Mohamed Emadeldin Ali
SV7 – Digital Verification diploma
No.109

UVM FIFO Project

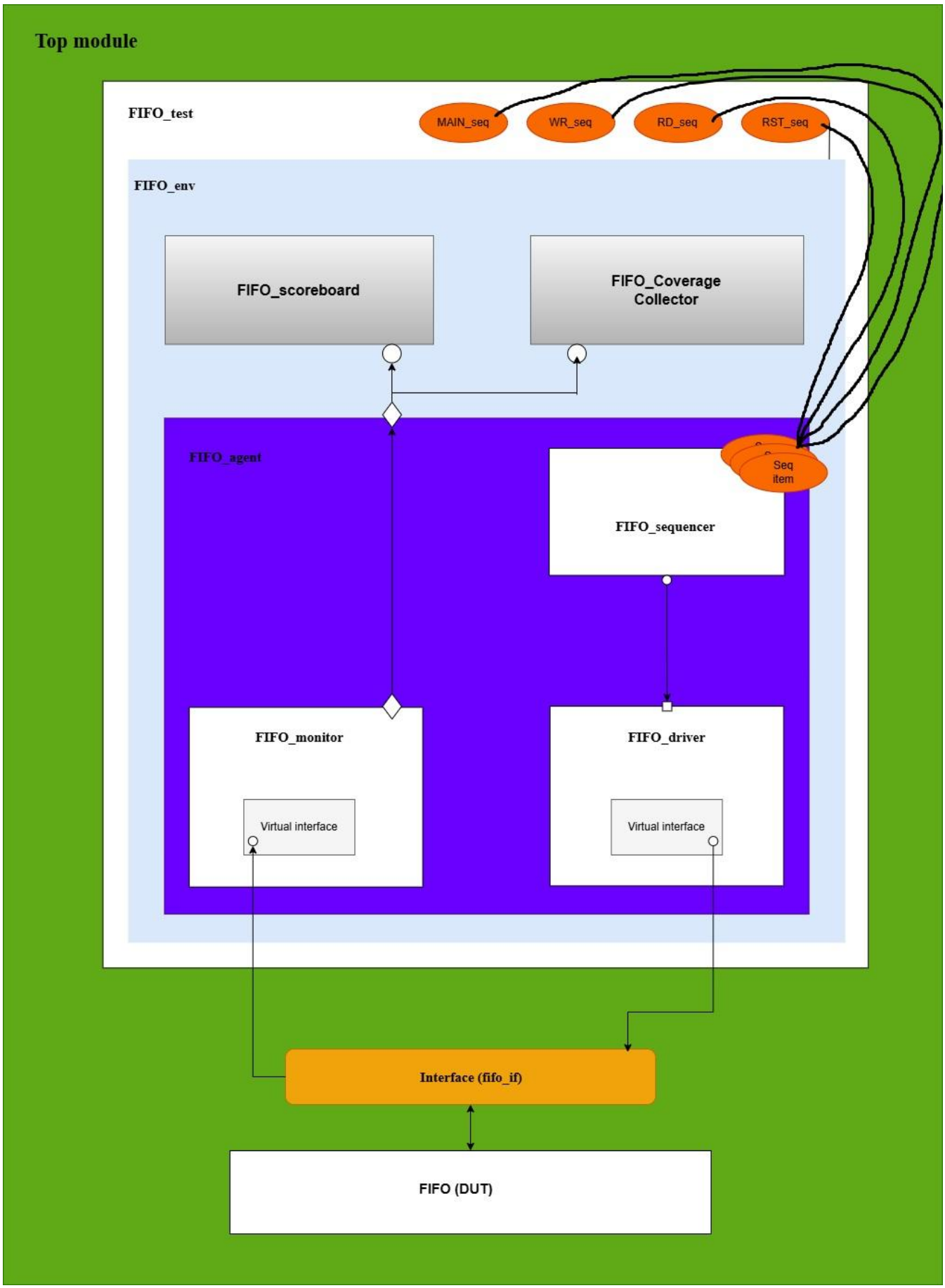
1) Bug Report :

- **wr_ack should be Low when reset is asserted .**
- **overflow should be Low when reset is asserted .**
- **underflow should be Low when reset is asserted .**
- **underflow output should be Sequential .**
- **Uncovered case when both wr_en and rd_en are high and FIFO if full , Reading process happens .**
- **Uncovered case when both wr_en and rd_en are high and FIFO if empty , Writing process happens .**
- **almostfull is high when there is two spots empty , while it should be when only one spot is empty .**

2) Verification Plan :

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted the FIFO outputs (data_out & other flags) should be low .	Directed at the start of the simulation ,then randomized with constraint that drive reset to be off most of the time .	-	Immediate assertion to check for the async reset functionality
FIFO_2	When the wr_en is asserted, the FIFO is not Full , Writing Operation takes place withl data_in input , wr_ack should be activated .	Directed with Write only Sequence that drives wr_en to 1 , rd_en to 0 , rst_n to 1 and randomizes data_in to take different values according to the constraints .	Cross Coverage betweenf wr_en with rd_en and wr_ack .	Concurrent assertion to check the wr_ack Functionality .
FIFO_3	When the FIFO is not empty & rd_en signal is high the data_out should take the value of the FIFO element with the rd_ptr address and the rd_ptr is incremented .	Directed with read only Sequence that drives wr_en to 0 , rd_en to 1 , rst_n to 1 and randomizes data_in to take different values according to the constraints .	-	Concurrent assertion to check the rd_ptr_wraparound & rd_ptr_theshold , Also a Reference model is made to check data_out Functionality .
FIFO_4	When the FIFO is full , the full flag should be activated .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and full .	Immediate assertion to check for the full Flag functionality
FIFO_5	When the FIFO is empty , the empty flag should be activated .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and empty .	Immediate assertion to check for the empty Flag functionality
FIFO_6	When the FIFO has only 1 space left , the almostfull flag should be activated .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and almostfull .	Immediate assertion to check for the almostfull Flag functionality
FIFO_7	When the FIFO has only 1 space occupied , the almostempty flag should be activated .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and almostempty .	Immediate assertion to check for the almostempty Flag functionality
FIFO_8	When the FIFO is empty and the rd_en signal is high , underflow signal should be activated .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and underflow .	Immediate assertion to check for the underflow Flag functionality
FIFO_9	When the FIFO is full and the wr_en signal is high , overflow signal should be activated .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and overflow .	Immediate assertion to check for the overflow Flag functionality

3) UVM Testbench Structure :



4) UVM Testbench Flow :

4.1 Top Module :

- Generates the clock .
- Instantiates the DUT (FIFO RTL) and the interface .
- Binds the DUT to the Assertions (SVA) module .
- Sets the interface using the configuration database to be accessed via any class below .
- Runs the test (FIFO_test) .

