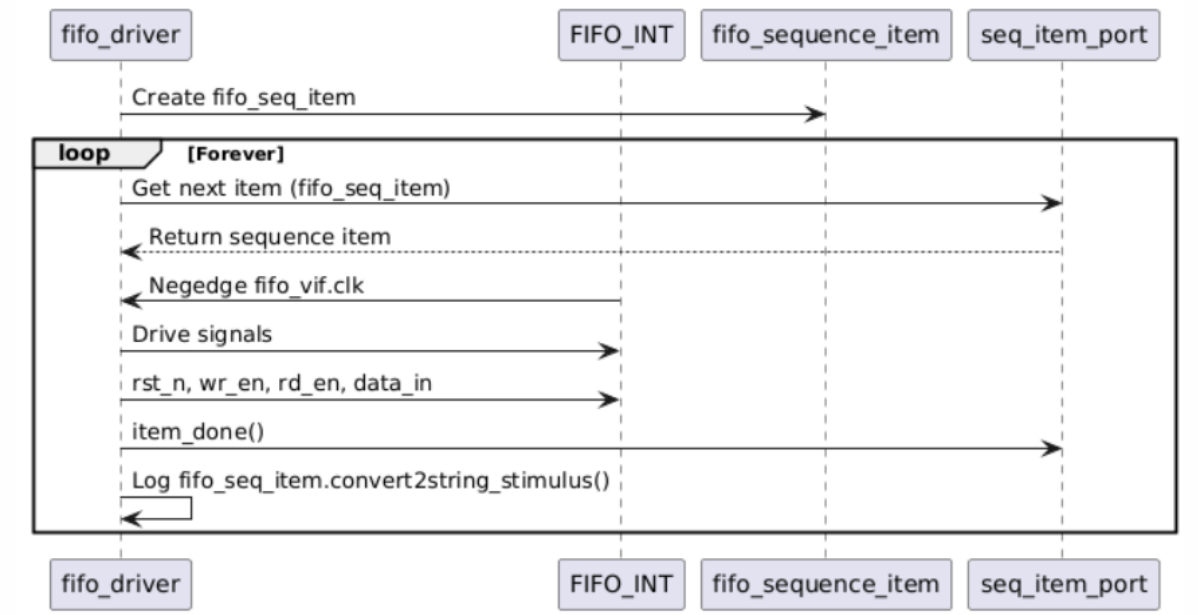
b.Fifo agent:

fifo agent encapsulates the sequencer ,that manages the transfer of sequence items between reset,write only,read only ,read write sequences and the driver it acts as a fifo the sequence push data to sequencer then driver pop from this fifo then the driver drives the interface as a virtual interface by those sequences.

Agent has a configuration object to retrieve the virtual interface handle and then connects the virtual interface of the driver and monitor.


fifo_agent
fifo_driver
fifo_sequencer
fifo_monitor
fifo_configuration_object
agent_conn
Build Phase
Get FIFO_CONFIG_OBJ
Return obj (configuration object)
Create fifo_driver
Create fifo_sequencer
Create fifo_monitor
Create agent_conn
Connect Phase
Connect obj.fifo_vif to driver.fifo_vif
Connect obj.fifo_vif to monitor.fifo_vif
Connect seq_item_port to seq_item_export
Connect mon_ap to agent_conn

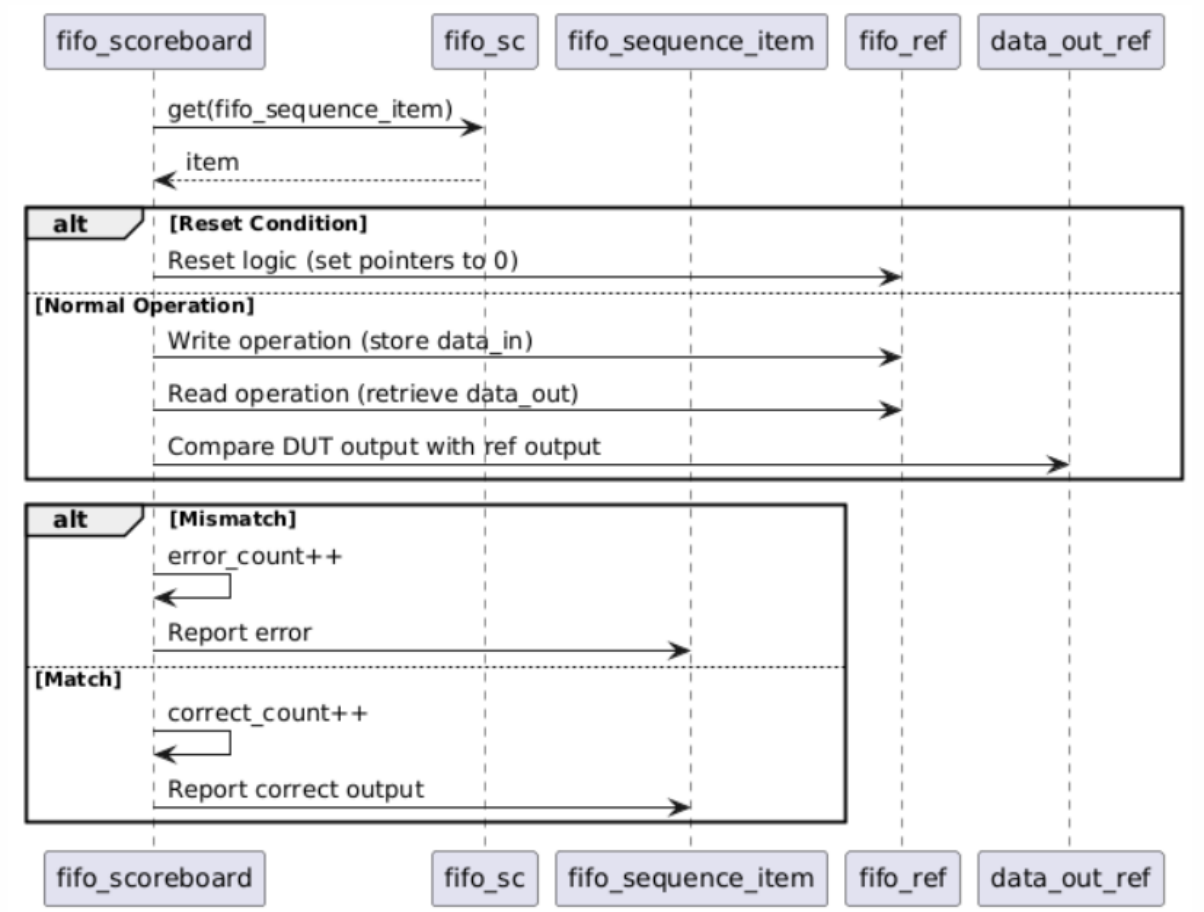

fifo_sequencer
fifo_sequence_item
seq_item_port
Generate fifo_sequence_item
Return generated item
loop
[Sequence execution]
Send fifo_seq_item to driver
Wait for item_done()


fifo_driver
FIFO_INT
fifo_sequence_item
seq_item_port
Create fifo_seq_item
loop
[Forever]
Get next item (fifo_seq_item)
Return sequence item
Negedge fifo_vif.clk
Drive signals
rst_n, wr_en, rd_en, data_in
item_done()
Log fifo_seq_item.convert2string_stimulus()

4.Scoreboard

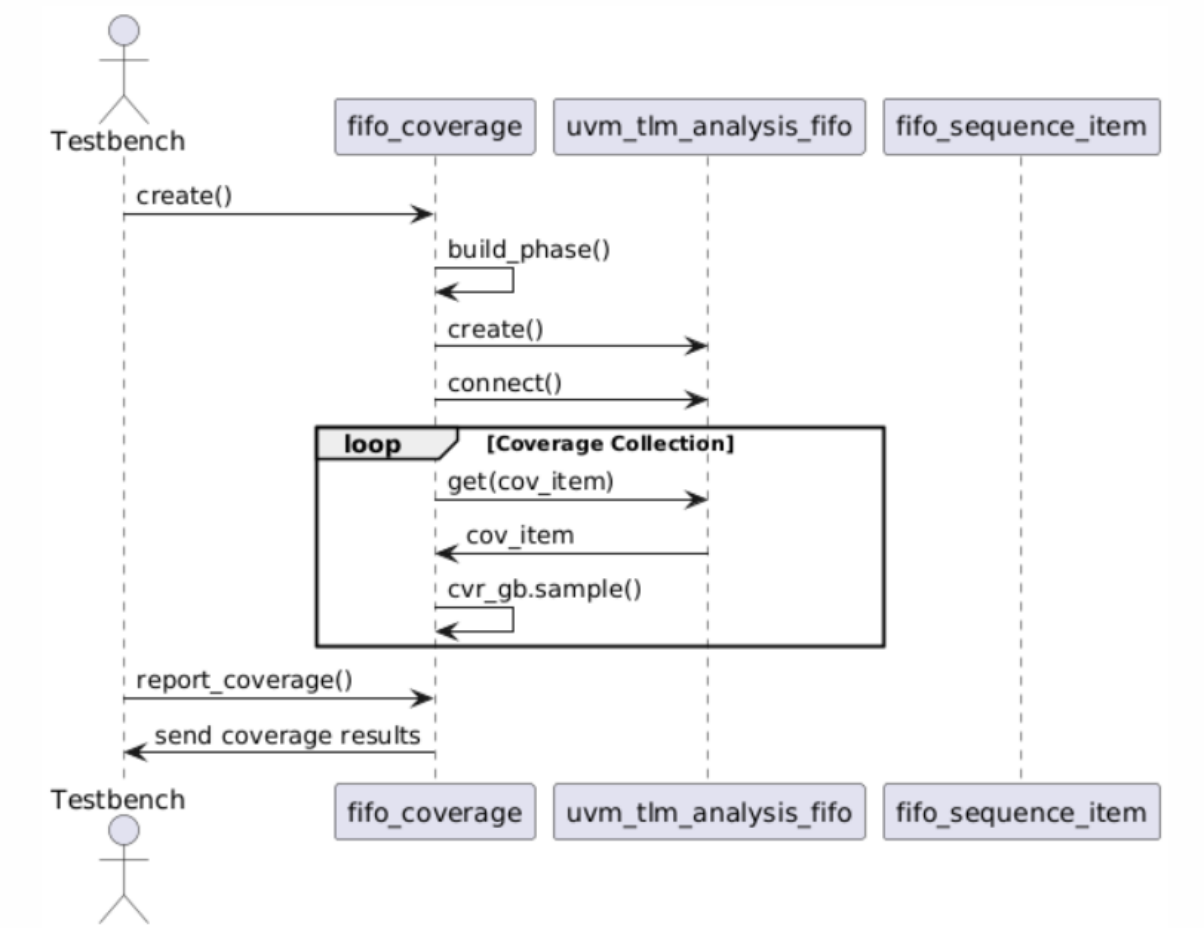
The fifo scoreboard compares the behavior of a FIFO against a reference model. It tracks errors and correctness and provides a mechanism to connect with a monitor through TLM interfaces. The scoreboard verifies that the DUT correctly implements FIFO behavior based on input control signals and compares the design's outputs to the expected results.

