FIFO_UVM_project

Design:

F: > digital verification assignments > FIFO_UVM project > FIFO.sv

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(FIFO_if.DUT F_if);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
logic [FIFO_WIDTH-1:0] data_in;
logic clk, rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0] data_out;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

assign clk = F_if.clk;
assign data_in = F_if.data_in;
assign rst_n = F_if.rst_n;
assign wr_en = F_if.wr_en;
assign rd_en = F_if.rd_en;
assign F_if.data_out = data_out;
assign F_if.wr_ack = wr_ack;
assign F_if.overflow = overflow;
assign F_if.full = full;
assign F_if.empty = empty;
assign F_if.almostfull = almostfull;
assign F_if.almostempty = almostempty;
assign F_if.underflow = underflow;

```

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
        overflow <= 0;
        underflow <= 0;//underflow is sure to be low as no operations occur other than rst
        wr_ack <=0;//wr_ack is sure to be low as no operations occur other than rst
    end
    else if (wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        overflow <= 0;//no overflow occurs if write happens
    end
    else begin
        wr_ack <= 0;
        if (full && wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rd_ptr <= 0;
    end
    else if (rd_en && count != 0) begin
        data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        underflow <= 0;//no underflow occurs if read happens
    end
    else begin //added this else as underflow is a sequential signal not combinational
```

```
69      else begin //added this else as underflow is a sequential signal not combinational
70          if ((empty && rd_en))
71              underflow <= 1;
72          else
73              underflow <= 0;
74      end
75  end
76
77  always @(posedge clk or negedge rst_n) begin
78      if (!rst_n) begin
79          count <= 0;
80      end
81      else begin
82          if  ( ({wr_en, rd_en} == 2'b10) && !full)
83              count <= count + 1;
84          else if ( ({wr_en, rd_en} == 2'b01) && !empty)
85              count <= count - 1;
86          else if ( ({wr_en, rd_en} == 2'b11) && empty)//added this case as only write happens which should increment the counter
87              count <= count + 1;
88          else if ( ({wr_en, rd_en} == 2'b11) && full)//added this case as only read happens which should decrement the counter
89              count <= count - 1;
90      end
91  end
92
93  assign full = (count == FIFO_DEPTH)? 1 : 0;
94  assign empty = (count == 0)? 1 : 0;
95  //assign underflow = (empty && rd_en)? 1 : 0; removed this as underflow is a sequential signal not combinational
96  assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;//changed from FIFO_DEPTH-2 to FIFO_DEPTH-1
97  assign almostempty = (count == 1)? 1 : 0;
98
99  endmodule
```

Bugs were found in lines 42,43,49,67,69,86-89,95 . comments are written in the code to explain the change and why it was required.

Interface:

```
1   interface FIFO_if(clk);
2   parameter FIFO_WIDTH = 16;
3   parameter FIFO_DEPTH = 8;
4   input bit clk;
5   logic [FIFO_WIDTH-1:0] data_in;
6   logic rst_n, wr_en, rd_en;
7   logic [FIFO_WIDTH-1:0] data_out;
8   logic wr_ack, overflow;
9   logic full, empty, almostfull, almostempty, underflow;
10
11  modport DUT (input data_in, wr_en, rd_en, clk, rst_n, output full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
12
13  modport TEST (input full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out, clk, output data_in, wr_en, rd_en, rst_n);
14
15  modport MONITOR (input full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out, clk, data_in, wr_en, rd_en, rst_n);
16
17  endinterface
```

FIFO_sequence_item:

```
package sequence_item_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
class FIFO_seq_item extends uvm_sequence_item;
 `uvm_object_utils(FIFO_seq_item)
rand bit [FIFO_WIDTH-1:0] data_in;
rand bit rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0] data_out;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;

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

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

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

constraint rst_n_C{rst_n dist{0:=3,1:=97};}//no reset 97% of the time
constraint wr_en_C{wr_en dist{1:=70,0:=30};}
constraint rd_en_C{wr_en dist{1:=30,0:=70};}
endclass
endpackage
```

FIFO_sequences:

```
package sequence_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import sequence_item_pkg::*;
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;
 repeat(10000) begin
     seq_item = FIFO_seq_item::type_id::create("seq_item");
     start_item(seq_item);
     assert(seq_item.randomize);
     finish_item(seq_item);
 end
endtask
endclass

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.data_in=0;
     seq_item.wr_en=0;
     seq_item.rd_en=0;
```

```
        seq_item.rd_en=0;
        finish_item(seq_item);
    endtask
    endclass

    class FIFO_write_sequence extends uvm_sequence #(FIFO_seq_item);
     `uvm_object_utils(FIFO_write_sequence)
     FIFO_seq_item seq_item;

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

    task body;
     repeat(20) begin
        seq_item = FIFO_seq_item::type_id::create("seq_item");
        start_item(seq_item);
        seq_item.rst_n=1;
        seq_item.data_in=$random;
        seq_item.wr_en=1;
        seq_item.rd_en=0;
        finish_item(seq_item);
     end
    endtask
    endclass

    class FIFO_read_sequence extends uvm_sequence #(FIFO_seq_item);
     `uvm_object_utils(FIFO_read_sequence)
     FIFO_seq_item seq_item;

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

```

```
task body;
 repeat(20) begin
    seq_item = FIFO_seq_item::type_id::create("seq_item");
    start_item(seq_item);
    seq_item.rst_n=1;
    seq_item.data_in=$random;
    seq_item.wr_en=0;
    seq_item.rd_en=1;
    finish_item(seq_item);
 end
endtask
endclass

endpackage
```

FIFO_sequencer:

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

FIFO_config:

```
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
```

FIFO_driver:

```
package FIFO_driver_pkg;
import uvm_pkg::*;
import FIFO_config_pkg::*;
import sequence_item_pkg::*;
`include "uvm_macros.svh"
class FIFO_driver extends uvm_driver #(FIFO_seq_item);
 `uvm_component_utils(FIFO_driver)
virtual FIFO_if FIFO_vif;
FIFO_seq_item stim_seq_item;

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

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.data_in=stim_seq_item.data_in;
    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;
    @(negedge FIFO_vif.clk);
    seq_item_port.item_done();
    `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
  end
endtask
endclass
endpackage
```

FIFO_Monitor:

```
package monitor_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import sequence_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.data_in=FIFO_vif.data_in;
    rsp_seq_item.rst_n=FIFO_vif.rst_n;
    rsp_seq_item.wr_en=FIFO_vif.wr_en;
    rsp_seq_item.rd_en=FIFO_vif.rd_en;
    rsp_seq_item.data_out=FIFO_vif.data_out;
    rsp_seq_item.wr_ack=FIFO_vif.wr_ack;
    rsp_seq_item.overflow=FIFO_vif.overflow;
    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.underflow=FIFO_vif.underflow;
```

```
        rsp_seq_item.underflow=FIFO_vif.underflow;
        mon_ap.write(rsp_seq_item);
        `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)
      end
    endtask
    endclass
    endpackage
```

FIFO_Agent:

```
package agent_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import sequencer_pkg::*;
import FIFO_driver_pkg::*;
import monitor_pkg::*;
import FIFO_config_pkg::*;
import sequence_item_pkg::*;
class FIFO_agent extends uvm_agent;
 `uvm_component_utils(FIFO_agent)
  FIFO_sequencer sqr;
  FIFO_driver drv;
  FIFO_monitor mon;
  FIFO_config_obj FIFO_cfg;
  uvm_analysis_port #(FIFO_seq_item) agt_ap;



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_config_obj)::get(this,"","CFG",FIFO_cfg ))begin
   `uvm_fatal("build_phase", "unable to get configuration object");
 end
 sqr= FIFO_sequencer::type_id::create("sqr",this);
 drv= FIFO_driver::type_id::create("drv",this);
 mon= FIFO_monitor::type_id::create("mon",this);
 agt_ap= new("agt_ap",this);
endfunction

function void connect_phase (uvm_phase phase);
  drv.FIFO_vif = FIFO_cfg.FIFO_vif;
  mon.FIFO_vif = FIFO_cfg.FIFO_vif;
  drv.seq_item_port.connect(sqr.seq_item_export);
```

```
  function void connect_phase (uvm_phase phase);
    drv.FIFO_vif = FIFO_cfg.FIFO_vif;
    mon.FIFO_vif = FIFO_cfg.FIFO_vif;
    drv.seq_item_port.connect(sqr.seq_item_export);
    mon.mon_ap.connect(agt_ap);
  endfunction
endclass
endpackage
```