4.2 FIFO_test :

- Builds the Environment (FIFO_env).
- Retrieve the virtual interface from configuration database by configuration object.
- Builds different Sequences .
- Passes these Sequences to the Sequencer .

4.3 FIFO_Sequences :

- **Read_only_Sequence :** drives the DUT and Ref. model to do a Read only operation .
- **Write_only_Sequence :** drives the DUT and Ref. model to do a Write only operation .
- **Reset Sequence :** Resets the DUT and Ref. model .
- **Main Sequence :** Randomizes all stimulus .

4.4 FIFO_Sequence_item :

- Contains the constrains blocks .
- Randomizes the input Signals .

4.5 FIFO_Environment :

- Builds Scoreboard , Coverage collector and Agent .
- Connects the Agent to both Scoreboard and Coverage collector .

4.6 FIFO_Scoreboard :

- Recieves the Seq_item from the monitor (via the agent) .
- Runs Ref. model and compare output signals with ref. signals.
- Reports the no. of both correct and incorrect comparisons .

4.7 FIFO_Coverage_Collector :

- Recieves the Seq_item from the monitor (via the agent) .
- Contains the Covergroups to ensure the Functional Coverage mentioned in the Verification Plan .

4.8 FIFO_Sequencer :

- Generates Transactions as Class objects and send it to the driver to be executed .

4.9 FIFO_Driver :

- Gets the next item from the Sequencer .
- Drives this item using the Virtual interface .

4.10 FIFO_Monitor :

- Captures signals information from DUT, translates it into sequence items and finally sends it analysis components (Ports & Exports).

5) Assertions Table :

Feature	Assertion
When reset is asserted , all pointers and flags should be tied to low .	if(!fifoif.rst_n) : assert final ((!FIFO.count)&&(!FIFO.wr_ptr)&& (!FIFO.rd_ptr)&&(!fifoif.wr_ack)&&(!fifoif.underflow)&&(!fifoif.overflow));
When reset is deactivated and fifo is full of elements (count == fifo_depth) Full flag should be activated .	if((fifoif.rst_n)&&(FIFO.count == fifoif.FIFO_DEPTH)) : assert final (fifoif.full);
When reset is deactivated and fifo is almost full of elements (count == fifo_depth - 1) Almostfull flag should be activated .	if((fifoif.rst_n)&&(FIFO.count == (fifoif.FIFO_DEPTH-1))) : assert final (fifoif.almostfull);
When reset is deactivated and fifo is almost empty (count == 1) Almostempty flag should be active .	if((fifoif.rst_n)&&(FIFO.count == 1)) : assert final (fifoif.almostempty);
When reset is deactivated and fifo is empty (count == 0) , Empty flag should be active .	if((fifoif.rst_n)&&(FIFO.count == 0)) : assert final (fifoif.empty);

When reset is deactivated and fifo is not full , wr_en is asserted ,then after a clock cycle wr_ack should be activated .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (fifoif.wr_en && !fifoif.full) |=> (fifoif.wr_ack) ;
When reset is deactivated and fifo is full , wr_en is asserted ,then after a clock cycle overflow flag should be activated .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (fifoif.wr_en && fifoif.full) |=> (fifoif.overflow) ;
When reset is deactivated and fifo is empty , rd_en is asserted ,then after a clock cycle underflow flag should be activated	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (fifoif.rd_en && fifoif.empty) |=> (fifoif.underflow) ;
After writing FIFO_DEPTH entries (0 to 7), the wr_ptr should eventually wrap around back to 0.	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (fifoif.wr_en && !fifoif.full) |=> ((FIFO.wr_ptr == ($past(FIFO.wr_ptr) + 1))\|\| ((FIFO.wr_ptr==0)&&($past(FIFO.wr_ptr)+1 == 8)));
After reading FIFO_DEPTH entries (0 to 7), the rd_ptr should eventually wrap around back to 0.	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (fifoif.rd_en && !fifoif.empty) |=> ((FIFO.rd_ptr == ($past(FIFO.rd_ptr)+1))\|\| ((FIFO.rd_ptr==0)&&($past(FIFO.rd_ptr)+1 == 8)));
After writing or reading FIFO_DEPTH entries (0 to 8) the counter should wraparound .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (({fifoif.wr_en, fifoif.rd_en} == 2'b10) && !fifoif.full) || (({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.empty) |=> (FIFO.count == $past(FIFO.count)+1); @(posedge fifoif.clk) disable iff (!fifoif.rst_n) (({fifoif.wr_en, fifoif.rd_en} == 2'b01) && !fifoif.empty) || (({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.full) |=> (FIFO.count == $past(FIFO.count)-1);
Wr_ptr should not exceed FIFO_DEPTH .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (FIFO.wr_ptr < fifoif.FIFO_DEPTH) ;
Rd_ptr should not exceed FIFO_DEPTH .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (FIFO.rd_ptr < fifoif.FIFO_DEPTH) ;
Counter should not exceed FIFO_DEPTH .	@(posedge fifoif.clk) disable iff (!fifoif.rst_n) (FIFO.count <= fifoif.FIFO_DEPTH) ;

6) Design Code :

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

localparam max_fifo_addr = $clog2(fifoif.FIFO_DEPTH);

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

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


always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
  if (!fifoif.rst_n) begin
    wr_ptr <= 0;
    // BUG DETECTED : wr_ack should be Low when reset is asserted .
    fifoif.wr_ack <= 0 ;
    // BUG DETECTED : overflow should be Low when reset is asserted .
    fifoif.overflow <= 0 ;

  end
  // Writing Block
  else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH ) begin
    mem[wr_ptr] <= fifoif.data_in;
    fifoif.wr_ack <= 1;
    wr_ptr <= wr_ptr + 1;
  end
  else begin
    fifoif.wr_ack <= 0;
    if (fifoif.full & fifoif.wr_en)
      fifoif.overflow <= 1;
    else
      fifoif.overflow <= 0;
  end
end