The monitor will use the “write” method of the analysis_port to broadcast the sequence item to the scoreboard, the “uvm_tlm_analysis_fifo” receives the sequence item from analysis port through analysis export connected to it.



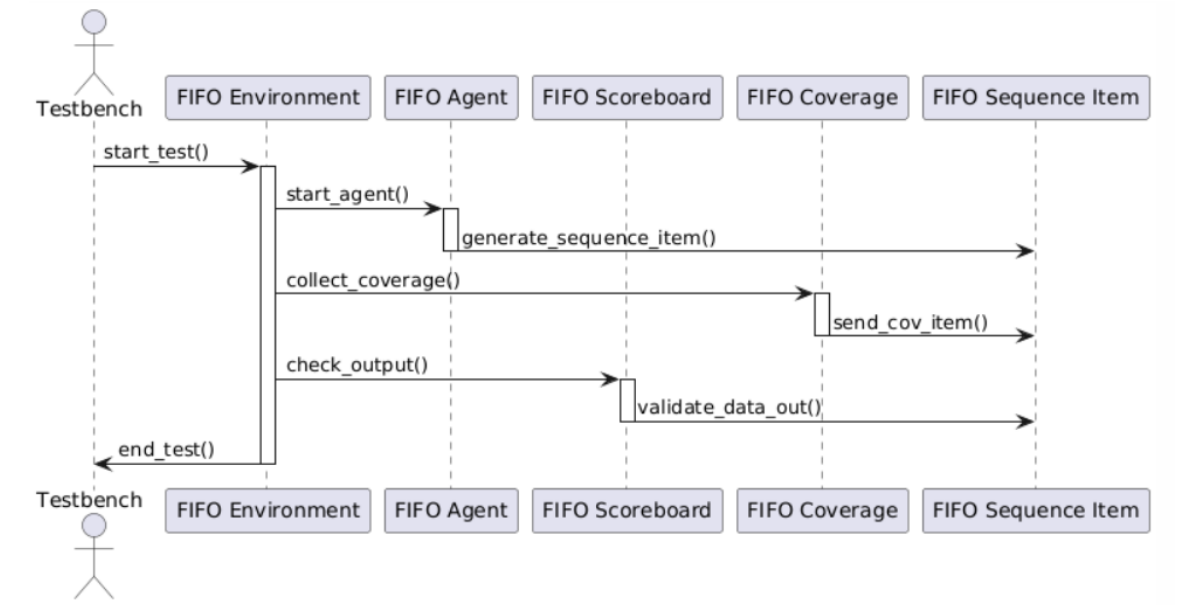
5.Coverage collector

The fifo coverage class capture functional coverage data for a FIFO design. It monitors various control signals and status conditions to assess the behavior of the FIFO during simulation. By defining both individual coverpoints and cross coverage between important signals, this coverage component helps identify how thoroughly the FIFO has been exercised in the testbench.



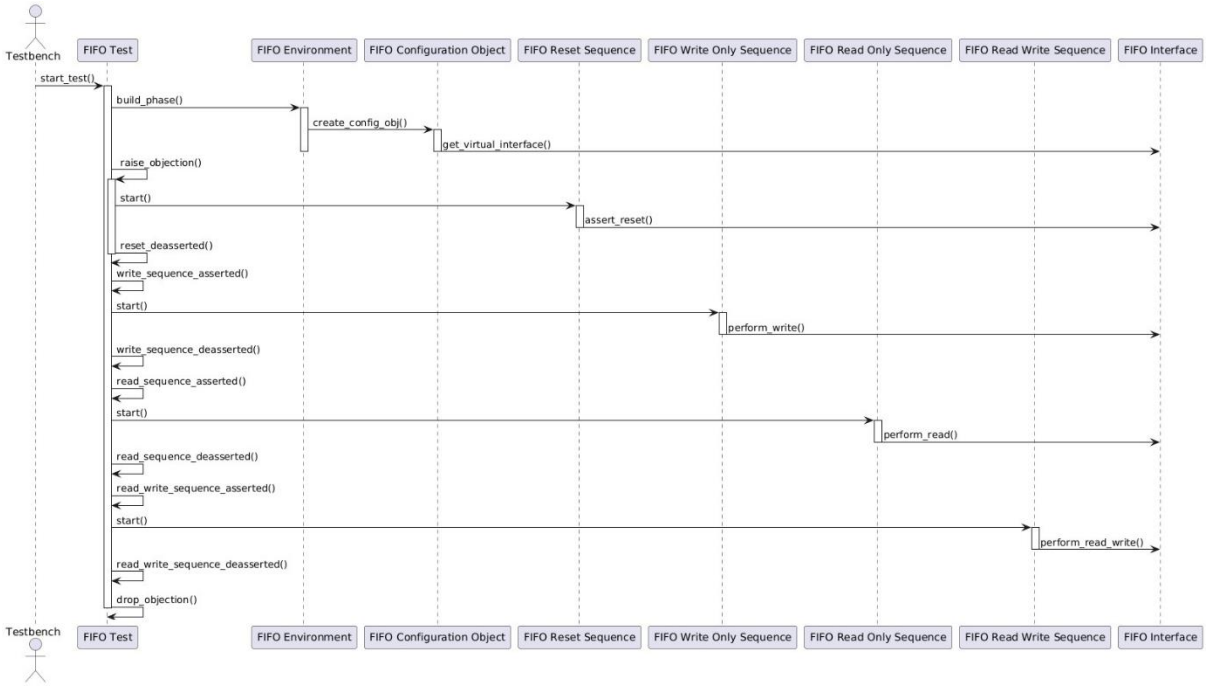
6.Environment

The fifo_environment class instantiates the the agent, scoreboard, and coverage components during the build phase and establishes their connections during the connect phase.The environment is critical in orchestrating the interaction between stimulus generation, monitoring, and verification, thus ensuring comprehensive testing of the FIFO design, it also connects the agent to scoreboard and coverage collector.



7.Test

The fifo_test class imports necessary packages, including those for environment setup, configuration objects, and different sequences for resetting, writing, reading, and reading/writing sequences. In its constructor, it initializes the test name and parent component. During the build_phase, the class creates instances of the FIFO environment and various sequences, while also attempting to retrieve a virtual interface for the FIFO design from the UVM configuration database. If the retrieval fails, it issues a fatal error message. The run_phase starts by raising an objection to prevent simulation termination until all operations are complete. It then sequentially starts the reset sequence, write sequence, read sequence, and finally the read/write sequence, logging information at each step to indicate the sequence status. After completing all sequences, it drops the objection to allow the simulation to proceed.



8.Assertions

The assertions are introduced in a separate file to check the values of fifo flags it is bined with design in the top module, The code includes fatal assertions that trigger errors if any property fails during simulation, ensuring comprehensive verification of the FIFO's behavior.