FIFO_Coverage:

```
package Coverage_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import sequence_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: coverpoint seq_item_cov.wr_en;
rd_en: coverpoint seq_item_cov.rd_en;
overflow: coverpoint seq_item_cov.overflow;
almostempty: coverpoint seq_item_cov.almostempty;
empty: coverpoint seq_item_cov.empty;
almostfull: coverpoint seq_item_cov.almostfull;
underflow: coverpoint seq_item_cov.underflow;
full: coverpoint seq_item_cov.full;
wr_ack: coverpoint seq_item_cov.wr_ack;

```

```
wr_ack_cvr:cross wr_en, rd_en, wr_ack {illegal_bins wr_wr_ack = binsof(wr_en) intersect {0} && binsof(wr_ack) intersect {1};}//no write achkn
overflow_cvr:cross wr_en, rd_en, overflow{illegal_bins write_overflow = binsof(wr_en) intersect {0} && binsof(overflow) intersect {1};}//no o
full_cvr:cross wr_en, rd_en, full{illegal_bins read_full = binsof(rd_en) intersect {1} && binsof(full) intersect {1};}//the fifo can't be ful
empty_cvr:cross wr_en, rd_en, empty;
almostfull_cvr:cross wr_en, rd_en, almostfull;
almostempty_cvr:cross wr_en, rd_en, almostempty;
underflow_cvr:cross wr_en, rd_en, underflow{illegal_bins read_full = binsof(rd_en) intersect {0} && binsof(underflow) intersect {1};}//no und
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
```

FIFO_Scoreboard:

```
package Scoreboard_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import sequence_item_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 [FIFO_WIDTH-1:0] data_out_ref;
logic wr_ack_ref, overflow_ref;
logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;
bit [FIFO_WIDTH-1:0] FIFO_queue[$];
integer count=0;

int error_count = 0;
int correct_count =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 connect_phase(uvm_phase phase);
  super.connect_phase(phase);
  sb_export.connect(sb_fifo.analysis_export);
endfunction

```

```
35  task run_phase(uvm_phase phase);
36    super.run_phase(phase);
37    forever begin
38      sb_fifo.get(seq_item_sb);
39      reference_model(seq_item_sb);
40      if(data_out_ref!== seq_item_sb.data_out ||
41       (wr_ack_ref!== seq_item_sb.wr_ack) ||
42       (overflow_ref!== seq_item_sb.overflow) ||
43       (full_ref!== seq_item_sb.full) ||
44       (empty_ref!== seq_item_sb.empty) ||
45       (almostfull_ref!== seq_item_sb.almostfull) ||
46       (almostempty_ref!== seq_item_sb.almostempty) ||
47       (underflow_ref!== seq_item_sb.underflow))begin
48        `uvm_error("run_phase", $sformatf("comparison failed, transaction received by the DUT:%s while the reference data_c
49        error_count = error_count + 1;
50      end
51    else
52    `uvm_info("run_phase", $sformatf("Correct FIFO out: %s", seq_item_sb.convert2string()), UVM_HIGH);
53    correct_count = correct_count + 1;
54     end
55  endtask
56
57  task reference_model (FIFO_seq_item seq_item_chk);
58  if(seq_item_chk.rst_n==0) begin
59    overflow_ref=0;
60    full_ref=0;
61    empty_ref=1;
62    almostfull_ref=0;
63    almostempty_ref=0;
64    count=0;
65    wr_ack_ref=0;
66    FIFO_queue.delete();
67    underflow_ref=0;
68  end
```

```
else if(seq_item_chk.wr_en==1 && seq_item_chk.rd_en==1 && count==8)begin
    data_out_ref=FIFO_queue.pop_back();
    wr_ack_ref=0;
    overflow_ref=1;
    underflow_ref=0;
    count=count-1;
end
else if(seq_item_chk.wr_en==1 && seq_item_chk.rd_en==1 && count==0)begin
    FIFO_queue.push_front(seq_item_chk.data_in);
    wr_ack_ref=1;
    overflow_ref=0;
    underflow_ref=1;
    count=count+1;
end
else if(seq_item_chk.wr_en==1 && count==8)begin
    wr_ack_ref=0;
    overflow_ref=1;
    underflow_ref=0;
end
else if(seq_item_chk.rd_en==1 && count==0)begin
    wr_ack_ref=0;
    overflow_ref=0;
    underflow_ref=1;
end
else if(seq_item_chk.wr_en==1 && seq_item_chk.rd_en==1)begin
    FIFO_queue.push_front(seq_item_chk.data_in);
    data_out_ref=FIFO_queue.pop_back();
    wr_ack_ref=1;
    overflow_ref=0;
    underflow_ref=0;
end
else if(seq_item_chk.wr_en==1)begin
    FIFO_queue.push_front(seq_item_chk.data_in);
    count=count+1;
```

```
else if(seq_item_chk.wr_en==1)begin
    count=count+1;
    wr_ack_ref=1;
    overflow_ref=0;
    underflow_ref=0;
end
else if(seq_item_chk.rd_en==1)begin
    data_out_ref=FIFO_queue.pop_back();
    count=count-1;
    wr_ack_ref=0;
    overflow_ref=0;
    underflow_ref=0;
end
else begin
    overflow_ref=0;
    underflow_ref=0;
    wr_ack_ref=0;
end
if(count == 0)begin
  full_ref=0;
  empty_ref=1;
  almostfull_ref=0;
  almostempty_ref=0;
end
else if(count == 8)begin
  full_ref=1;
  empty_ref=0;
  almostfull_ref=0;
  almostempty_ref=0;
end
else if(count == 7)begin
  full_ref=0;
  empty_ref=0;
  almostfull_ref=1;
  almostempty_ref=0;
end
```

```
else if(count == 1)begin
  full_ref=0;
  empty_ref=0;
  almostfull_ref=0;
  almostempty_ref=1;
end
else begin
  full_ref=0;
  empty_ref=0;
  almostfull_ref=0;
  almostempty_ref=0;
end
endtask

function void report_phase(uvm_phase phase);
 super.report_phase(phase);
 `uvm_info("report_phase", $sformatf("Total successful transactions: %0d",correct_count), UVM_MEDIUM);
 `uvm_info("report_phase", $sformatf("Total failed transactions: %0d",error_count), UVM_MEDIUM);
endfunction
endclass
endpackage
```

FIFO_env:

```
package FIFO_env_pkg;
import uvm_pkg::*;
import agent_pkg::*;
import Coverage_pkg::*;
import Scoreboard_pkg::*;
`include "uvm_macros.svh"
class FIFO_env extends uvm_env;
  `uvm_component_utils(FIFO_env);

FIFO_agent agt;
FIFO_Scoreboard sb;
FIFO_Coverage cov;

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: build_phase

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

endclass
endpackage
```

FIFO_test:

```
package FIFO_test_pkg;
import uvm_pkg::*;
import FIFO_env_pkg::*;
import FIFO_driver_pkg::*;
import FIFO_config_pkg::*;
import sequence_pkg::*;
`include "uvm_macros.svh"

class FIFO_test extends uvm_test;
`uvm_component_utils(FIFO_test)

FIFO_env env;
virtual FIFO_if FIFO_vif;
FIFO_config_obj FIFO_config_obj_test;
FIFO_main_sequence main_seq;
FIFO_reset_sequence reset_seq;
FIFO_write_sequence write_seq;
FIFO_read_sequence read_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_config_obj_test = FIFO_config_obj::type_id::create("FIFO_config_obj_test",this);
 main_seq = FIFO_main_sequence::type_id::create("main_seq",this);
 reset_seq = FIFO_reset_sequence::type_id::create("reset_seq",this);
 write_seq = FIFO_write_sequence::type_id::create("write_seq",this);
 read_seq = FIFO_read_sequence::type_id::create("read_seq",this);

 if(!uvm_config_db #(virtual FIFO_if)::get(this,"","FIFO_if",FIFO_config_obj_test.FIFO_vif ))
   `uvm_fatal("build_phase", "Test - unable to get the virtual interface of the FIFO from the uvm_config_db");

 uvm_config_db #(FIFO_config_obj)::set(this, "*", "CFG", FIFO_config_obj_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.agt.sqr);
 `uvm_info("run_phase","reset Deasserted", UVM_LOW)
 //
 `uvm_info("run_phase","full write operation started", UVM_LOW)
 write_seq.start(env.agt.sqr);
 `uvm_info("run_phase","full write operation  ended", UVM_LOW)
 //
 `uvm_info("run_phase","full read operation  started", UVM_LOW)
 read_seq.start(env.agt.sqr);
 `uvm_info("run_phase","full read operation  ended", UVM_LOW)
 //
 `uvm_info("run_phase","Stimulus Generation started", UVM_LOW)
 main_seq.start(env.agt.sqr);
 `uvm_info("run_phase","Stimulus Generation ended", UVM_LOW)
 phase.drop_objection(this);
endtask
endclass
endpackage
```