// Reading Block
always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
  if (!fifoif.rst_n) begin
    rd_ptr <= 0;
    // BUG DETECTED : Underflow should be Low when reset is asserted .
    fifoif.underflow <= 0 ;
  end
  else if (fifoif.rd_en && count != 0 ) begin
    fifoif.data_out <= mem[rd_ptr];
    rd_ptr <= rd_ptr + 1;
  end
  // BUG DETECTED : Underflow output should be Sequential .
  else begin
    if (fifoif.empty & fifoif.rd_en)
      fifoif.underflow <= 1;
    else
      fifoif.underflow <= 0;
end
end

always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
  if (!fifoif.rst_n) begin
    count <= 0;
  end
  else begin
    if  ( ({fifoif.wr_en, fifoif.rd_en} == 2'b10) && !fifoif.full)
      count <= count + 1;
    else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b01) && !fifoif.empty)
      count <= count - 1;
      // BUG DETECTED : Uncovered case when both wr_en and rd_en are high and FIFO if full , Reading process happens .
      else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.full)
      count <= count - 1;
      //BUG DETECTED : Uncovered case when both wr_en and rd_en are high and FIFO if empty , Writing process happens .
      else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.empty)
      count <= count + 1;

  end
end

assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
assign fifoif.empty = (count == 0)? 1 : 0;
//BUG DETECTED : almostfull is high when there is two spots empty , while it should be only one .
assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0;
assign fifoif.almostempty = (count == 1)? 1 : 0;


endmodule
```

7) SVA code :

```
module fifo_sva (fifo_if.ASSR fifoif);

//Assertions

always_comb begin : RST_check
  if(!fifoif.rst_n)
  rst_assetion : assert final ((!FIFO.count)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!fifoif.wr_ack)&&(!fifoif.underflow)&&(!fifoif.overflow));
  rst_cover : cover final ((!FIFO.count)&&(!FIFO.wr_ptr)&&(!FIFO.rd_ptr)&&(!fifoif.wr_ack)&&(!fifoif.underflow)&&(!fifoif.overflow));
end

always_comb begin : Full_check
  if((fifoif.rst_n)&&(FIFO.count == fifoif.FIFO_DEPTH))
  full_assertion : assert final (fifoif.full);
  full_cover : cover final (fifoif.full);
end

always_comb begin : Almostfull_check
  if((fifoif.rst_n)&&(FIFO.count == (fifoif.FIFO_DEPTH-1)))
  almostfull_assertion : assert final (fifoif.almostfull);
  almostfull_cover : cover final (fifoif.almostfull);
end

always_comb begin : Empty_check
  if((fifoif.rst_n)&&(FIFO.count == 0))
  empty_assertion : assert final (fifoif.empty);
  empty_cover : cover final (fifoif.empty);
end

always_comb begin :Almostempty_check
  if((fifoif.rst_n)&&(FIFO.count == 1))
  almostempty_assertion : assert final (fifoif.almostempty);
  almostempty_cover : cover final (fifoif.almostempty);
end


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

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

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

property wr_ptr_wraparound;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  (fifoif.wr_en && !fifoif.full) |=> ((FIFO.wr_ptr == ($past(FIFO.wr_ptr) + 1))|| ((FIFO.wr_ptr==0)&&($past(FIFO.wr_ptr)+1 == 8)));
endproperty

property rd_ptr_wraparound;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  (fifoif.rd_en && !fifoif.empty ) |=> ((FIFO.rd_ptr == ( $past(FIFO.rd_ptr)+1 ))|| ((FIFO.rd_ptr==0)&&($past(FIFO.rd_ptr)+1 == 8)));
endproperty

property counter_wraparound_incr;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  ( ({fifoif.wr_en, fifoif.rd_en} == 2'b10) && !fifoif.full) || ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.empty)
   |=> (FIFO.count == $past(FIFO.count)+1);
endproperty

property counter_wraparound_decr;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  ( ({fifoif.wr_en, fifoif.rd_en} == 2'b01) && !fifoif.empty) || ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.full)
   |=> (FIFO.count == $past(FIFO.count)-1);
endproperty

property wr_ptr_thershold ;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  (FIFO.wr_ptr < fifoif.FIFO_DEPTH) ;
endproperty

property rd_ptr_thershold ;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  (FIFO.rd_ptr < fifoif.FIFO_DEPTH) ;
endproperty

property count_thershold ;
  @(posedge fifoif.clk)
  disable iff (!fifoif.rst_n)
  (FIFO.count <= fifoif.FIFO_DEPTH) ;
endproperty

```

```
97
98
99
100 wr_ack_assertion : assert property (wr_ack_p) ;
101 wr_ack_cover : cover property (wr_ack_p) ;
102
103 overflow_assertion : assert property (overflow_p) ;
104 overflow_cover : cover property (overflow_p) ;
105
106 underflow_assertion : assert property (underflow_p) ;
107 underflow_cover : cover property (underflow_p) ;
108
109 wr_ptr_wraparound_assertion : assert property (wr_ptr_wraparound);
110 wr_ptr_wraparound_cover : cover property (wr_ptr_wraparound);
111
112 rd_ptr_wraparound_assertion : assert property (rd_ptr_wraparound);
113 rd_ptr_wraparound_cover : cover property (rd_ptr_wraparound);
114
115 counter_wraparound_incr_assertion : assert property (counter_wraparound_incr);
116 counter_wraparound_incr_cover : cover property (counter_wraparound_incr);
117
118 counter_wraparound_decr_assertion : assert property (counter_wraparound_decr);
119 counter_wraparound_decr_cover : cover property (counter_wraparound_decr);
120
121 wr_ptr_thershold_assertion : assert property (wr_ptr_thershold);
122 wr_ptr_thershold_cover : cover property (wr_ptr_thershold);
123
124 rd_ptr_thershold_assertion : assert property (rd_ptr_thershold);
125 rd_ptr_thershold_cover : cover property (rd_ptr_thershold);
126
127 count_thershold_assertion : assert property (count_thershold);
128 count_thershold_cover : cover property (count_thershold);
129
130 endmodule
```

8) Top Module :