Code Coverage

```
================================================================================
=== Instance: /top_module/fifo_interface_instance
=== Design Unit: work.FIFO_INT
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

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

Toggle Coverage for instance /top_module/fifo_interface_instance --

                                                 Node      1H->0L      0L->1H  "Coverage"
                                                 ----------------------------------------
                                          almostempty           1           1      100.00
                                           almostfull           1           1      100.00
                                                  clk           1           1      100.00
                                         data_in[15-0]          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
                                                rst_n           1           1      100.00
                                            underflow           1           1      100.00
                                               wr_ack           1           1      100.00
                                                wr_en           1           1      100.00

Total Node Count      =         43
Toggled Node Count    =         43
Untoggled Node Count  =          0

Toggle Coverage       =     100.00% (86 of 86 bins)
```

```
================================================================================
=== Instance: /top_module/DUT
=== Design Unit: work.FIFO
================================================================================
Branch Coverage:
    Enabled Coverage            Bins      Hits    Misses  Coverage
    ----------------            ----      ----    ------  --------
    Branches                      23        23         0   100.00%

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

Branch Coverage for instance /top_module/DUT

    Line        Item                    Count    Source
    ----        ----                    -----    ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    18                                  8120    Count coming in to IF
    18              1                   2508      if (!fifo_interface_instance.rst_n) begin
    23              1                   3825      else if (fifo_interface_instance.wr_en && count < fifo_interface_instance.FIFO_DEPTH) begin
    28              1                   1787      else begin
Branch totals: 3 hits of 3 branches = 100.00%
------------------------------------IF Branch------------------------------------
    30                                  1787    Count coming in to IF
    30              1                    760      if (fifo_interface_instance.full && fifo_interface_instance.wr_en)
    32              1                   1027      else
Branch totals: 2 hits of 2 branches = 100.00%
------------------------------------IF Branch------------------------------------
    38                                  8120    Count coming in to IF
    38              1                   2508      if (!fifo_interface_instance.rst_n) begin
    42              1                    163      else if (fifo_interface_instance.rd_en && count != 0) begin
    46              1                   5449      else begin
Branch totals: 3 hits of 3 branches = 100.00%
------------------------------------IF Branch------------------------------------
    47                                  5449    Count coming in to IF
    47              1                    849      if (fifo_interface_instance.empty && fifo_interface_instance.rd_en)
    49              1                   4600      else
Branch totals: 2 hits of 2 branches = 100.00%
------------------------------------IF Branch------------------------------------
    55                                  6609    Count coming in to IF
    55              1                   2266      if (!fifo_interface_instance.rst_n) begin
    58              1                   4343      else begin
Branch totals: 2 hits of 2 branches = 100.00%
------------------------------------IF Branch------------------------------------
    59                                  4343    Count coming in to IF
    59              1                   3715      if    ( ((((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b10) && (!fifo_interface_instance.full)) || ((((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b11) && (fifo_interface_instance.empty)))
    61              1                     53      else if ((((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b01) && (!fifo_interface_instance.empty)) || ((((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b11) && (fifo_interface_instance.full)))
                                          575    All false Count
Branch totals: 3 hits of 3 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
    66                                    4687     Count coming in to IF
    66              1                      179     assign fifo_interface_instance.full = (count == fifo_interface_instance.FIFO_DEPTH)? 1 : 0;

    66              2                     4508     assign fifo_interface_instance.full = (count == fifo_interface_instance.FIFO_DEPTH)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                    4687     Count coming in to IF
    67              1                      943     assign fifo_interface_instance.empty = (count == 0)? 1 : 0;

    67              2                     3744     assign fifo_interface_instance.empty = (count == 0)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    68                                    4687     Count coming in to IF
    68              1                      232     assign fifo_interface_instance.almostfull = (count == fifo_interface_instance.FIFO_DEPTH-1)? 1 : 0;

    68              2                     4455     assign fifo_interface_instance.almostfull = (count == fifo_interface_instance.FIFO_DEPTH-1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    69                                    4687     Count coming in to IF
    69              1                      952     assign fifo_interface_instance.almostempty = (count == 1)? 1 : 0;

    69              2                     3735     assign fifo_interface_instance.almostempty = (count == 1)? 1 : 0;

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

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      25        25         0   100.00%

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

Statement Coverage for instance /top_module/DUT --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File FIFO.sv
    8                                               module FIFO(FIFO_INT.DUT fifo_interface_instance);

    9

    10                                              localparam max_fifo_addr = $clog2(fifo_interface_instance.FIFO_DEPTH);

    11

    12                                              reg [fifo_interface_instance.FIFO_WIDTH-1:0] mem [fifo_interface_instance.FIFO_DEPTH-1:0];

    13

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

    15                                              reg [max_fifo_addr:0] count;

    16

    17              1                       8120    always @(posedge fifo_interface_instance.clk or negedge fifo_interface_instance.rst_n) begin

    18                                                 if (!fifo_interface_instance.rst_n) begin

    19              1                       2508               wr_ptr <= 0;

    20              1                       2508               fifo_interface_instance.overflow <= 0;

    21              1                       2508               fifo_interface_instance.wr_ack <= 0;

    22                                                 end

    23                                                 else if (fifo_interface_instance.wr_en && count < fifo_interface_instance.FIFO_DEPTH) begin

    24              1                       3825               mem[wr_ptr] <= fifo_interface_instance.data_in;

    25              1                       3825               fifo_interface_instance.wr_ack <= 1;

    26              1                       3825               wr_ptr <= wr_ptr + 1;

    27                                                 end

    28                                                 else begin

    29              1                       1787               fifo_interface_instance.wr_ack <= 0;

    30                                                         if (fifo_interface_instance.full && fifo_interface_instance.wr_en)

    31              1                        760                       fifo_interface_instance.overflow <= 1;

    32                                                         else

    33              1                       1027                       fifo_interface_instance.overflow <= 0;

    34                                                 end

    35                                              end
```

```
36
37          1                8120    always @(posedge fifo_interface_instance.clk or negedge fifo_interface_instance.rst_n) begin
38                                     if (!fifo_interface_instance.rst_n) begin
39          1                2508              rd_ptr <= 0;
40          1                2508              fifo_interface_instance.underflow <= 0;
41                                     end
42                                     else if (fifo_interface_instance.rd_en && count != 0) begin
43          1                 163              fifo_interface_instance.data_out <= mem[rd_ptr];
44          1                 163              rd_ptr <= rd_ptr + 1;
45                                     end
46                                     else begin
47                                              if (fifo_interface_instance.empty && fifo_interface_instance.rd_en)
48          1                 849                       fifo_interface_instance.underflow <= 1;
49                                              else
50          1                4600                      fifo_interface_instance.underflow <= 0;
51                                     end
52                                   end
53
54          1                6609    always @(posedge fifo_interface_instance.clk or negedge fifo_interface_instance.rst_n) begin
55                                     if (!fifo_interface_instance.rst_n) begin
56          1                2266              count <= 0;
57                                     end
58                                     else begin
59                                              if      ( (((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b10) && (!fifo_interface_instance.full)) || (((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b11) && (fifo_interface_instance.empty)))
60          1                3715                      count <= count + 1;
61                                              else if ((((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b01) && (!fifo_interface_instance.empty)) || (((fifo_interface_instance.wr_en, fifo_interface_instance.rd_en) == 2'b11) && (fifo_interface_instance.full)))
62          1                  53                      count <= count - 1;
63                                     end
64                                   end
65
66          1                4688    assign fifo_interface_instance.full = (count == fifo_interface_instance.FIFO_DEPTH)? 1 : 0;
67          1                4688    assign fifo_interface_instance.empty = (count == 0)? 1 : 0;
68          1                4688    assign fifo_interface_instance.almostfull = (count == fifo_interface_instance.FIFO_DEPTH-1)? 1 : 0;
69          1                4688    assign fifo_interface_instance.almostempty = (count == 1)? 1 : 0;
```

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

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

Toggle Coverage for instance /top_module/DUT --

                                            Node      1H->0L      0L->1H  "Coverage"
                                            ----------------------------------------
                                       count[3-0]           1           1      100.00
                                      rd_ptr[2-0]           1           1      100.00
                                      wr_ptr[2-0]           1           1      100.00

Total Node Count     =         10
Toggled Node Count   =         10
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (20 of 20 bins)
```

Functional Coverage