FIFO_sva:

```
module FIFO_sva(FIFO_if.DUT F_if);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
localparam max_fifo_addr = $clog2(FIFO_DEPTH);
logic [FIFO_WIDTH-1:0] data_in;
logic clk, rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0] data_out;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;
logic [max_fifo_addr:0] count;

assign clk = F_if.clk;
assign data_in = F_if.data_in;
assign rst_n = F_if.rst_n;
assign wr_en = F_if.wr_en;
assign rd_en = F_if.rd_en;
assign data_out = F_if.data_out;
assign wr_ack = F_if.wr_ack;
assign overflow = F_if.overflow;
assign full = F_if.full;
assign empty = F_if.empty;
assign almostfull = F_if.almostfull;
assign almostempty = F_if.almostempty;
assign underflow = F_if.underflow;
assign wr_ptr = dut.wr_ptr;
assign rd_ptr = dut.rd_ptr;
assign count = dut.count;

property p_rst;
    @(negedge clk) ((rst_n==0) |-> (rd_ptr==0 && wr_ptr==0 && count==0 && full==0 && empty==1 && almostfull==0 && almostempty==0));
endproperty

property p_full;
    @(posedge clk) ((count==FIFO_DEPTH) |-> (full==1 && empty==0 && almostfull==0 && almostempty==0));
endproperty

```

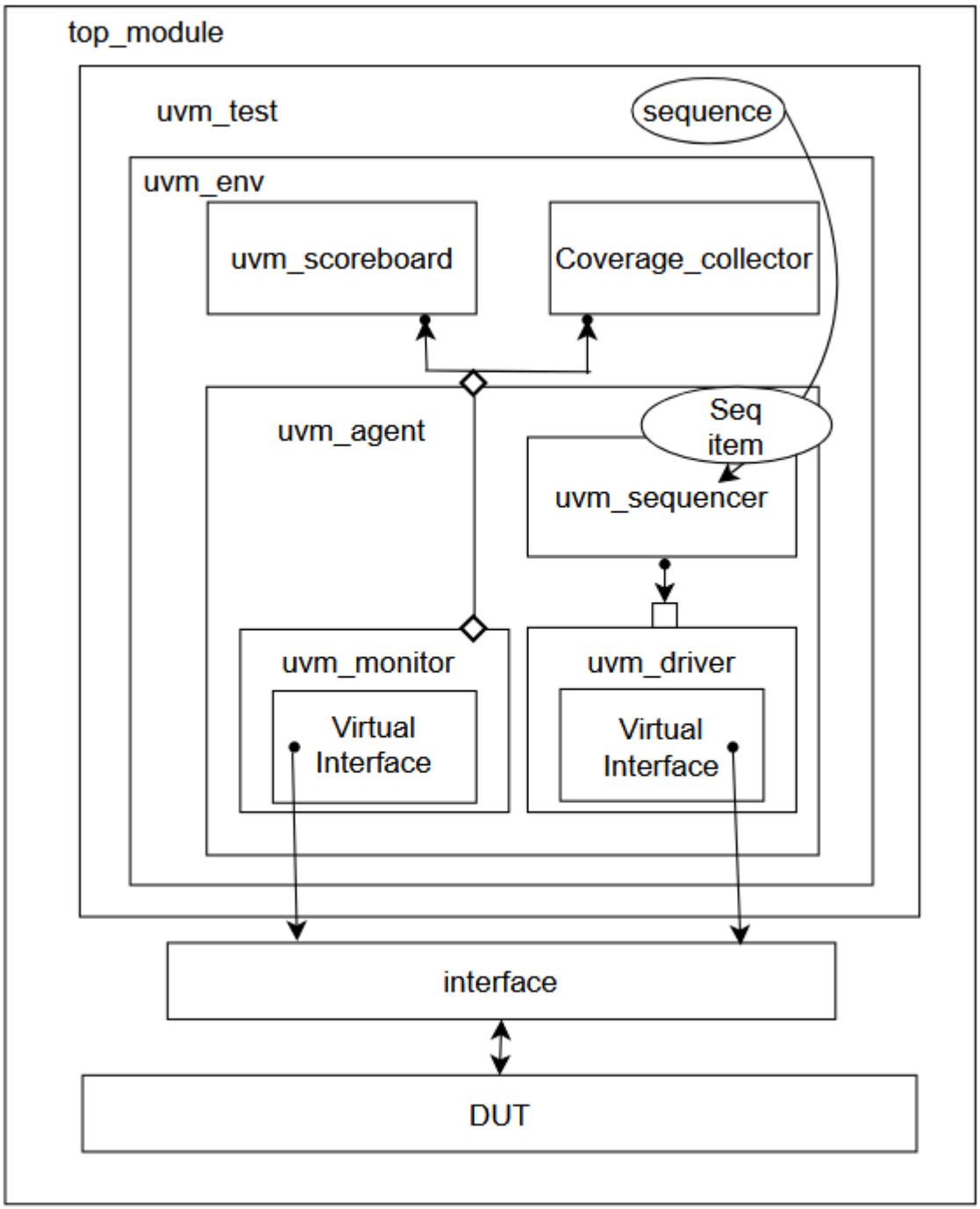
```
property p_almostfull;
    @(posedge clk) ((count==FIFO_DEPTH-1) |-> (full==0 && empty==0 && almostfull==1 && almostempty==0));
endproperty

property p_almostempty;
    @(posedge clk) ((count==1) |-> (full==0 && empty==0 && almostfull==0 && almostempty==1));
endproperty

property p_empty;
    @(posedge clk) ((count==0) |-> (full==0 && empty==1 && almostfull==0 && almostempty==0));
endproperty

property p_underflow;
    @(posedge clk) disable iff (!rst_n)((rd_en && empty) |=> (underflow==1));
endproperty

property p_overflow;
    @(posedge clk) disable iff (!rst_n)((wr_en && full) |=> (overflow==1));
endproperty

property p_rd_ptr;
    @(posedge clk) disable iff (!rst_n)((rd_en && !empty) |=> (rd_ptr==$past(rd_ptr)+1'b1));
endproperty

property p_wr_ptr;
    @(posedge clk) disable iff (!rst_n)((wr_en && !full) |=> (wr_ptr==$past(wr_ptr)+1'b1));
endproperty

property p_wr_ack;
    @(negedge clk) (( wr_en==1 && rst_n==1 && overflow==0) |-> (wr_ack==1));
endproperty
```

```
rst_assertion: assert property(p_rst);
full_assertion: assert property(p_full);
almostfull_assertion: assert property(p_almostfull);
almostempty_assertion: assert property(p_almostempty);
empty_assertion: assert property(p_empty);
underflow_assertion: assert property(p_underflow);
overflow_assertion: assert property(p_overflow);
rd_ptr_assertion: assert property(p_rd_ptr);
wr_ptr_assertion: assert property(p_wr_ptr);
wr_ack_assertion: assert property(p_wr_ack);

rst_cover: cover property(p_rst);
full_cover: cover property(p_full);
almostfull_cover: cover property(p_almostfull);
almostempty_cover: cover property(p_almostempty);
empty_cover: cover property(p_empty);
underflow_cover: cover property(p_underflow);
overflow_cover: cover property(p_overflow);
rd_ptr_cover: cover property(p_rd_ptr);
wr_ptr_cover: cover property(p_wr_ptr);
wr_ack_cover: cover property(p_wr_ack);

endmodule
```

FIFO_top:

```
import FIFO_test_pkg::*;
import uvm_pkg::*;
import FIFO_driver_pkg::*;
import FIFO_env_pkg::*;
`include "uvm_macros.svh"
module FIFO_top();
 bit clk;

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

 FIFO_if F_if(clk);
 FIFO dut(F_if);
 bind FIFO FIFO_sva FIFO_sva_inst(F_if);

initial begin
  uvm_config_db#(virtual FIFO_if)::set(null, "uvm_test_top","FIFO_if", F_if);
  run_test("FIFO_test");
end

endmodule
```

UVM Testbench Drwaing:



UVM flow:

1-the top module instantiates both the Interface and DUT while generating the clock and then passes the interface to the configuration database

2-The test starts by initializing the UVM environment and configuration object and triggering sequences.