```
1  import fifo_test_pkg ::*;
2  import uvm_pkg ::*;
3  `include "uvm_macros.svh"
4  module top ;
5    bit clk ;
6
7    initial begin
8      clk = 0 ;
9      forever begin
10       #1 clk = ~clk ;
11     end
12   end
13
14   fifo_if fifoif (clk) ;
15   FIFO DUT (fifoif);
16
17   // Assertion binding :
18   bind FIFO fifo_sva ASSR (fifoif) ;
19
20   initial begin
21     uvm_config_db#(virtual fifo_if)::set(null , "uvm_test_top" , "FIFO_IF" , fifoif);
22     run_test("fifo_test") ;
23   end
24
25
26
27 endmodule
```

9) Interface :

```
interface fifo_if (clk);

  parameter FIFO_WIDTH = 16;
  parameter FIFO_DEPTH = 8;

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

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

endinterface

```

10) FIFO_Test :

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

import fifo_env_pkg ::* ;
import fifo_config_pkg::* ;
import fifo_sequence_pkg ::* ;

  class fifo_test extends uvm_test ;
    `uvm_component_utils(fifo_test)

    fifo_env env ;
    fifo_config_obj fifo_cfg ;
    virtual fifo_if fifo_vif ;
    fifo_reset_sequence reset_seq ;
    fifo_main_sequence main_seq ;
    fifo_read_only_sequence read_only_seq ;
    fifo_write_only_sequence write_only_seq ;

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

    function void build_phase (uvm_phase phase) ;
      super.build_phase(phase);
      env = fifo_env::type_id::create("env",this);
      fifo_cfg = fifo_config_obj::type_id::create("fifo_cfg");

      reset_seq = fifo_reset_sequence::type_id::create("reset_seq");
      main_seq = fifo_main_sequence::type_id::create("main_seq");
      read_only_seq = fifo_read_only_sequence::type_id::create("read_only_seq");
      write_only_seq = fifo_write_only_sequence::type_id::create("write_only_seq");

```

```
    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")

    uvm_config_db#(fifo_config_obj)::set(this,"*","CFG",fifo_cfg);

    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.agt.sqr);
      `uvm_info("run_phase","Reset Deasserted" , UVM_LOW)
      `uvm_info("run_phase","Write only Sequence  Started", UVM_LOW)
      write_only_seq.start(env.agt.sqr);
      `uvm_info("run_phase","Write only Sequence Ended", UVM_LOW)
      `uvm_info("run_phase","Read only Sequence  Started", UVM_LOW)
      read_only_seq.start(env.agt.sqr);
      `uvm_info("run_phase","Read only Sequence Ended", UVM_LOW)
      `uvm_info("run_phase","Main Sequence   Started", UVM_LOW)
      main_seq.start(env.agt.sqr);
      `uvm_info("run_phase","Main Sequence Ended", UVM_LOW)
      phase.drop_objection(this);

    endtask

  endclass

endpackage
```

11) FIFO_Sequences :

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

import fifo_seq_item_pkg ::*;

class fifo_reset_sequence extends uvm_sequence#(fifo_seq_item) ;
`uvm_object_utils(fifo_reset_sequence)
  fifo_seq_item seq_item ;

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

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

endclass //fifo_reset_sequence

```

```
class fifo_write_only_sequence extends uvm_sequence#(fifo_seq_item) ;
`uvm_object_utils(fifo_write_only_sequence)

  fifo_seq_item seq_item ;

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

  task body ;
    seq_item  = fifo_seq_item::type_id::create("seq_item");
    repeat (100) begin
    start_item(seq_item);
    seq_item.wr_en = 1 ;
    seq_item.rd_en = 0 ;
    seq_item.rst_n = 1 ;
    seq_item.rst_n.rand_mode(0);
    seq_item.wr_en.rand_mode(0);
    seq_item.rd_en.rand_mode(0);
    assert(seq_item.randomize())
    finish_item(seq_item);
    end
  endtask

endclass //fifo_write_only_sequence

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

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

  task body ;
    seq_item  = fifo_seq_item::type_id::create("seq_item");
    repeat (100) begin
    start_item(seq_item);
    seq_item.wr_en = 0 ;
    seq_item.rd_en = 1 ;
    seq_item.rst_n = 1 ;
    assert(seq_item.randomize(data_in))
    finish_item(seq_item);
    end
  endtask

endclass //fifo_read_only_sequence

class fifo_main_sequence extends uvm_sequence#(fifo_seq_item) ;
`uvm_object_utils(fifo_main_sequence)
  fifo_seq_item seq_item ;

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

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

endclass //fifo_main_sequence

endpackage
```

12) FIFO_Seq _item :

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

class fifo_seq_item extends uvm_sequence_item ;
`uvm_object_utils(fifo_seq_item)

  rand bit rst_n, wr_en, rd_en;
  rand logic [15:0] data_in;

  logic [15:0] data_out;
  bit  wr_ack, overflow;
  bit full, empty, almostfull, almostempty, underflow;

  int RD_EN_ON_DIST = 70 ;
  int WR_EN_ON_DIST = 70;

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

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

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

  // Constrain blocks :
  constraint rst_n_c { rst_n dist {0:/10 , 1:/90};}
  constraint wr_en_c { wr_en dist {0:/(100-WR_EN_ON_DIST)
  , 1:/WR_EN_ON_DIST};}
  constraint rd_en_c { rd_en dist {0:/(100-RD_EN_ON_DIST)
  , 1:/RD_EN_ON_DIST};}
  constraint data_in_c {
    data_in dist {
      {16{1'b0}}           := 10, // All zeros
      {16{1'b1}}           := 10, // All ones
      {8{2'b10}}    := 10, // Alternating 1/0
      {8{2'b01}}    := 10, // Alternating 0/1
      1                          := 5, // LSB only
      1 << (15)          := 5, // MSB only
      [16'h0002:16'hFFFE]                    := 50  // Rest of the values
    };
  }

endclass
endpackage
```

13) FIFO_Sequencer :

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

import fifo_seq_item_pkg ::*;

class fifo_sequencer extends uvm_sequencer #(fifo_seq_item);
`uvm_component_utils(fifo_sequencer)

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

endclass
endpackage
```

14) FIFO_Environment :

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

import fifo_coverage_collector_pkg ::*;
import fifo_agent_pkg ::*;
import fifo_scoreboard_pkg ::*;


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

  function void build_phase (uvm_phase phase);
    super.build_phase(phase);
    agt = fifo_agent::type_id::create("agt",this);
    sb = fifo_scoreboard::type_id::create("sb",this);
    cov = fifo_coverage::type_id::create("cov",this);
  endfunction

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

endclass
endpackage
```