```
================================================================================
=== Instance: /UVM_FIFO_coverage_pkg
=== Design Unit: work.UVM_FIFO_coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses    16        na        na        na
            Covergroup Bins    74        74         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /UVM_FIFO_coverage_pkg/fifo_coverage/cvr_gb     100.00%        100          -    Covered
    covered/total bins:                                   74         74          -
    missing/total bins:                                    0         74          -
    % Hit:                                            100.00%        100          -
    Coverpoint cp_rd_en                               100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      5711          1          -    Covered
        bin auto[1]                                      1290          1          -    Covered
    Coverpoint cp_wr_en                               100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      1317          1          -    Covered
        bin auto[1]                                      5684          1          -    Covered
    Coverpoint cp_wr_ack                              100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      3176          1          -    Covered
        bin auto[1]                                      3825          1          -    Covered
    Coverpoint cp_overflow                            100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      6241          1          -    Covered
        bin auto[1]                                       760          1          -    Covered
    Coverpoint cp_full                                100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      6062          1          -    Covered
        bin auto[1]                                       938          1          -    Covered
    Coverpoint cp_empty                               100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      4728          1          -    Covered
        bin auto[1]                                      2272          1          -    Covered
    Coverpoint cp_almostfull                          100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                      6764          1          -    Covered
        bin auto[1]                                       236          1          -    Covered
```

Coverpoint cp_almostempty	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	5956	1	-	Covered
bin auto[1]	1044	1	-	Covered
Coverpoint cp_underflow	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	6139	1	-	Covered
bin auto[1]	862	1	-	Covered
Cross #cross__0#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	112	1	-	Covered
bin <auto[0],auto[1],auto[1]>	3541	1	-	Covered
bin <auto[1],auto[0],auto[1]>	48	1	-	Covered
bin <auto[0],auto[0],auto[1]>	124	1	-	Covered
bin <auto[1],auto[1],auto[0]>	92	1	-	Covered
bin <auto[0],auto[1],auto[0]>	1939	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1038	1	-	Covered
bin <auto[0],auto[0],auto[0]>	107	1	-	Covered
Cross #cross__1#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	1	1	-	Covered
bin <auto[0],auto[1],auto[1]>	756	1	-	Covered
bin <auto[1],auto[0],auto[1]>	1	1	-	Covered
bin <auto[0],auto[0],auto[1]>	2	1	-	Covered
bin <auto[1],auto[1],auto[0]>	203	1	-	Covered
bin <auto[0],auto[1],auto[0]>	4724	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1085	1	-	Covered
bin <auto[0],auto[0],auto[0]>	229	1	-	Covered
Cross #cross__2#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3	1	-	Covered
bin <auto[0],auto[1],auto[1]>	933	1	-	Covered
bin <auto[1],auto[0],auto[1]>	1	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1	1	-	Covered
bin <auto[1],auto[1],auto[0]>	201	1	-	Covered
bin <auto[0],auto[1],auto[0]>	4547	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1085	1	-	Covered
bin <auto[0],auto[0],auto[0]>	229	1	-	Covered

bin <auto[0],auto[0],auto[0]>	229	1		Covered
Cross #cross__3#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	61	1	-	Covered
bin <auto[0],auto[1],auto[1]>	1119	1	-	Covered
bin <auto[1],auto[0],auto[1]>	1020	1	-	Covered
bin <auto[0],auto[0],auto[1]>	72	1	-	Covered
bin <auto[1],auto[1],auto[0]>	143	1	-	Covered
bin <auto[0],auto[1],auto[0]>	4361	1	-	Covered
bin <auto[1],auto[0],auto[0]>	66	1	-	Covered
bin <auto[0],auto[0],auto[0]>	158	1	-	Covered
Cross #cross__4#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3	1	-	Covered
bin <auto[0],auto[1],auto[1]>	224	1	-	Covered
bin <auto[1],auto[0],auto[1]>	3	1	-	Covered
bin <auto[0],auto[0],auto[1]>	6	1	-	Covered
bin <auto[1],auto[1],auto[0]>	201	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5256	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1083	1	-	Covered
bin <auto[0],auto[0],auto[0]>	224	1	-	Covered
Cross #cross__5#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	61	1	-	Covered
bin <auto[0],auto[1],auto[1]>	899	1	-	Covered
bin <auto[1],auto[0],auto[1]>	29	1	-	Covered
bin <auto[0],auto[0],auto[1]>	55	1	-	Covered
bin <auto[1],auto[1],auto[0]>	143	1	-	Covered
bin <auto[0],auto[1],auto[0]>	4581	1	-	Covered
bin <auto[1],auto[0],auto[0]>	1057	1	-	Covered
bin <auto[0],auto[0],auto[0]>	175	1	-	Covered
Cross #cross__6#	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	12	1	-	Covered
bin <auto[0],auto[1],auto[1]>	32	1	-	Covered
bin <auto[1],auto[0],auto[1]>	790	1	-	Covered
bin <auto[0],auto[0],auto[1]>	28	1	-	Covered
bin <auto[1],auto[1],auto[0]>	192	1	-	Covered
bin <auto[0],auto[1],auto[0]>	5448	1	-	Covered
bin <auto[1],auto[0],auto[0]>	296	1	-	Covered
bin <auto[0],auto[0],auto[0]>	203	1	-	Covered

Sequential Domain Coverage

```
=== Instance: /top/DUT/FIFO_SVA
=== Design Unit: work.SVA
================================================================================

Assertion Coverage:
    Assertions                      17        17         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUT/FIFO_SVA/a_reset
                     UVM_FIFO_assertions.sv(59)
                                                        0          1
/top/DUT/FIFO_SVA/b_reset
                     UVM_FIFO_assertions.sv(60)
                                                        0          1
/top/DUT/FIFO_SVA/c_reset
                     UVM_FIFO_assertions.sv(61)
                                                        0          1
/top/DUT/FIFO_SVA/d_reset
                     UVM_FIFO_assertions.sv(62)
                                                        0          1
/top/DUT/FIFO_SVA/e_reset
                     UVM_FIFO_assertions.sv(63)
                                                        0          1
/top/DUT/FIFO_SVA/f_reset
                     UVM_FIFO_assertions.sv(64)
                                                        0          1
/top/DUT/FIFO_SVA/ast1
                     UVM_FIFO_assertions.sv(68)
                                                        0          1
/top/DUT/FIFO_SVA/ast2
                     UVM_FIFO_assertions.sv(70)
                                                        0          1
/top/DUT/FIFO_SVA/ast3
                     UVM_FIFO_assertions.sv(72)
                                                        0          1
/top/DUT/FIFO_SVA/ast4
                     UVM_FIFO_assertions.sv(74)
                                                        0          1
/top/DUT/FIFO_SVA/ast5
                     UVM_FIFO_assertions.sv(76)
                                                        0          1
/top/DUT/FIFO_SVA/ast6
                     UVM_FIFO_assertions.sv(78)
                                                        0          1
/top/DUT/FIFO_SVA/ast7
                     UVM_FIFO_assertions.sv(80)
                                                        0          1
/top/DUT/FIFO_SVA/ast8
                     UVM_FIFO_assertions.sv(82)
                                                        0          1
/top/DUT/FIFO_SVA/ast9
                     UVM_FIFO_assertions.sv(84)
                                                        0          1
/top/DUT/FIFO_SVA/ast10
                     UVM_FIFO_assertions.sv(86)
                                                        0          1
/top/DUT/FIFO_SVA/ast11
                     UVM_FIFO_assertions.sv(88)
                                                        0          1
```

Bug Report

a.

1-Description:

Almostfull occurs when count equals to fifo depth – 1

2-Before:

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

3-After:

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

b.

1-Description:

Underflow flag is a sequential signal so it should be triggered with posedge of clk in always block not continuous assign

2-Before:

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

3-After:

```
if (empty & rd_en)
    underflow <= 1;
else
    underflow <= 0;
```

c.

1-Description:

Registers of the design should be low when reset is asserted

2-Before:

```
if (!fifo_interface_instance.rst_n) begin
    wr_ptr <= 0;
```

3-After:

```
if (!fifo_interface_instance.rst_n) begin
    wr_ptr <= 0;
    fifo_interface_instance.overflow <= 0;
    fifo_interface_instance.wr_ack <= 0;
end
```

d.

1-Description:

Underflow is a sequential signal should be low when reset is asserted

2-Before:

```
if (!fifo_interface_instance.rst_n) begin
    rd_ptr <= 0;
```

3-After:

```
if (!fifo_interface_instance.rst_n) begin
    rd_ptr <= 0;
    fifo_interface_instance.underflow <= 0;
```

e.

1-Description:

Count is increased or decreased at some extreme cases when rd_en and wr_en are both high

2-Before:

```
else begin
    if  ( (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b10) && (!fifo_interface_instance.full)))
        count <= count + 1;
    else if ((({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b01) && (!fifo_interface_instance.empty)))
        count <= count - 1;
end
```

3-After:

```
e_instance.full)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.empty)))

face_instance.empty)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.full)))
```

f.

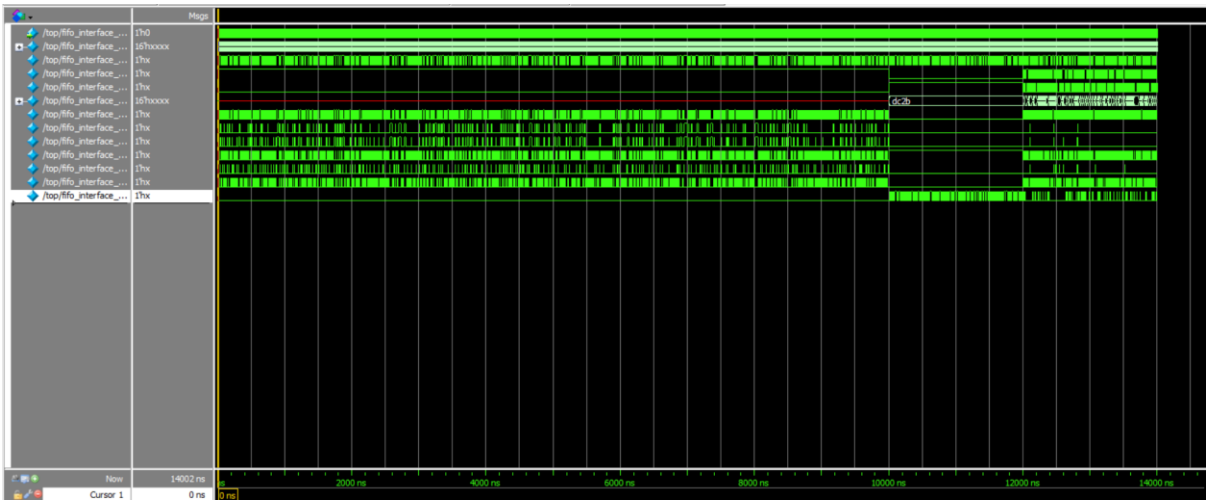
1-Description:

& will work correctly in this case because the signals are 1-bit, it is better to use && for clarity, readability, and to align with common design practices.

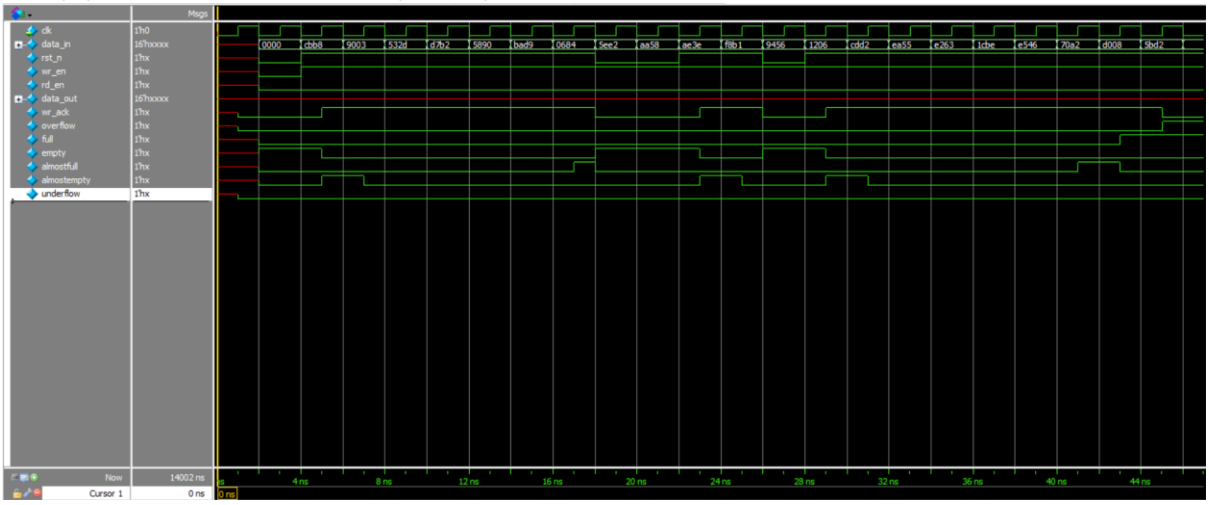
```
if (fifo_interface_instance.full && fifo_interface_instance.wr_en)
    fifo_interface_instance.overflow <= 1;
else
    fifo_interface_instance.overflow <= 0;
```

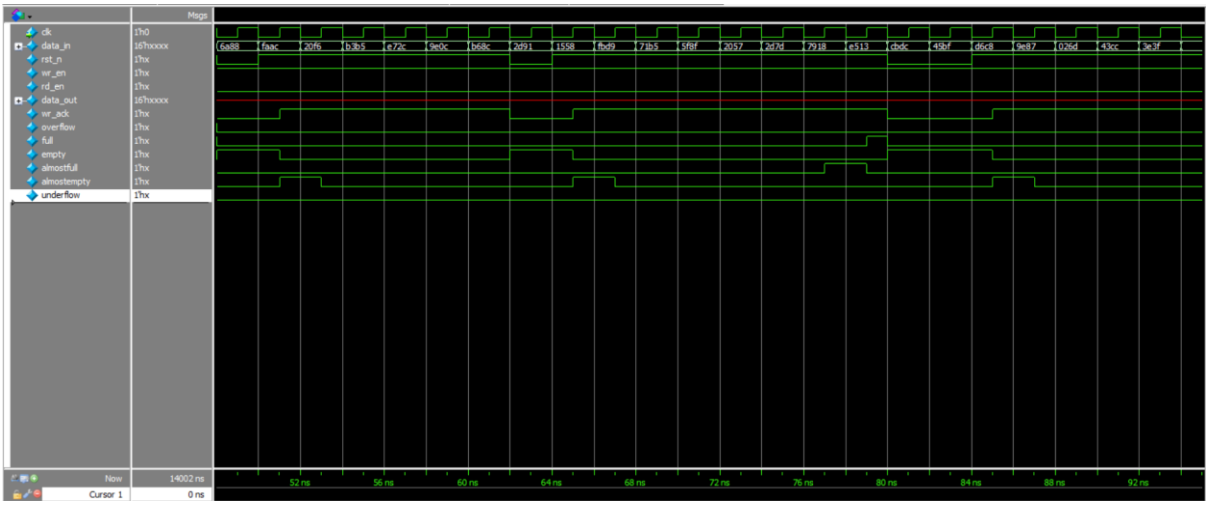
Questasim Snippets