3-The configuration is passed down to the environment and further to the agent, driver, sequencer, and monitor.

4-The sequencer generates sequence items, which are sent to the driver.

5-The driver translates these sequence items into transactions that stimulate the DUT.

6-The DUT responds, and its outputs are observed by the monitor.

7-The monitor sends the observed data to the scoreboard and coverage collector.

8-The scoreboard checks the DUT's output against expected values, and the coverage collector tracks what portions of the DUT's functionality have been tested.

Verification plan:

	A	B	C	D	E
1	**Label**	**Description**	**Stimulus Generation**	**Functional Coverage (Later)**	**Functionality Check**
2	FIFO_1	when rst_n is low ,the FIFO pointers and count should be low.	directed at start of simulation then Randomized under constraints that drive the rst_n to be high 97% of the time		A checker in the scoreboard package to make sure the output is correct, and also an immediate assertion to check for functionality
3	FIFO_2	when wr_en is high and FIFO is not full , data_in should be stored in FIFO with the wr_ack turning high and the wr_ptr incrementing, and if full then wr_en should be ignored with the overflow signal turning high.	directed after reset sequence then Randomized under constraints that drive the wr_en to be high 70% of the time	helps in cross_covering all bin combinations of wr_en, rd_en and all output signals except for specified illegal_bins	A checker in the scoreboard package to make sure the output is correct, and also an immediate assertion to check for full, almostfull signals as they re combinational, and concurrent assertions to check for overflow and wr_ptr signals.
4	FIFO_3	when rd_en is high and FIFO is not empty, data_out should be earliest data_in that was written and not read before, and the rd_ptr should increment, and if empty then rd_en should be ignored with the underflow signal turning high.	directed after write sequence then Randomized under constraints that drive the rd_en to be high 30% of the time	helps in cross_covering all bin combinations of wr_en, rd_en and all output signals except for specified illegal_bins	A checker in the scoreboard package to make sure the output is correct, and also an immediate assertion to check for empty and almostempty signals as they re combinational, and concurrent assertions to check for underflow and rd_ptr signals.

Assertions:

Feature	Assertion

Whenever rst_n is low, all counters should be reset and empty flag should be high.	@(negedge clk) ((rst_n==0) |-> (rd_ptr==0 && wr_ptr==0 && count==0 && full==0 && empty==1 && almostfull==0 && almostempty==0))
Whenever FIFO is full (count is highest value), the full flag should be high.	@(posedge clk) ((count==FIFO_DEPTH) |-> (full==1 && empty==0 && almostfull==0 && almostempty==0))
Whenever FIFO has only one place available for writing, the almostfull flag should be high.	@(posedge clk) ((count==FIFO_DEPTH-1) |-> (full==0 && empty==0 && almostfull==1 && almostempty==0))
Whenever FIFO has only one place written in it, the almostempty flag should be high.	@(posedge clk) ((count==1) |-> (full==0 && empty==0 && almostfull==0 && almostempty==1))
Whenever FIFO has nothing to read the almostempty flag should be high.	@(posedge clk) ((count==0) |-> (full==0 && empty==1 && almostfull==0 && almostempty==0))
Whenever FIFO attempts to read while its empty the underflow flag should be high.	@(posedge clk) disable iff (!rst_n)((rd_en && empty) |=> (underflow==1))
Whenever FIFO attempts to write while its full the overflow flag should be high.	@(posedge clk) disable iff (!rst_n)((wr_en && full) |=> (overflow==1))
Whenever FIFO attempts to read and is successful rd_ptr should increment.	@(posedge clk) disable iff (!rst_n)((rd_en && !empty) |=> (rd_ptr==$past(rd_ptr)+1'b1))
Whenever FIFO attempts to write and is successful wr_ptr should increment.	@(posedge clk) disable iff (!rst_n)((wr_en && !full) |=> (wr_ptr==$past(wr_ptr)+1'b1));
Whenever FIFO attempts to write and is successful wr_ack should be high.	@(negedge clk) ((wr_en==1 && rst_n==1 && overflow==0) |-> (wr_ack==1))

DO file:

run.do - Notepad

File Edit Format View Help

```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.FIFO_top -cover
add wave /FIFO_top/F_if/*
add wave -position insertpoint  \
sim:/FIFO_top/dut/mem \
sim:/FIFO_top/dut/wr_ptr \
sim:/FIFO_top/dut/rd_ptr \
sim:/FIFO_top/dut/count
coverage save top.ucdb -onexit
run -all
```

Src_files.list:

src_files.list - Notepad

File Edit Format View Help

```
FIFO.sv
FIFO_interface.sv
FIFO_sequence_item.sv
FIFO_sequences.sv
FIFO_sequencer.sv
FIFO_config_pkg.sv
FIFO_driver.sv
FIFO_monitor_pkg.sv
FIFO_agent.sv
FIFO_coverage.sv
FIFO_Scoreboard.sv
FIFO_env.sv
FIFO_test.sv
FIFO_sva.sv
FIFO_top.sv
```

Code, Functional and Sequential Domain Coverage:

```
================================================================================
=== Instance: /FIFO_top/dut/FIFO_sva_inst
=== Design Unit: work.FIFO_sva
================================================================================

Assertion Coverage:
    Assertions                      10        10         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/FIFO_top/dut/FIFO_sva_inst/rst_assertion
                     FIFO_sva.sv(69)                    0          1
/FIFO_top/dut/FIFO_sva_inst/full_assertion
                     FIFO_sva.sv(70)                    0          1
/FIFO_top/dut/FIFO_sva_inst/almostfull_assertion
                     FIFO_sva.sv(71)                    0          1
/FIFO_top/dut/FIFO_sva_inst/almostempty_assertion
                     FIFO_sva.sv(72)                    0          1
/FIFO_top/dut/FIFO_sva_inst/empty_assertion
                     FIFO_sva.sv(73)                    0          1
/FIFO_top/dut/FIFO_sva_inst/underflow_assertion
                     FIFO_sva.sv(74)                    0          1
/FIFO_top/dut/FIFO_sva_inst/overflow_assertion
                     FIFO_sva.sv(75)                    0          1
/FIFO_top/dut/FIFO_sva_inst/rd_ptr_assertion
                     FIFO_sva.sv(76)                    0          1
/FIFO_top/dut/FIFO_sva_inst/wr_ptr_assertion
                     FIFO_sva.sv(77)                    0          1
/FIFO_top/dut/FIFO_sva_inst/wr_ack_assertion
                     FIFO_sva.sv(78)                    0          1
```

```
Directive Coverage:
    Directives                      10        10         0   100.00%

DIRECTIVE COVERAGE:
-----------------------------------------------------------------------------------------
Name                                      Design Design   Lang File(Line)      Hits Status
                                          Unit   UnitType
-----------------------------------------------------------------------------------------
/FIFO_top/dut/FIFO_sva_inst/rst_cover    FIFO_sva Verilog  SVA  FIFO_sva.sv(80)  312 Covered
/FIFO_top/dut/FIFO_sva_inst/full_cover   FIFO_sva Verilog  SVA  FIFO_sva.sv(81)  236 Covered
/FIFO_top/dut/FIFO_sva_inst/almostfull_cover
                                         FIFO_sva Verilog  SVA  FIFO_sva.sv(82)  447 Covered
/FIFO_top/dut/FIFO_sva_inst/almostempty_cover
                                         FIFO_sva Verilog  SVA  FIFO_sva.sv(83) 2444 Covered
/FIFO_top/dut/FIFO_sva_inst/empty_cover  FIFO_sva Verilog  SVA  FIFO_sva.sv(84) 2029 Covered
/FIFO_top/dut/FIFO_sva_inst/underflow_cover
                                         FIFO_sva Verilog  SVA  FIFO_sva.sv(85)  855 Covered
/FIFO_top/dut/FIFO_sva_inst/overflow_cover
                                         FIFO_sva Verilog  SVA  FIFO_sva.sv(86)  124 Covered
/FIFO_top/dut/FIFO_sva_inst/rd_ptr_cover FIFO_sva Verilog  SVA  FIFO_sva.sv(87) 3841 Covered
/FIFO_top/dut/FIFO_sva_inst/wr_ptr_cover FIFO_sva Verilog  SVA  FIFO_sva.sv(88) 4602 Covered
/FIFO_top/dut/FIFO_sva_inst/wr_ack_cover FIFO_sva Verilog  SVA  FIFO_sva.sv(89) 4744 Covered
```

```
================================================================================
=== Instance: /FIFO_top/dut
=== Design Unit: work.FIFO
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        25        25         0   100.00%

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

Branch Coverage for instance /FIFO_top/dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    39                                     10342     Count coming in to IF
    39              1                        613        if (!rst_n) begin

    45              1                       4744        else if (wr_en && count < FIFO_DEPTH) begin

    51              1                       4985        else begin

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    53                                      4985     Count coming in to IF
    53              1                        127                if (full && wr_en)

    55              1                       4858                else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    61                                      9089     Count coming in to IF
    61              1                        607        if (!rst_n) begin

    64              1                       3957        else if (rd_en && count != 0) begin

    69              1                       4525        else begin //added this else as underflow is
```