15) FIFO_Configuration Object :

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

class fifo_config_obj extends uvm_object ;
  `uvm_object_utils(fifo_config_obj)

  virtual fifo_if fifo_vif ;
  function new(string name = "fifo_config_obj");
  super.new(name);
  endfunction

endclass
endpackage
```

16) FIFO_Scoreboard :

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

import fifo_seq_item_pkg ::*;
import fifo_config_pkg ::*;


class fifo_scoreboard extends uvm_scoreboard ;
`uvm_component_utils(fifo_scoreboard)
  uvm_analysis_export#(fifo_seq_item)sb_export;
  uvm_tlm_analysis_fifo#(fifo_seq_item)sb_fifo;
  fifo_seq_item seq_item_sb ;
  logic [15:0] data_out_ref ;
  logic [15:0] Queue [$] ;
  logic [4:0] count ;

  int error_counter = 0 ;
  int correct_counter = 0 ;

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

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

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

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


  task run_phase (uvm_phase phase);
    super.run_phase(phase);
    forever begin
      sb_fifo.get(seq_item_sb);
      Reference_model(seq_item_sb);
      #1;
      if((seq_item_sb.data_out !== data_out_ref) ) begin
        `uvm_error("run_phase",$sformatf("Comparison Failed , Transaction recieved bu DUT : %s
        while the reference data_out : 0h%0h",seq_item_sb.convert2string(),data_out_ref))
        error_counter ++ ;
      end
      else begin
        `uvm_info("run_phase",$sformatf("Correct FIFO out : %s ",seq_item_sb.convert2string() ),UVM_HIGH)
        correct_counter ++ ;
      end
    end
  endtask

  task Reference_model (fifo_seq_item seq_item_chk);
      // Reset logic
      if (!seq_item_chk.rst_n) begin
        Queue <= {};
        count <= 0;
      end
      else begin
        // Write operation if not full
        if (seq_item_chk.wr_en && count <  8) begin
          Queue.push_back(seq_item_chk.data_in);
          count <= Queue.size();
        end

        // Read operation if not empty
        if (seq_item_chk.rd_en && count != 0) begin
          data_out_ref <= Queue.pop_front();
          count <= Queue.size();
        end

      end

  endtask

  function void report_phase (uvm_phase phase);
    super.report_phase(phase);
    `uvm_info("report_phase",$sformatf("total successful transactions =
",correct_counter),UVM_MEDIUM)
    `uvm_info("report_phase",$sformatf("total failed transactions =
",error_counter),UVM_MEDIUM)

endclass
endpackage
```

17) FIFO_Coverage_Collector :

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

import fifo_seq_item_pkg ::*;

class fifo_coverage extends uvm_component ;
`uvm_component_utils(fifo_coverage)
uvm_analysis_export #(fifo_seq_item) cov_export ;
uvm_tlm_analysis_fifo #(fifo_seq_item) cov_fifo ;
fifo_seq_item seq_item_cov ;

  covergroup cvr_gp ;

    wr_en_cp: coverpoint seq_item_cov.wr_en ;
    rd_en_cp: coverpoint seq_item_cov.rd_en ;
    full_cp: coverpoint seq_item_cov.full ;
    almostfull_cp: coverpoint seq_item_cov.almostfull ;
    empty_cp: coverpoint seq_item_cov.empty ;
    almostempty_cp: coverpoint seq_item_cov.almostempty ;
    overflow_cp: coverpoint seq_item_cov.overflow ;
    underflow_cp: coverpoint seq_item_cov.underflow ;
    wr_ack_cp: coverpoint seq_item_cov.wr_ack ;

    cross_wr_rd_full:         cross wr_en_cp, rd_en_cp, full_cp;
    cross_wr_rd_almostfull:   cross wr_en_cp, rd_en_cp, almostfull_cp;
    cross_wr_rd_empty:        cross wr_en_cp, rd_en_cp, empty_cp;
    cross_wr_rd_almostempty:  cross wr_en_cp, rd_en_cp, almostempty_cp;
    cross_wr_rd_overflow:     cross wr_en_cp, rd_en_cp, overflow_cp{
      ignore_bins wr0_rd0_overflow1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(overflow_cp)intersect{1};
      ignore_bins wr0_rd1_overflow1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{1} && binsof(overflow_cp)intersect{1};
    }
    cross_wr_rd_underflow:    cross wr_en_cp, rd_en_cp, underflow_cp{
      ignore_bins wr0_rd0_underflow1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(underflow_cp)intersect{1};
      ignore_bins wr1_rd0_underflow1 = binsof(wr_en_cp)intersect{1} && binsof(rd_en_cp)intersect{0} && binsof(underflow_cp)intersect{1};
    }

    /*cross_wr_rd_wr_ack:       cross wr_en_cp, rd_en_cp, wr_ack_cp{
      ignore_bins wr0_rd1_ack_1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{1} && binsof(wr_ack_cp)intersect{1};
      ignore_bins wr0_rd0_ack_1 = binsof(wr_en_cp)intersect{0} && binsof(rd_en_cp)intersect{0} && binsof(wr_ack_cp)intersect{1};
    }*/

  endgroup

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

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

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

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

endclass


endpackage
```

18) FIFO_Agent :