```
# 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(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test fifo_test...
# UVM_INFO UVM_FIFO_test.sv(40) @ 0: uvm_test_top [run phase] Reset asserted
# UVM_INFO UVM_FIFO_test.sv(42) @ 2: uvm_test_top [run phase] Reset deasserted
# UVM_INFO UVM_FIFO_test.sv(43) @ 2: uvm_test_top [run phase] write sequence asserted
# UVM_INFO UVM_FIFO_test.sv(45) @ 10002: uvm_test_top [run phase] write sequence deasserted
# UVM_INFO UVM_FIFO_test.sv(46) @ 10002: uvm_test_top [run phase] read sequence asserted
# UVM_INFO UVM_FIFO_test.sv(48) @ 12002: uvm_test_top [run phase] read sequence deasserted
# UVM_INFO UVM_FIFO_test.sv(49) @ 12002: uvm_test_top [run phase] read write sequence asserted
# UVM_INFO UVM_FIFO_test.sv(51) @ 14002: uvm_test_top [run phase] read sequence deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 14002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO UVM_FIFO_scoreboard.sv(57) @ 14002: uvm_test_top.env.fifo_sc [report_phase] correct cases :       7001
# UVM_INFO UVM_FIFO_scoreboard.sv(58) @ 14002: uvm_test_top.env.fifo_sc [report_phase] error cases :          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]     2
# [run phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 14002 ns  Iteration: 61  Instance: /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
```

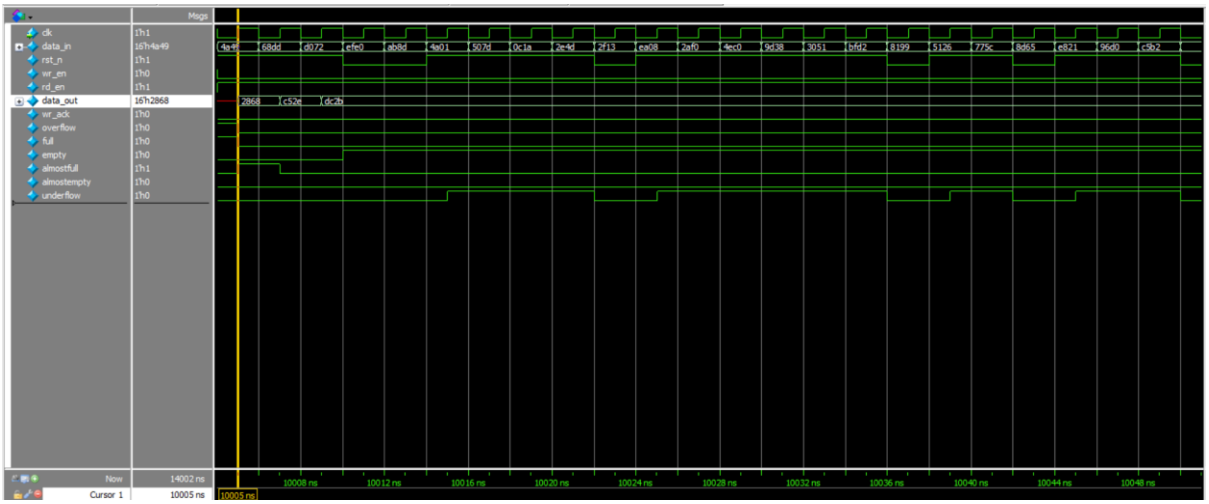


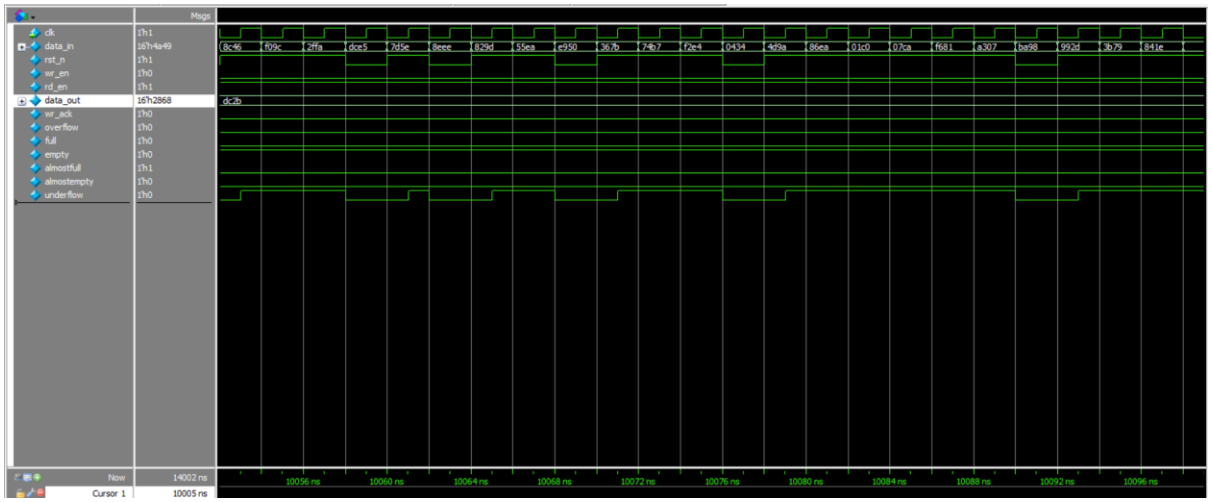
Snippets of write only sequence wave form:



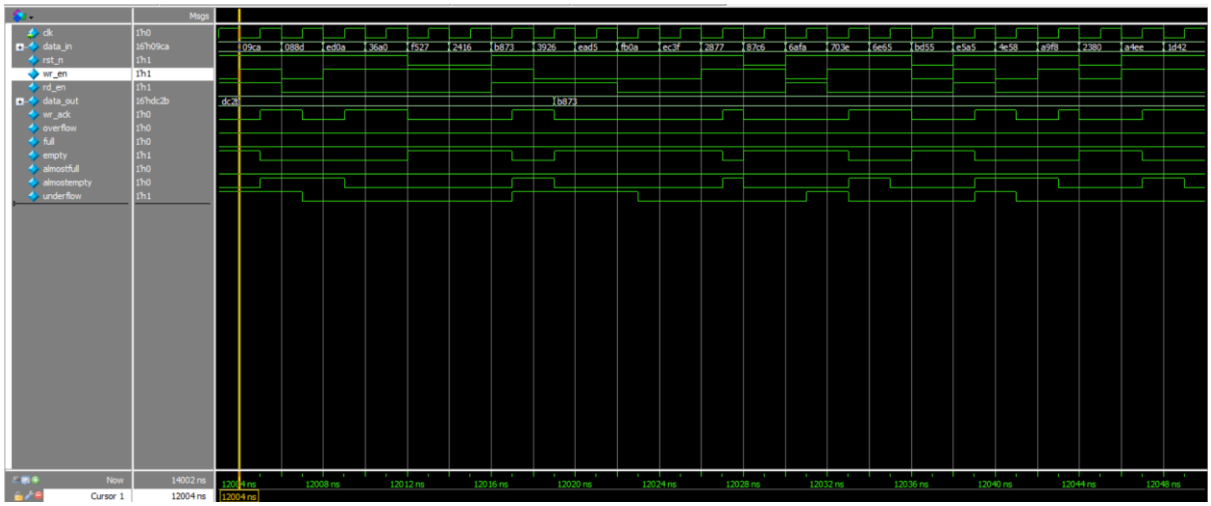


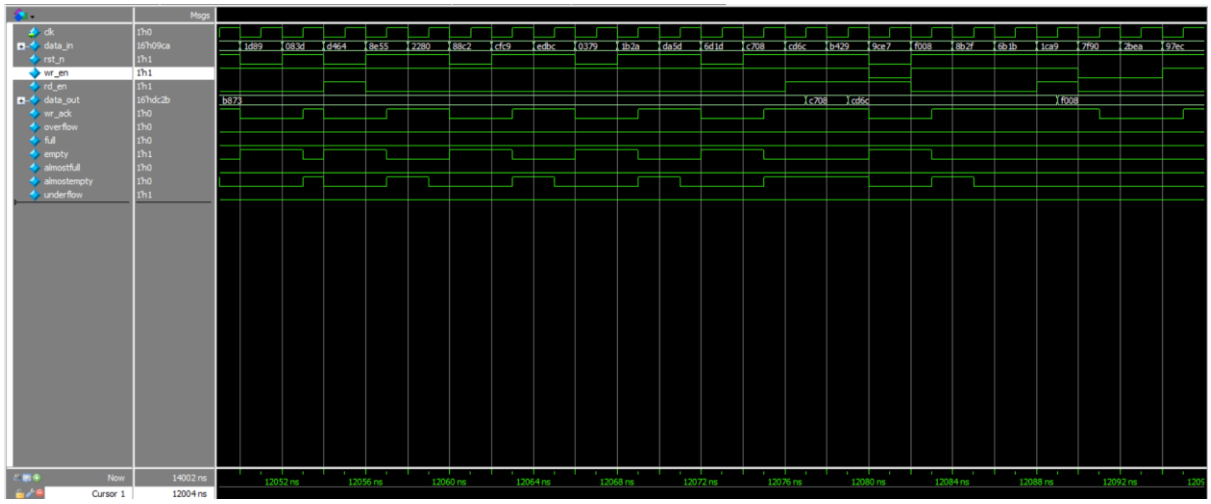
Snippets of read only sequence wave form:





Snippets of read write sequence wave form:





Assertions

Feature	Assertion
'wr_ack' should be asserted when write occurs and FIFO is not full	`@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (fifo_interface_instance.wr_en && !fifo_interface_instance.full) |=> fifo_interface_instance.wr_ack`
'full' flag should be asserted when fifo_count = depth	`@(posedge fifo_interface_instance.clk) (DUT.count == fifo_interface_instance.FIFO_DEPTH) |-> fifo_interface_instance.full == 1`
'empty' flag should be asserted when fifo_count = 0	`@(posedge fifo_interface_instance.clk) (DUT.count == 0) |-> fifo_interface_instance.empty == 1`
'almostfull' flag should be asserted when fifo_count = depth-1	`@(posedge fifo_interface_instance.clk) (DUT.count == fifo_interface_instance.FIFO_DEPTH-1) |-> fifo_interface_instance.almostfull`
'almostempty' flag should be asserted when fifo_count = 1	`@(posedge fifo_interface_instance.clk) (DUT.count == 1) |-> fifo_interface_instance.almostempty`
'overflow' flag if full and wr_en it should be assertd	`@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (fifo_interface_instance.full && fifo_interface_instance.wr_en) |=> fifo_interface_instance.overflow`
'underflow' flag if empty and rd_en it should be assertd	`@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (fifo_interface_instance.empty && fifo_interface_instance.rd_en ) |=> fifo_interface_instance.underflow`
'count' when wr_en is high and rd_en is low it should increased or both are high anf fifo is empty	`@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) ((({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b10) && (!fifo_interface_instance.full)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.empty))) |=> DUT.count == $past(DUT.count) + 4'b0001`
'count' when wr_en is low and rd_en is high it should decreased or both are high and fifo is full	`@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n)( (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b01) &&`

	(!fifo_interface_instance.empty)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.full))) |=> DUT.count == $past(DUT.count) - 4'b0001
'rd_ptr' should increased when rd_en is asserted and fifo is not empty	@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (fifo_interface_instance.rd_en && !fifo_interface_instance.empty) |=> DUT.rd_ptr == $past(DUT.rd_ptr)+3'b001
'wr_ptr' should increased when wr_en is asserted and fifo is not full	@(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (fifo_interface_instance.wr_en && !fifo_interface_instance.full) |=> DUT.wr_ptr == $past(DUT.wr_ptr)+3'b001

When reset_n is asserted count,overflow,underflow,rd_ptr,wr_ptr and wr_Ack should be low	always_comb begin
	if(!fifo_interface_instance.rst_n) begin
	a_reset:
	assert final(DUT.count == 0);
	b_reset:
	assert final(fifo_interface_instance.overflow == 0);
	c_reset:
	assert final(fifo_interface_instance.underflow == 0);
	d_reset:
	assert final(DUT.wr_ptr == 0);
	e_reset:
	assert final(DUT.rd_ptr == 0);
	f_reset:
	assert final(fifo_interface_instance.wr_ack == 0);
	end
	end

Code files

1-DESIGN CODE

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(FIFO_INT.DUT fifo_interface_instance);

localparam max_fifo_addr = $clog2(fifo_interface_instance.FIFO_DEPTH);

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

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

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

always @(posedge fifo_interface_instance.clk or negedge fifo_interface_instance.rst_n) begin
    if (!fifo_interface_instance.rst_n) begin
        rd_ptr <= 0;
        fifo_interface_instance.underflow <= 0;
    end
    else if (fifo_interface_instance.rd_en && count != 0) begin
        fifo_interface_instance.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
```

```
    end
    else begin
        if (fifo_interface_instance.empty && fifo_interface_instance.rd_en)
            fifo_interface_instance.underflow <= 1;
        else
            fifo_interface_instance.underflow <= 0;
    end
end

always @(posedge fifo_interface_instance.clk or negedge fifo_interface_instance.rst_n) begin
    if (!fifo_interface_instance.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b10) &&
(!fifo_interface_instance.full)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en}
== 2'b11) && (fifo_interface_instance.empty)))
            count <= count + 1;
        else if ((({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b01) &&
(!fifo_interface_instance.empty)) || (({fifo_interface_instance.wr_en,
fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.full)))
            count <= count - 1;
    end
end

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

endmodule
```