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

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

Condition Coverage for instance /FIFO_top/dut --

  File FIFO.sv
----------------Focused Condition View-------------------
Line       45 Item    1  (wr_en && (count < 8))
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       53 Item    1  (full && wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      34        34         0   100.00%

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

Statement Coverage for instance /FIFO_top/dut --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
    8                                                module FIFO(FIFO_if.DUT F_if);

    9                                                parameter FIFO_WIDTH = 16;

    10                                               parameter FIFO_DEPTH = 8;

    11                                               logic [FIFO_WIDTH-1:0] data_in;

    12                                               logic clk, rst_n, wr_en, rd_en;

    13                                               logic [FIFO_WIDTH-1:0] data_out;

    14                                               logic wr_ack, overflow;

    15                                               logic full, empty, almostfull, almostempty, underflow;

    16

    17                                               localparam max_fifo_addr = $clog2(FIFO_DEPTH);

    18

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

    20

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

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

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

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

Toggle Coverage for instance /FIFO_top/dut --

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

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

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

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

Covergroup Coverage:
    Covergroups                    1        na        na   100.00%
        Coverpoints/Crosses       16        na        na        na
            Covergroup Bins       66        66         0   100.00%
--------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------
 TYPE /Coverage_pkg/FIFO_Coverage/cvr_gp            100.00%        100          -    Covered
    covered/total bins:                                  66         66          -
    missing/total bins:                                   0         66          -
    % Hit:                                          100.00%        100          -
    Coverpoint wr_en                                100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    5013          1          -    Covered
        bin auto[1]                                    5028          1          -    Covered
    Coverpoint rd_en                                100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    5057          1          -    Covered
        bin auto[1]                                    4984          1          -    Covered
    Coverpoint overflow                             100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    9914          1          -    Covered
        bin auto[1]                                     127          1          -    Covered
    Coverpoint almostempty                          100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        bin auto[0]                                    7525          1          -    Covered
        bin auto[1]                                    2516          1          -    Covered