```
1  package fifo_agent_pkg ;
2  import uvm_pkg ::*;
3  `include "uvm_macros.svh"
4
5  import fifo_config_pkg::* ;
6  import fifo_sequencer_pkg ::* ;
7  import fifo_monitor_pkg ::*;
8  import fifo_driver_pkg ::*;
9  import fifo_seq_item_pkg ::*;
10
11 class fifo_agent extends uvm_agent ;
12 `uvm_component_utils(fifo_agent)
13 fifo_sequencer sqr ;
14 fifo_driver drv ;
15 fifo_monitor mon ;
16 fifo_config_obj fifo_cfg ;
17 uvm_analysis_port #(fifo_seq_item) agt_ap;
18
19 function new (string name = "fifo_agent",uvm_component parent = null) ;
20   super.new(name,parent);
21 endfunction
22
23 function void build_phase (uvm_phase phase);
24   super.build_phase(phase);
25   if(!uvm_config_db#(fifo_config_obj)::get(this,"","CFG",fifo_cfg))begin
26   `uvm_fatal("build_phase","Unable to get configuration object")
27   end
28   sqr = fifo_sequencer::type_id::create("sqr",this);
29   drv = fifo_driver::type_id::create("drv",this);
30   mon = fifo_monitor::type_id::create("mon",this);
31   agt_ap = new("agt_ap",this);
32 endfunction
33
34   function void connect_phase(uvm_phase phase);
35     drv.fifo_vif = fifo_cfg.fifo_vif ;
36     mon.fifo_vif = fifo_cfg.fifo_vif ;
37     drv.seq_item_port.connect(sqr.seq_item_export);
38     mon.mon_ap.connect(agt_ap);
39   endfunction
40
41 endclass
42 endpackage
```

19) FIFO_Monitor :

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

import fifo_seq_item_pkg ::*;

class fifo_monitor extends uvm_monitor ;
`uvm_component_utils(fifo_monitor)
virtual fifo_if fifo_vif ;
fifo_seq_item rsp_seq_item ;
uvm_analysis_port#(fifo_seq_item) mon_ap ;

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

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

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

20) FIFO_Driver :

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

import fifo_config_pkg ::* ;
import fifo_seq_item_pkg ::*;

  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



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

        fifo_vif.rst_n         = stim_seq_item.rst_n;
        fifo_vif.wr_en         = stim_seq_item.wr_en;
        fifo_vif.rd_en         = stim_seq_item.rd_en;
        fifo_vif.data_in       = stim_seq_item.data_in;
        @(negedge fifo_vif.clk);


        seq_item_port.item_done();
        `uvm_info("run_phase",stim_seq_item.convert2string_stimulus(),UVM_HIGH)
      end

    endtask

  endclass
endpackage
```

21) do file :

```
vlib work
vlog -f src_files.list
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all +vcs+dumpvars
add wave /top/fifoif/*
run -all
```

22) Src_files :

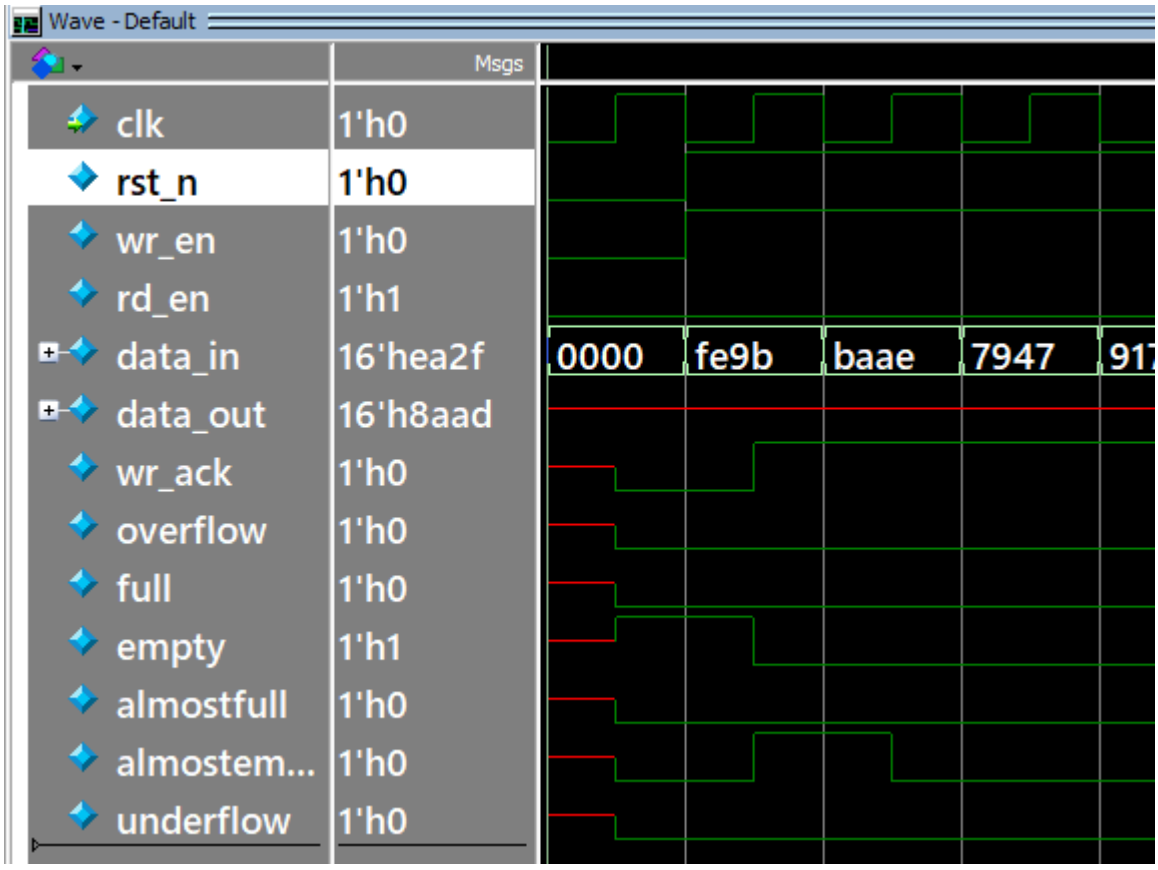
```
FIFO.sv
fifo_sva.sv
fifo_seq_item_pkg.sv
fifo_sequence_pkg.sv
fifo_coverage_collector_pkg.sv
fifo_sequencer_pkg.sv
fifo_driver_pkg.sv
fifo_monitor_pkg.sv
fifo_agent_pkg.sv
fifo_scoreboard_pkg.sv
fifo_env_pkg.sv
fifo_config_pkg.sv
fifo_test_pkg.sv
fifo_if.sv
top.sv
```

23) Questasim Transcriprt Snippet :