2-INTERFACE

```
interface FIFO_INT(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;

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

```
endinterface
```

3-SEQUENCE ITEM

```
package sequence_item_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class fifo_sequence_item extends uvm_sequence_item;
    `uvm_object_utils(fifo_sequence_item)
    parameter WIDTH_F = 16;
    parameter DEPTH_F = 8;
    parameter RD_EN_ON_DIST = 30;
    parameter WR_EN_ON_DIST = 70;
    rand logic [WIDTH_F-1:0] data_in;
    rand logic rst_n, wr_en, rd_en;
    logic [WIDTH_F-1:0] data_out;
    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;
    function new(string name = "fifo_sequence_item");
        super.new(name);
    endfunction

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

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

    constraint reset_n {
        rst_n dist {
            0 := 20,
            1 := 80
        };
    }

    constraint write_enable {
        wr_en dist {
            0 := 100 - WR_EN_ON_DIST,
            1 := WR_EN_ON_DIST
```

```
        };
    }

    constraint read_enable {
        rd_en dist {
            0 := 100 - RD_EN_ON_DIST,
            1 := RD_EN_ON_DIST
        };
    }
endclass
endpackage
```

4-CONFIGURATION OBJECT

```
package configuration_object_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_configuration_object extends uvm_object;
        `uvm_object_utils(fifo_configuration_object)
        virtual FIFO_INT fifo_vif;
        function new(string name = "fifo_configuration_object");
            super.new(name);
        endfunction
    endclass
endpackage
```

5-DRIVER

```
package driver_pkg;
    import configuration_object_pkg::*;
    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_driver extends uvm_driver #(fifo_sequence_item);
        `uvm_component_utils(fifo_driver);
        virtual FIFO_INT fifo_vif;
        fifo_sequence_item fifo_seq_item;

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_seq_item = fifo_sequence_item::type_id::create("fifo_seq_item");
                seq_item_port.get_next_item(fifo_seq_item);
```

```
                @(negedge fifo_vif.clk);
                fifo_vif.rst_n   = fifo_seq_item.rst_n;
                fifo_vif.wr_en   = fifo_seq_item.wr_en;
                fifo_vif.rd_en   = fifo_seq_item.rd_en;
                fifo_vif.data_in = fifo_seq_item.data_in;
                seq_item_port.item_done();
                `uvm_info("run_phase",fifo_seq_item.convert2string_stimulus(),UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