    Coverpoint empty                                100.00%        100          -    Covered
        covered/total bins:                               2          2          -
```

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cur
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-	
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-	
/sequence_pkg::FIFO_main_sequence::body/#ublk#50851543#14/immed__17	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/FIFO_sva_inst/rst_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/full_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/almostfull_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/almostempty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/empty_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/underflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/overflow_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/rd_ptr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/wr_ptr_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/FIFO_sva_inst/wr_ack_assertion	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	

Cover Directives

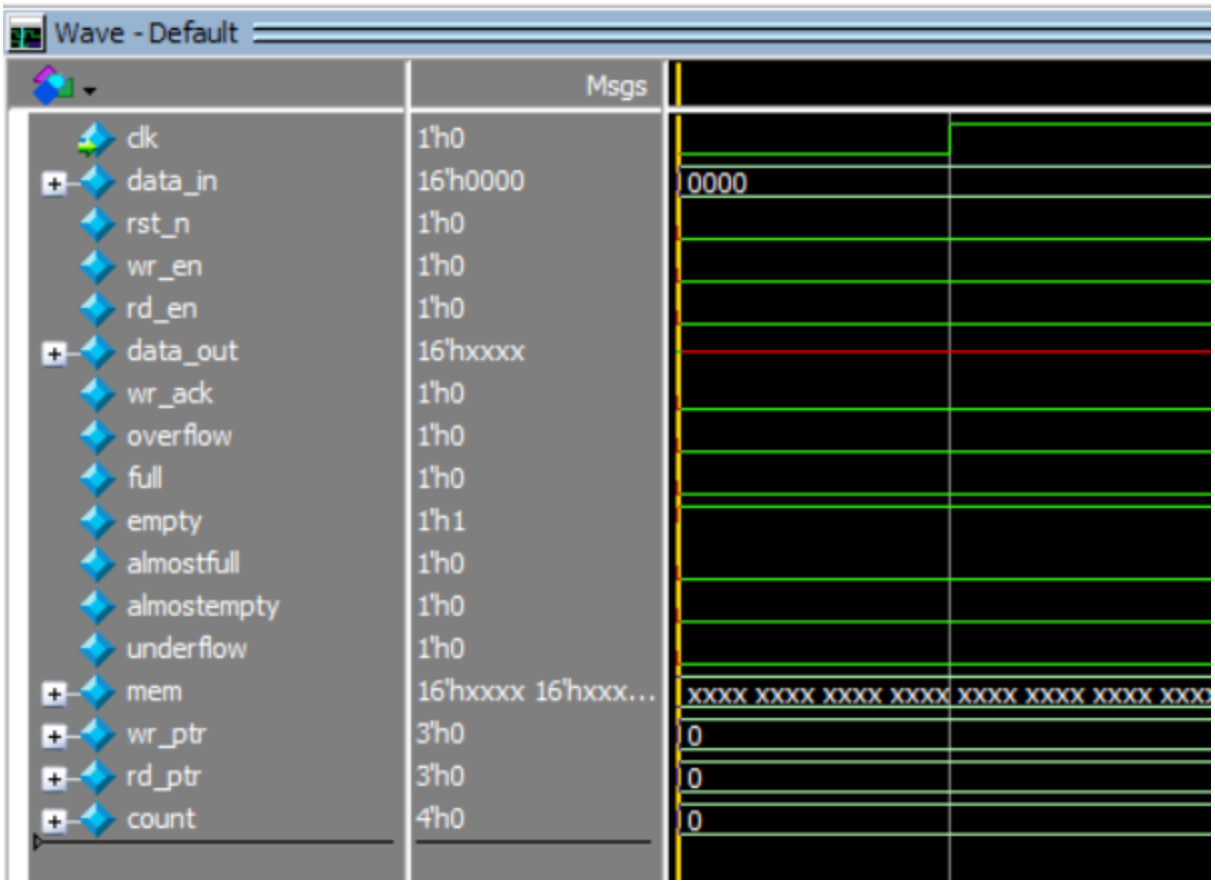
Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Th
/FIFO_top/dut/FIFO_sva_inst/rst_cover	SVA	✓	Off	312	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/full_cover	SVA	✓	Off	236	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/almostfull_cover	SVA	✓	Off	447	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/almostempty_cover	SVA	✓	Off	2444	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/empty_cover	SVA	✓	Off	2029	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/underflow_cover	SVA	✓	Off	855	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/overflow_cover	SVA	✓	Off	124	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/rd_ptr_cover	SVA	✓	Off	3841	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/wr_ptr_cover	SVA	✓	Off	4602	1	Unli...	1	100%		✓	0	0	0 ns	