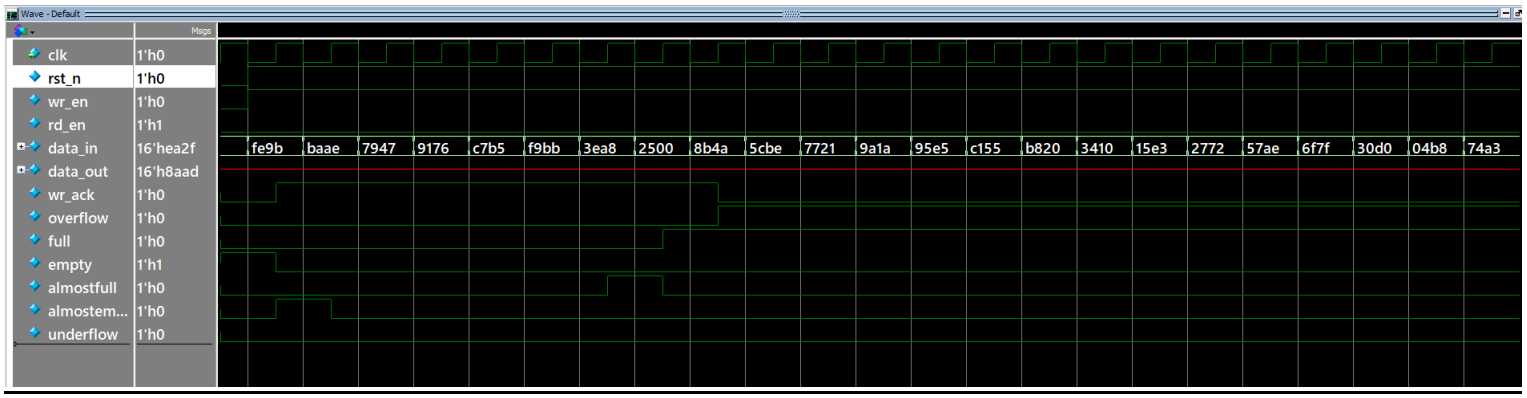
```
# UVM_INFO fifo_test_pkg.sv(46) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO fifo_test_pkg.sv(47) @ 2: uvm_test_top [run_phase] Write only Sequence  Started
# UVM_INFO fifo_test_pkg.sv(49) @ 202: uvm_test_top [run_phase] Write only Sequence Ended
# UVM_INFO fifo_test_pkg.sv(50) @ 202: uvm_test_top [run_phase] Read only Sequence  Started
# UVM_INFO fifo_test_pkg.sv(52) @ 402: uvm_test_top [run_phase] Read only Sequence Ended
# UVM_INFO fifo_test_pkg.sv(53) @ 402: uvm_test_top [run_phase] Main Sequence   Started
# UVM_INFO fifo_test_pkg.sv(55) @ 2402: uvm_test_top [run_phase] Main Sequence Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 2402: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO fifo_scoreboard_pkg.sv(80) @ 2402: uvm_test_top.env.sb [report_phase] total successful transactions = 1200
# UVM_INFO fifo_scoreboard_pkg.sv(81) @ 2402: uvm_test_top.env.sb [report_phase] total failed transactions = 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:/Users/MO  Emad/Desktop/Digital design diploma V14/Mentor_Graphics_QuestaSim_2021.1x64/Questa/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 2402 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/Users/MO  Emad/Desktop/Digital design diploma V14/Mentor_Graphics_QuestaSim_2021.1x64/Questa/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

24) Waveform Snippets :

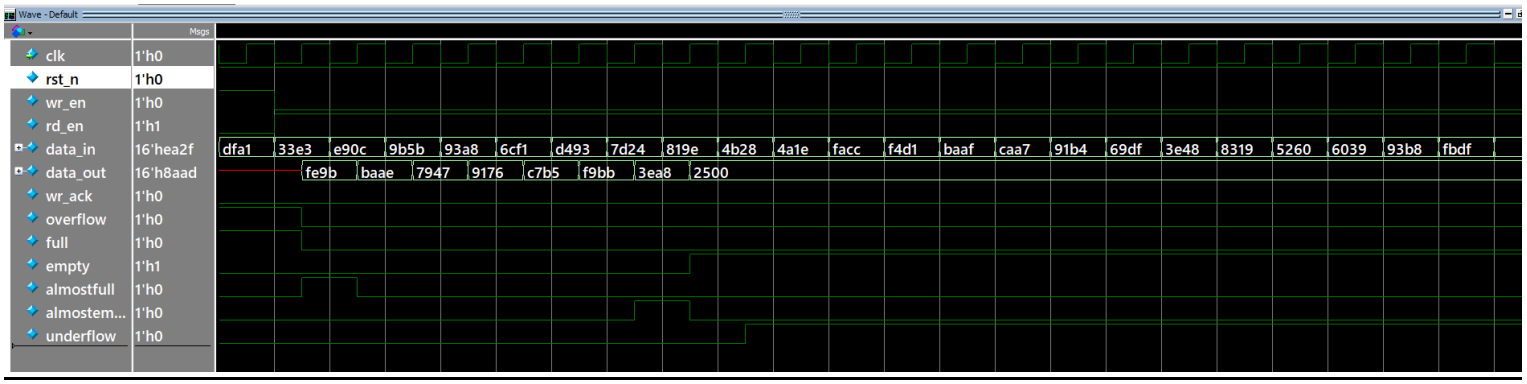
i) Reset Sequence :



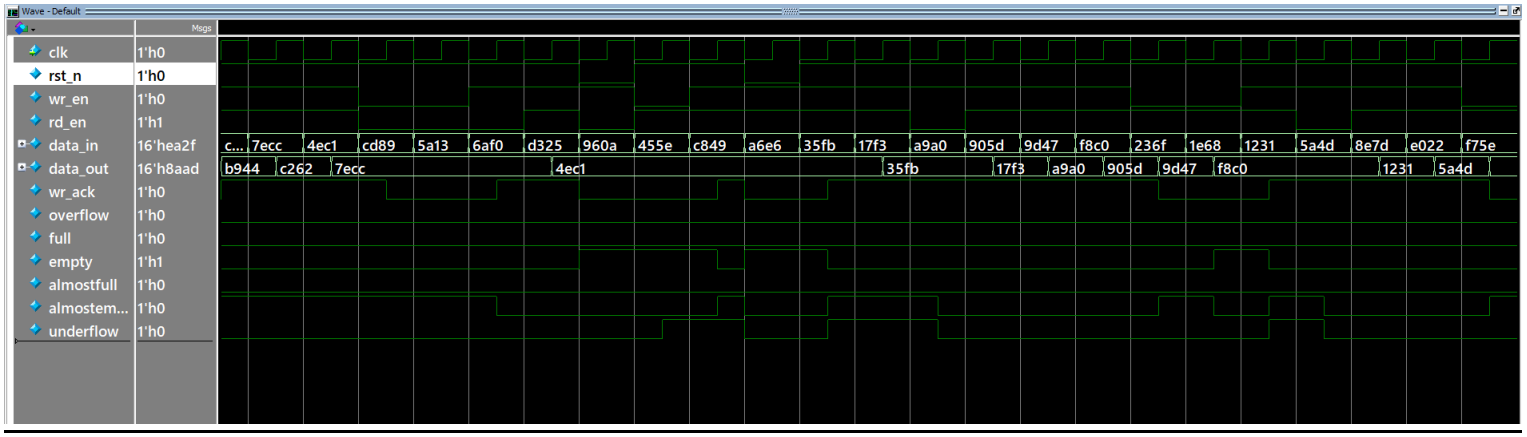
ii) Write only Sequence :



iii) Read only Sequence :



iv) Main Sequence :



25) Code Coverage :

- **Statement Coverage :**

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      32        32         0   100.00%

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

Statement Coverage for Design Unit work.FIFO --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
    17              1                       4183
    19              1                        573
    21              1                        573
    23              1                        573
    28              1                       1863
    29              1                       1863
    30              1                       1863
    33              1                       1747
    35              1                        625
    37              1                       1122
    43              1                       4183
    45              1                        573
    47              1                        573
    50              1                        992
    51              1                        992
    56              1                        132
    58              1                       2486
    62              1                       2885
    64              1                        374
    68              1                       1194
    70              1                        277
    73              1                        111
    76              1                         65
    81              1                       1817
    82              1                       1817
    84              1                       1817
    85              1                       1817
    89              1                       3260
    95              1                       2180
    101             1                       2180
    107             1                       2180
    113             1                       2180
```