6-MONITOR

```
package monitor_pkg;
    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    class fifo_monitor extends uvm_monitor;
        `uvm_component_utils(fifo_monitor)
        virtual FIFO_INT fifo_vif;
        fifo_sequence_item fifo_seq_item;
        uvm_analysis_port #(fifo_sequence_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_seq_item = fifo_sequence_item::type_id::create("fifo_seq_item");
                @(negedge fifo_vif.clk);
                fifo_seq_item.rst_n       = fifo_vif.rst_n;
                fifo_seq_item.data_in     = fifo_vif.data_in;
                fifo_seq_item.wr_en       = fifo_vif.wr_en;
                fifo_seq_item.rd_en       = fifo_vif.rd_en;
                fifo_seq_item.data_out    = fifo_vif.data_out;
                fifo_seq_item.wr_ack      = fifo_vif.wr_ack;
                fifo_seq_item.full        = fifo_vif.full;
                fifo_seq_item.empty       = fifo_vif.empty;
                fifo_seq_item.almostfull  = fifo_vif.almostfull;
                fifo_seq_item.almostempty = fifo_vif.almostempty;
```

```
                fifo_seq_item.overflow      = fifo_vif.overflow;
                fifo_seq_item.underflow     = fifo_vif.underflow;
                mon_ap.write(fifo_seq_item);
                `uvm_info("run_phase", fifo_seq_item.convert2string(),UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

7-SCOREBOARD

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

    class fifo_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(fifo_scoreboard)
        parameter WIDTH = 16;
        parameter DEPTH = 8;
        static int error_count = 0;
        static int correct_count  = 0;
        logic [WIDTH-1:0] data_out_ref;
        parameter max_fifo_addr_ref = $clog2(DEPTH);
        uvm_analysis_export #(fifo_sequence_item) sc_export;
        uvm_tlm_analysis_fifo #(fifo_sequence_item) fifo_sc;
        fifo_sequence_item item;
        logic [WIDTH-1:0] fifo_ref [DEPTH-1:0]; //  memory for the FIFO reference
        bit [max_fifo_addr_ref : 0] fifo_count_ref;    // Reference count of items in the FIFO
        bit [max_fifo_addr_ref-1:0] write_pointer_ref; // Reference write pointer
        bit [max_fifo_addr_ref-1:0] read_pointer_ref;  // Reference read pointer
         function new(string name = "fifo_scoreboard",uvm_component parent = null);
            super.new(name,parent);
            write_pointer_ref= 0;
            read_pointer_ref = 0;
            fifo_count_ref   = 0;
        endfunction

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

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            sc_export.connect(fifo_sc.analysis_export);
        endfunction
```

```
        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_sc.get(item);
                ref_model(item);
                if(item.data_out != data_out_ref) begin
                    `uvm_error("run_phase",$sformatf("error encoutered expected out = 0b%b , design
stimulus and output : %s",data_out_ref,item.convert2string()))
                    error_count++;
                end
                else begin
                    `uvm_info("run_phase",$sformatf("correct output :
%s",item.convert2string()),UVM_HIGH)
                    correct_count++;
                end
            end
        endtask

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

        function void ref_model(fifo_sequence_item item);
            if(!item.rst_n) begin
                write_pointer_ref<= 0;
                read_pointer_ref <= 0;
                fifo_count_ref   <= 0;
             end
             else begin
                fork
                   begin
                      // Write operation logic
                      if (item.wr_en && !item.full) begin
                         fifo_ref[write_pointer_ref] <= item.data_in;
                         write_pointer_ref <= write_pointer_ref + 1;
                      end
                   end
                   begin
                      // Read operation logic
                      if (item.rd_en && !item.empty) begin
                         data_out_ref <= fifo_ref[read_pointer_ref];
                         read_pointer_ref <= read_pointer_ref + 1;
                      end
                   end
                   begin
```

```
                    // Counter operation logic
                    if (({item.wr_en, item.rd_en} == 2'b10 && !item.full) ||
                        ({item.wr_en, item.rd_en} == 2'b11 && item.empty))
                       fifo_count_ref <= fifo_count_ref + 1;
                    else if (({item.wr_en, item.rd_en} == 2'b01 && !item.empty) ||
                             ({item.wr_en, item.rd_en} == 2'b11 && item.full))
                       fifo_count_ref <= fifo_count_ref - 1;
                end
            join
        end
    endfunction
endclass
endpackage
```

8-COVERAGE

```
package coverage_pkg;

    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_coverage extends uvm_component;
        `uvm_component_utils(fifo_coverage)
        uvm_analysis_export #(fifo_sequence_item) cov_export;
        uvm_tlm_analysis_fifo #(fifo_sequence_item) fifo_cov;
        fifo_sequence_item cov_item;

        covergroup cvr_gb;
            cp_rd_en:       coverpoint cov_item.rd_en;
            cp_wr_en:       coverpoint cov_item.wr_en;
            cp_wr_ack:      coverpoint cov_item.wr_ack;
            cp_overflow:    coverpoint cov_item.overflow;
            cp_full:        coverpoint cov_item.full;
            cp_empty:       coverpoint cov_item.empty;
            cp_almostfull:  coverpoint cov_item.almostfull;
            cp_almostempty: coverpoint cov_item.almostempty;
            cp_underflow:   coverpoint cov_item.underflow;

            cross cp_rd_en, cp_wr_en, cp_wr_ack;
            cross cp_rd_en, cp_wr_en, cp_overflow;
            cross cp_rd_en, cp_wr_en, cp_full;
            cross cp_rd_en, cp_wr_en, cp_empty;
            cross cp_rd_en, cp_wr_en, cp_almostfull;
            cross cp_rd_en, cp_wr_en, cp_almostempty;
            cross cp_rd_en, cp_wr_en, cp_underflow;
      endgroup
```

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

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

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

        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_cov.get(cov_item);
                cvr_gb.sample();
            end
        endtask

    endclass
endpackage
```

9-SEQUENCER

```
package sequencer_pkg;
    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_sequencer extends uvm_sequencer #(fifo_sequence_item);
        `uvm_component_utils(fifo_sequencer);
        function new(string name = "fifo_sequencer",uvm_component parent = null);
            super.new(name,parent);
        endfunction
    endclass
endpackage
```

10-MAIN SEQUENCE

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

```
    class fifo_read_only_sequence extends uvm_sequence #(fifo_sequence_item);
        `uvm_object_utils(fifo_read_only_sequence)
        fifo_sequence_item seq_item;

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

        task body;
            repeat(1000) begin
                seq_item = fifo_sequence_item::type_id::create("seq_item");
                start_item(seq_item);
                assert(seq_item.randomize() with {wr_en == 0; rd_en == 1;});
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

```
package write_only_sequence_pkg;
import sequence_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class fifo_write_only_sequence extends uvm_sequence #(fifo_sequence_item);
    `uvm_object_utils(fifo_write_only_sequence)
    fifo_sequence_item seq_item;

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

    task body;
        repeat(5000) begin
            seq_item = fifo_sequence_item::type_id::create("seq_item");
            start_item(seq_item);
            assert(seq_item.randomize() with {wr_en == 1; rd_en == 0;});
            finish_item(seq_item);
        end
    endtask
endclass
endpackage
```

```
package read_write_sequence_pkg;
    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_read_write_sequence extends uvm_sequence #(fifo_sequence_item);
        `uvm_object_utils(fifo_read_write_sequence)
        fifo_sequence_item seq_item;
```

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

        task body;
            repeat(1000) begin
                seq_item = fifo_sequence_item::type_id::create("seq_item");
                start_item(seq_item);
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

11-RESET SEQUENCE

```
package reset_sequence_pkg;
    import sequence_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_reset_sequence extends uvm_sequence #(fifo_sequence_item);
        `uvm_object_utils(fifo_reset_sequence)
        fifo_sequence_item seq_item;

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

        task body;
                seq_item = fifo_sequence_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n   = 0;
                seq_item.data_in = 0;
                seq_item.rd_en   = 0;
                seq_item.wr_en   = 0;
                finish_item(seq_item);
        endtask
    endclass
endpackage
```

12-AGENT

```
package agent_pkg;
    import configuration_object_pkg::*;
    import driver_pkg::*;
    import monitor_pkg::*;
```

```
    import sequence_item_pkg::*;
    import reset_sequence_pkg::*;
    import read_only_sequence_pkg::*;
    import write_only_sequence_pkg::*;
    import read_write_sequence_pkg::*;
    import sequencer_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_agent extends uvm_agent;
        `uvm_component_utils(fifo_agent);
        fifo_driver drv;
        fifo_sequencer sqr;
        fifo_monitor mtr;
        fifo_configuration_object obj;
        uvm_analysis_port #(fifo_sequence_item) agent_conn;

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

        function void build_phase (uvm_phase phase);
            super.build_phase(phase);
            if(!uvm_config_db#(fifo_configuration_object)::get(this,"","FIFO_CONFIG_OBJ",obj)) begin
                `uvm_fatal("build phase","Unable to get the configuration object");
            end
            drv = fifo_driver::type_id::create("drv",this);
            sqr = fifo_sequencer::type_id::create("sqr",this);
            mtr = fifo_monitor::type_id::create("mtr",this);
            agent_conn = new("agent_conn",this);
        endfunction

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            drv.fifo_vif = obj.fifo_vif;
            mtr.fifo_vif = obj.fifo_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
            mtr.mon_ap.connect(agent_conn);
        endfunction
    endclass
endpackage
```

13-ENVIRONMENT

```
package environment_pkg;
    import sequence_item_pkg::*;
    import reset_sequence_pkg::*;
    import read_only_sequence_pkg::*;
    import write_only_sequence_pkg::*;
```

```
    import read_write_sequence_pkg::*;
    import sequencer_pkg::*;
    import agent_pkg::*;
    import coverage_pkg::*;
    import scoreboard_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class fifo_environment extends uvm_env;
        `uvm_component_utils(fifo_environment)
        fifo_agent fifo_agent_env;
        fifo_scoreboard fifo_sc;
        fifo_coverage fifo_cvr;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            fifo_agent_env=fifo_agent::type_id::create("fifo_agent_env",this);
            fifo_sc=fifo_scoreboard::type_id::create("fifo_sc",this);
            fifo_cvr=fifo_coverage::type_id::create("fifo_cvr",this);
        endfunction

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            fifo_agent_env.agent_conn.connect(fifo_sc.sc_export);
            fifo_agent_env.agent_conn.connect(fifo_cvr.cov_export);
        endfunction

    endclass
endpackage
```

14-ASSERTIONS

```
module SVA (FIFO_INT.DUT fifo_interface_instance);
    `ifdef SIM

    property write_acknowledge;
    @(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n)
(fifo_interface_instance.wr_en && !fifo_interface_instance.full) |=> fifo_interface_instance.wr_ack
    endproperty



    property full_flag;
    @(posedge fifo_interface_instance.clk) (DUT.count == fifo_interface_instance.FIFO_DEPTH) |->
fifo_interface_instance.full == 1
```

```
    endproperty


    property empty_flag;
    @(posedge fifo_interface_instance.clk) (DUT.count == 0) |-> fifo_interface_instance.empty == 1
    endproperty


    property almostfull_flag;
    @(posedge fifo_interface_instance.clk) (DUT.count == fifo_interface_instance.FIFO_DEPTH-1) |->
fifo_interface_instance.almostfull
    endproperty


    property almostempty_flag;
    @(posedge fifo_interface_instance.clk) (DUT.count == 1) |-> fifo_interface_instance.almostempty
    endproperty


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

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


    property counter_operation_up;
    @(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n) (
(({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b10) &&
(!fifo_interface_instance.full)) || (({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en}
== 2'b11) && (fifo_interface_instance.empty))) |=> DUT.count == $past(DUT.count) + 4'b0001
    endproperty


    property counter_operation_down;
    @(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n)(
(({fifo_interface_instance.wr_en, fifo_interface_instance.rd_en} == 2'b01) &&
(!fifo_interface_instance.empty)) || (({fifo_interface_instance.wr_en,
fifo_interface_instance.rd_en} == 2'b11) && (fifo_interface_instance.full))) |=> DUT.count ==
$past(DUT.count) - 4'b0001
    endproperty


    property read_pointer_operation;
```

```
    @(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n)
(fifo_interface_instance.rd_en && !fifo_interface_instance.empty) |=> DUT.rd_ptr ==
$past(DUT.rd_ptr)+3'b001
    endproperty


    property write_pointer_operatrion;
    @(posedge fifo_interface_instance.clk) disable iff(!fifo_interface_instance.rst_n)
(fifo_interface_instance.wr_en && !fifo_interface_instance.full) |=> DUT.wr_ptr ==
$past(DUT.wr_ptr)+3'b001
    endproperty

    always_comb begin
        if(!fifo_interface_instance.rst_n) begin
        a_reset: assert final(DUT.count == 0);
        b_reset: assert final(fifo_interface_instance.overflow == 0);
        c_reset: assert final(fifo_interface_instance.underflow == 0);
        d_reset: assert final(DUT.wr_ptr == 0);
        e_reset: assert final(DUT.rd_ptr == 0);
        f_reset: assert final(fifo_interface_instance.wr_ack == 0);
        end
    end

    ast1  : assert property(write_acknowledge)        else $fatal("Assertion failed: 'wr_ack' should
be asserted when write occurs and FIFO is not full!");
    cvr1  : cover property(write_acknowledge);
    ast2  : assert property(full_flag)                else $fatal("Assertion failed: 'full' flag
mismatch with fifo_count!");
    cvr2  : cover property(full_flag);
    ast3  : assert property(empty_flag)               else $fatal("Assertion failed: 'empty' flag
mismatch with fifo_count!");
    cvr3  : cover property(empty_flag);
    ast4  : assert property(almostfull_flag)          else $fatal("Assertion failed: 'almostfull'
flag mismatch with fifo_count!");
    cvr4  : cover property(almostfull_flag);
    ast5  : assert property(almostempty_flag)         else $fatal("Assertion failed: 'almostempty'
flag mismatch with fifo_count!");
    cvr5  : cover property(almostempty_flag);
    ast6  : assert property(overflow_flag)            else $fatal("Assertion failed: 'overflow' flag
mismatch if full and wr_en it should be assertd!");
    cvr6  : cover property(overflow_flag);
    ast7  : assert property(underflow_flag)           else $fatal("Assertion failed: 'underflow'
flag mismatch if empty and rd_en it should be assertd!");
    cvr7  : cover property(underflow_flag);
    ast8 : assert property(counter_operation_up)     else $fatal("Assertion failed: 'count' when
wr_en is high and rd_en is low it should increased!");
    cvr8 : cover property(counter_operation_up);
    ast9 : assert property(counter_operation_down)   else $fatal("Assertion failed: 'count' when
wr_en is low and rd_en is high it should decreased!");
```

```
    cvr9 : cover property(counter_operation_down);
    ast10 : assert property(read_pointer_operation)   else $fatal("Assertion failed: 'rd_ptr' should
increased!");
    cvr10 : cover property(read_pointer_operation);
    ast11 : assert property(write_pointer_operatrion) else $fatal("Assertion failed: 'wr_ptr' should
increased!");
    cvr11 : cover property(write_pointer_operatrion);
    `endif
endmodule
```

15-TEST

```
package test_pkg;
    import configuration_object_pkg::*;
    import environment_pkg::*;
    import reset_sequence_pkg::*;
    import write_only_sequence_pkg::*;
    import read_only_sequence_pkg::*;
    import read_write_sequence_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class fifo_test extends uvm_test;
        `uvm_component_utils(fifo_test)
        fifo_configuration_object fifo_config_obj_test;
        fifo_environment env;
        virtual FIFO_INT fifo_vif;
        fifo_reset_sequence reset_seq;
        fifo_write_only_sequence write_seq;
        fifo_read_only_sequence read_seq;
        fifo_read_write_sequence read_write_seq;
        function new(string name = "fifo_test",uvm_component parent = null);
            super.new(name,parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env=fifo_environment::type_id::create("env",this);
            fifo_config_obj_test =
fifo_configuration_object::type_id::create("fifo_config_obj_test");
            reset_seq = fifo_reset_sequence::type_id::create("reset_seq");
            write_seq = fifo_write_only_sequence::type_id::create("write_seq");
            read_seq = fifo_read_only_sequence::type_id::create("read_seq");
            read_write_seq = fifo_read_write_sequence::type_id::create("read_write_seq");
            if(!uvm_config_db#(virtual
FIFO_INT)::get(this,"","FIFO_INTERFACE",fifo_config_obj_test.fifo_vif))
                `uvm_fatal("build phase","Unable to get the virtual interface");
```

```
            uvm_config_db#(fifo_configuration_object)::set(null,"*","FIFO_CONFIG_OBJ",fifo_config_ob
j_test);
        endfunction

        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.fifo_agent_env.sqr);
            `uvm_info("run phase","Reset deasserted",UVM_LOW)
            `uvm_info("run phase","write sequence asserted",UVM_LOW)
            write_seq.start(env.fifo_agent_env.sqr);
            `uvm_info("run phase","write sequence deasserted",UVM_LOW)
            `uvm_info("run phase","read sequence asserted",UVM_LOW)
            read_seq.start(env.fifo_agent_env.sqr);
            `uvm_info("run phase","read sequence deasserted",UVM_LOW)
            `uvm_info("run phase","read write sequence asserted",UVM_LOW)
            read_write_seq.start(env.fifo_agent_env.sqr);
            `uvm_info("run phase","read sequence deasserted",UVM_LOW)
            phase.drop_objection(this);
        endtask

    endclass

endpackage
```

16-TOP MODULE

```
import test_pkg::*;
import environment_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh";
module top_module;

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

    FIFO_INT fifo_interface_instance (clk);
    FIFO DUT (fifo_interface_instance);
    bind FIFO SVA FIFO_SVA (fifo_interface_instance);
    initial begin
        uvm_config_db#(virtual
FIFO_INT)::set(null,"uvm_test_top","FIFO_INTERFACE",fifo_interface_instance);
        run_test("fifo_test");
    end
```

```
endmodule
```

17-DO FILE

```
vlib work
vlog -f src_files.list -define SIM +cover -covercells
vsim -voptargs=+acc work.top_module -cover
add wave /top_module/fifo_interface_instance/*
coverage save top_module.ucdb -onexit
run -all
#quit -sim
#vcover report top_module.ucdb -details -all -annotate -output coverage_report_fifo_uvm.txt
```

18-SOURCE LIST

```
FIFO.sv
FIFO_INT.sv
UVM_FIFO_configuration_object.sv
UVM_FIFO_Sequence_item.sv
UVM_FIFO_Reset_sequence.sv
UVM_FIFO_Write_only_sequence.sv
UVM_FIFO_Read_only_sequence.sv
UVM_FIFO_Read_write_sequence.sv
UVM_FIFO_sequencer.sv
UVM_FIFO_driver.sv
UVM_FIFO_monitor.sv
UVM_FIFO_agent.sv
UVM_FIFO_coverage.sv
UVM_FIFO_scoreboard.sv
UVM_FIFO_environment.sv
UVM_FIFO_test.sv
top_module.sv
```

19-GITHUB LINK

repo