- **Branch Coverage :**

```
================================================================================
=== Design Unit: work.FIFO
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        35        35         0   100.00%

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

Branch Coverage for Design Unit work.FIFO

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    18                                     4183     Count coming in to IF
    18              1                       573
    27              1                      1863
    32              1                      1747
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    34                                     1747     Count coming in to IF
    34              1                       625
    36              1                      1122
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    44                                     4183     Count coming in to IF
    44              1                       573
    49              1                       992
    54              1                      2618
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    55                                     2618     Count coming in to IF
    55              1                       132
    57              1                      2486
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                     2885     Count coming in to IF
    63              1                       374
    66              1                      2511
Branch totals: 2 hits of 2 branches = 100.00%
```

```
Branch totals: 5 hits of 5 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    81                                    1816     Count coming in to IF
    81              1                      196
    81              2                     1620
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    82                                    1816     Count coming in to IF
    82              1                      199
    82              2                     1617
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    84                                    1816     Count coming in to IF
    84              1                      282
    84              2                     1534
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    85                                    1816     Count coming in to IF
    85              1                      228
    85              2                     1588
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    90                                    3260     Count coming in to IF
    90              1                      359
                                          2901     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    96                                    2180     Count coming in to IF
    96              1                      196
                                          1984     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    102                                   2180     Count coming in to IF
    102             1                      282
                                          1898     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    108                                   2180     Count coming in to IF
    108             1                      212
                                          1968     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    114                                   2180     Count coming in to IF
```

```
-----------------------------------IF Branch------------------------------------
    114                                   2180     Count coming in to IF
    114             1                      228
                                          1952     All False Count
Branch totals: 2 hits of 2 branches = 100.00%
```

- **Toggle Coverage :**

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

================================================================================
=== Design Unit: work.fifo_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

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

Toggle Coverage for Design Unit work.fifo_if

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                    almostempty           1           1      100.00
                                     almostfull           1           1      100.00
                                            clk           1           1      100.00
                                   data_in[0-15]          1           1      100.00
                                  data_out[0-15]          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)
```

26) Assertions Coverage :

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/RST_check/rst_cover	SVA	✓	Off	404	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/Full_check/full_cover	SVA	✓	Off	181	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/Almostfull_check/almostfull_cover	SVA	✓	Off	259	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/Empty_check/empty_cover	SVA	✓	Off	424	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/Almostempty_check/almostempty_cover	SVA	✓	Off	228	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/wr_ack_cover	SVA	✓	Off	1816	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/overflow_cover	SVA	✓	Off	597	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/underflow_cover	SVA	✓	Off	114	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/wr_ptr_wraparound_cover	SVA	✓	Off	1816	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/rd_ptr_wraparound_cover	SVA	✓	Off	890	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/counter_wraparound_incr_cover	SVA	✓	Off	1246	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/counter_wraparound_decr_cover	SVA	✓	Off	320	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/wr_ptr_thershold_cover	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/rd_ptr_thershold_cover	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/count_thershold_cover	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/top/DUT/wr_ack_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (((fifoif.wr_en&&~fifoif.full))|=>fifoif.wr...	✓
/top/DUT/overflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (((fifoif.wr_en&&fifoif.full))|=>fifoif.over...	✓
/top/DUT/underflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (((fifoif.rd_en&&fifoif.empty))|=>fifoif.u...	✓
/top/DUT/wr_ptr_wraparound_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (((fifoif.wr_en&&~fifoif.full))|=>(wr_ptr...	✓
/top/DUT/rd_ptr_wraparound_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (((fifoif.rd_en&&~fifoif.empty))|=>(rd_p...	✓
/top/DUT/counter_wraparound_incr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) ((((~fifoif.rd_en&fifoif.wr_en&&~fifoif.ful...	✓
/top/DUT/counter_wraparound_decr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) ((((fifoif.rd_en&~fifoif.wr_en&&~fifoif.e...	✓
/top/DUT/wr_ptr_thershold_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (wr_ptr<fifoif.FIFO_DEPTH))	✓
/top/DUT/rd_ptr_thershold_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (rd_ptr<fifoif.FIFO_DEPTH))	✓
/top/DUT/count_thershold_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifoif.clk) disable iff (~fifoif.rst_n) (wr_ptr<=fifoif.FIFO_DEPTH))	✓
/top/DUT/RST_check/rst_assetion	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (!count&!wr_ptr&!rd_ptr&~fifoif.wr_ack&...)	✓
/top/DUT/Full_check/full_assertion	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (fifoif.full)	✓
/top/DUT/Almostfull_check/almostfull_assertion	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (fifoif.almostfull)	✓
/top/DUT/Empty_check/empty_assertion	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (fifoif.empty)	✓
/top/DUT/Almostempty_check/almostempty_assertion	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (fifoif.almostempty)	✓
/top/TEST/#ublk#217929410#11/immed__12	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (test_txn.randomize())	✓
/top/MON/#ublk#111992834#12/immed__14	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (FIFO_mon_txn.randomize())	✓