/FIFO_top/dut/FIFO_sva_inst/wr_ack_cover	SVA	✓	Off	4744	1	Unli...	1	100%		✓	0	0	0 ns	

Covergroups

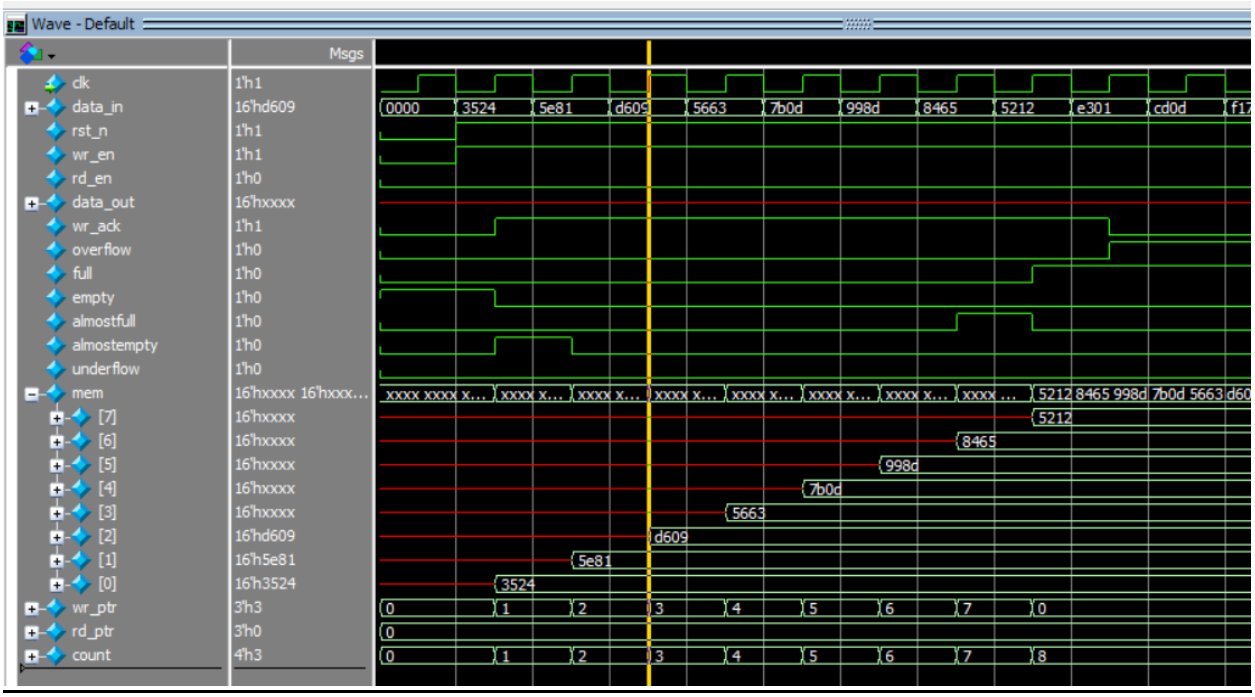
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/Coverage_pkg/FIFO_Coverage		100.00%							
TYPE cvr_gp		100.00%	100	100.00...		✓	auto(1)		
CVP cvr_gp::wr_en		100.00%	100	100.00...		✓			
CVP cvr_gp::rd_en		100.00%	100	100.00...		✓			
CVP cvr_gp::overflow		100.00%	100	100.00...		✓			
CVP cvr_gp::almostempty		100.00%	100	100.00...		✓			
CVP cvr_gp::empty		100.00%	100	100.00...		✓			
CVP cvr_gp::almostfull		100.00%	100	100.00...		✓			
CVP cvr_gp::underflow		100.00%	100	100.00...		✓			
CVP cvr_gp::full		100.00%	100	100.00...		✓			
CVP cvr_gp::wr_ack		100.00%	100	100.00...		✓			
CROSS cvr_gp::wr_ack_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::overflow_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::full_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::empty_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::almostfull_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::almostempty_cvr		100.00%	100	100.00...		✓			
CROSS cvr_gp::underflow_cvr		100.00%	100	100.00...		✓			

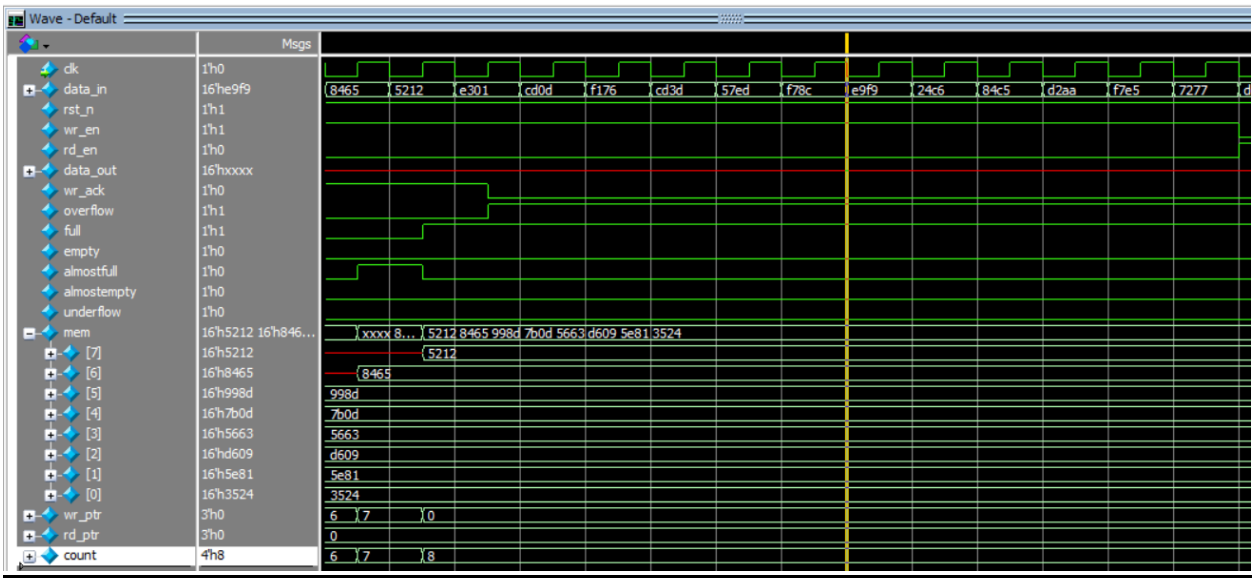
Questasim snippets:

Reset sequence:

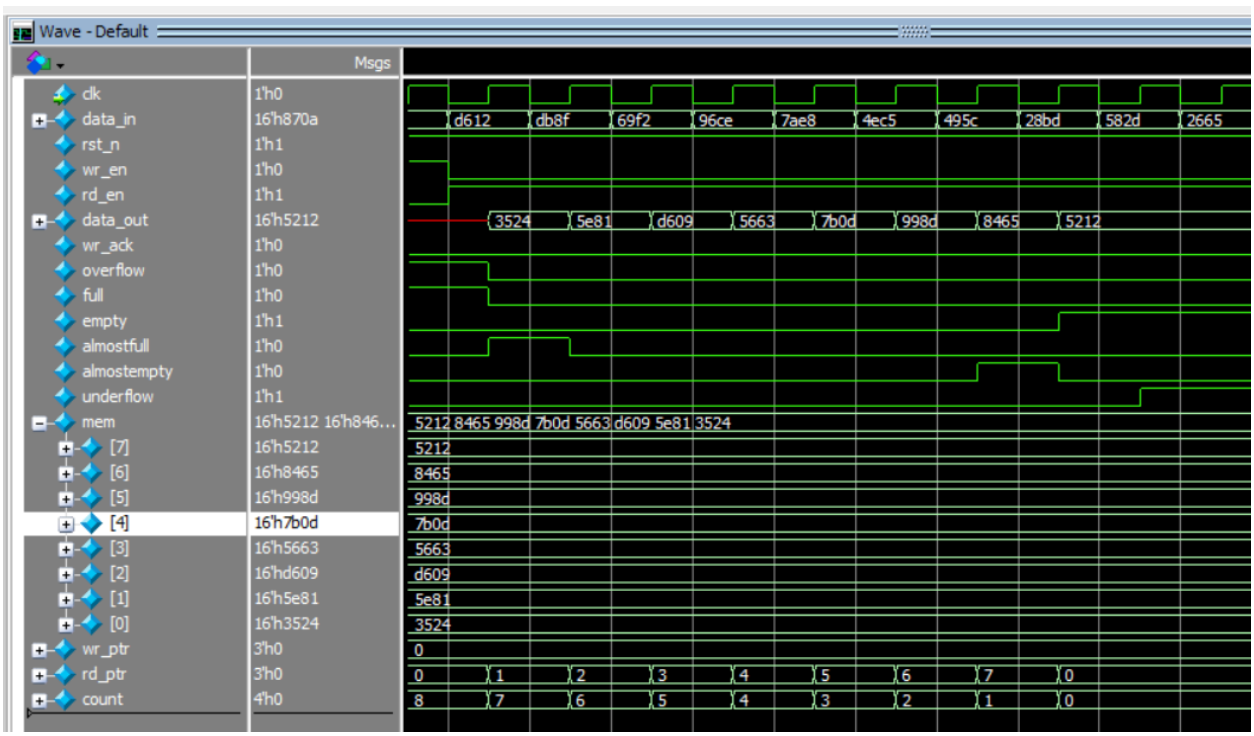


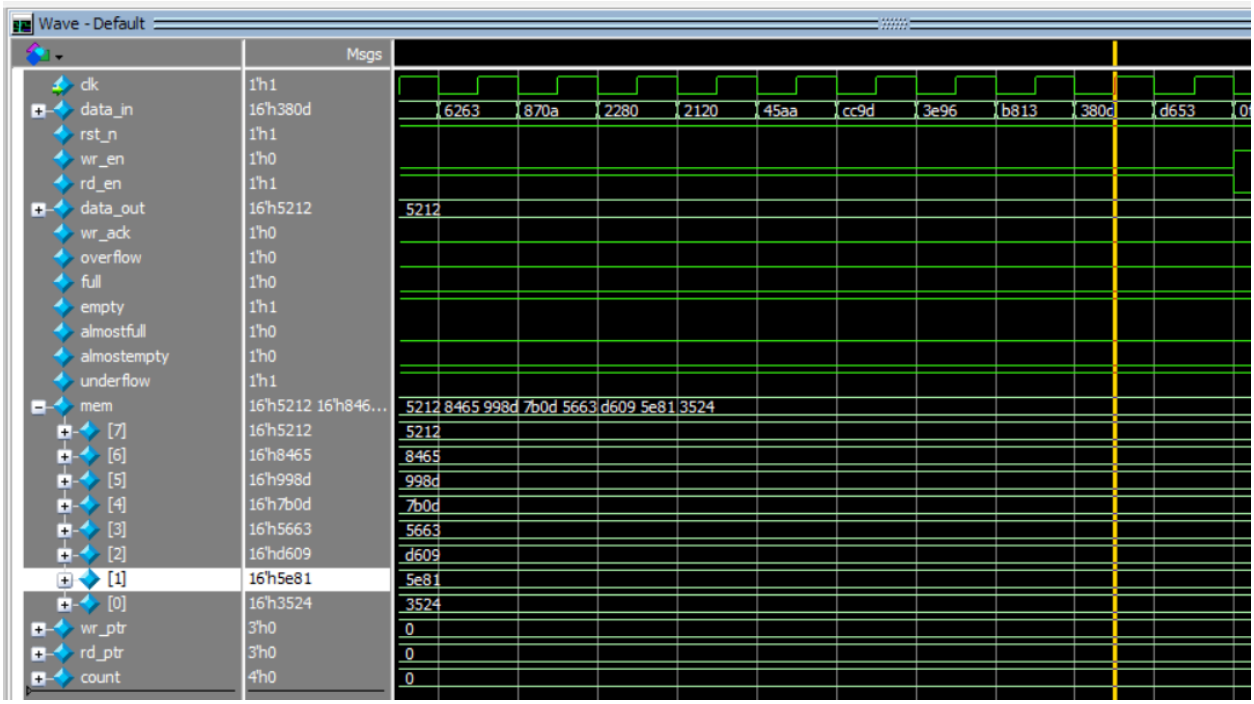
Write_sequence:



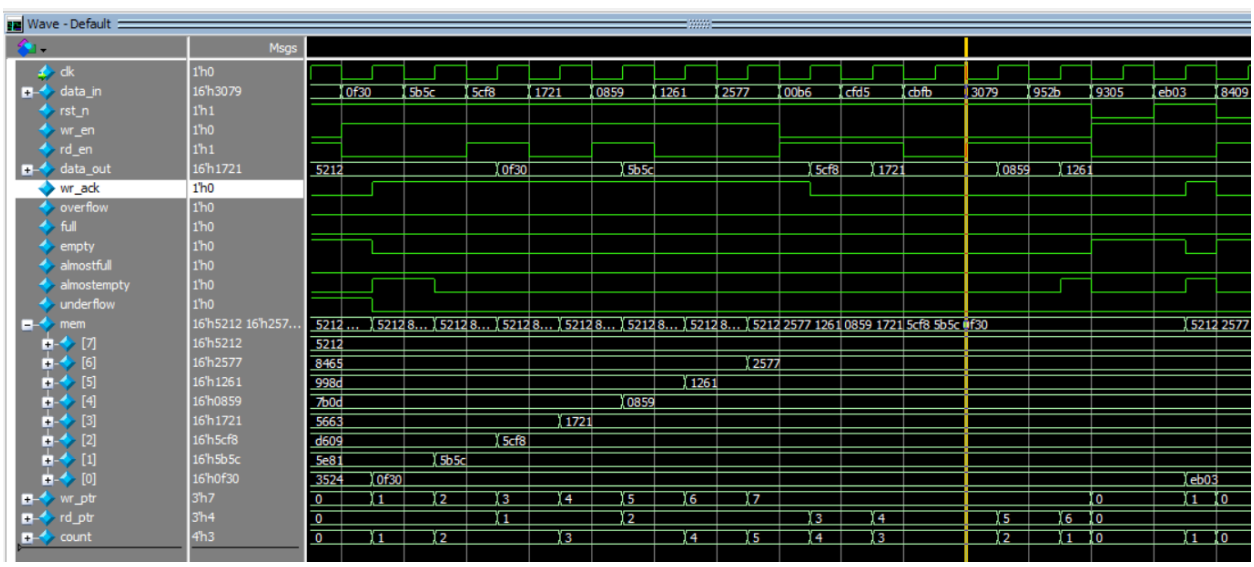


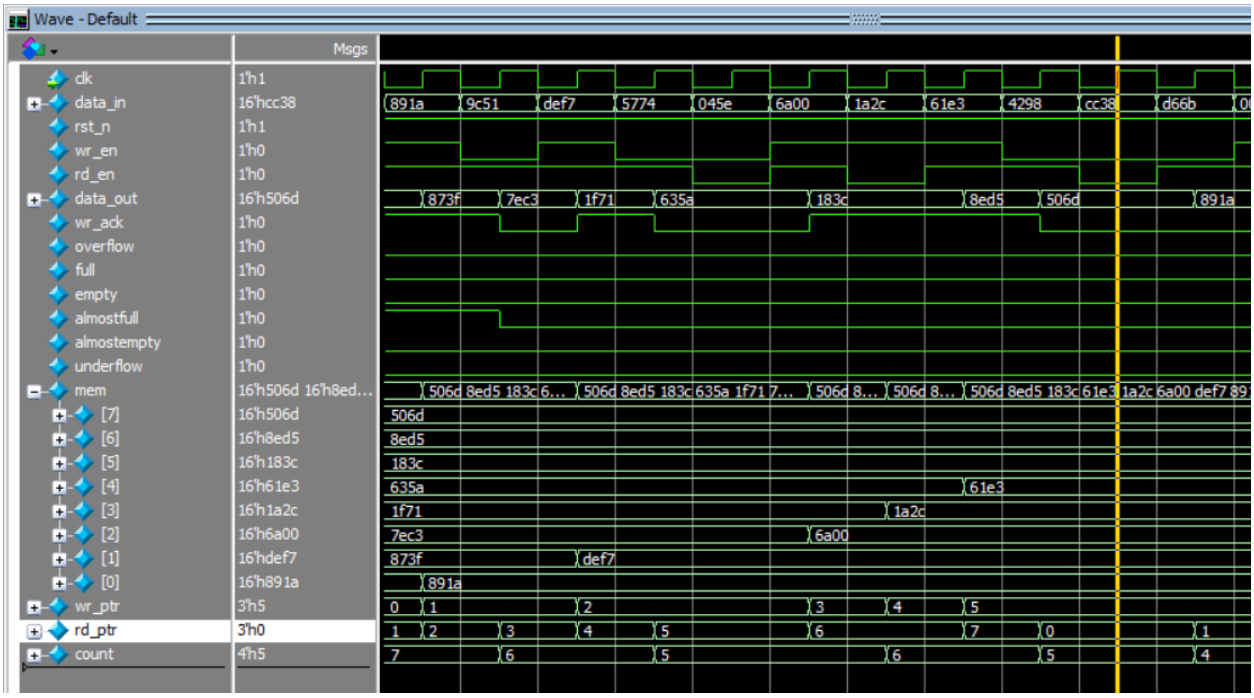
Read_sequence:

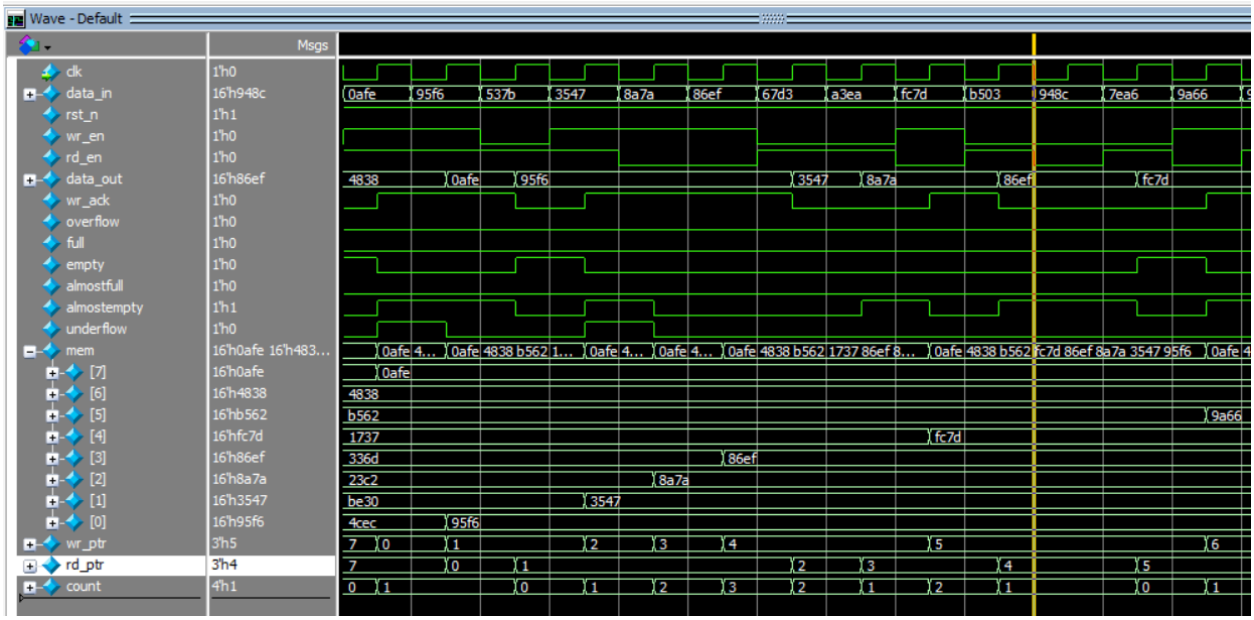


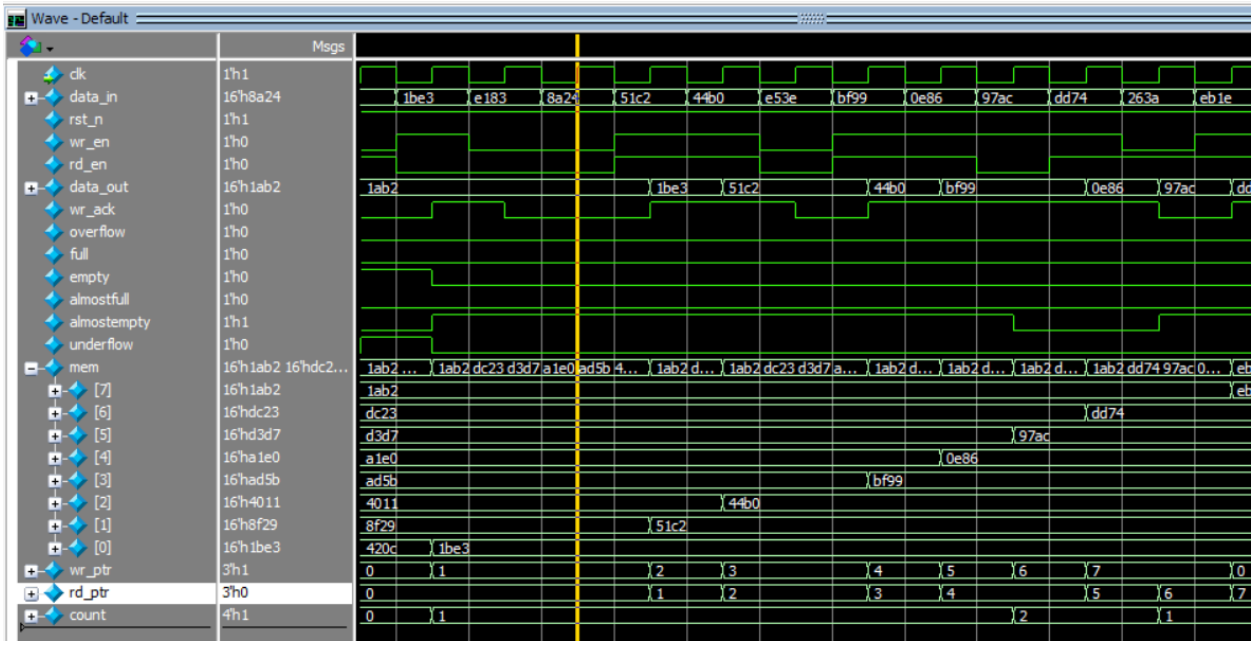


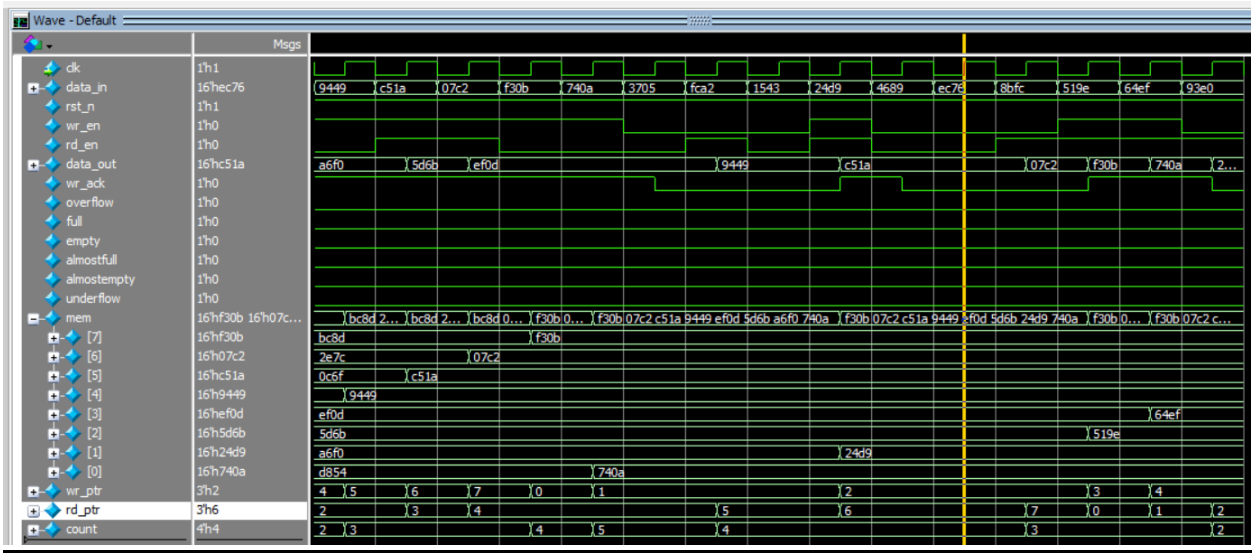
Main_sequence (total Randomization):




Wave - Default
Msgs
clk 1'h1
data_in 16'hcc38
rst_n 1'h1
wr_en 1'h0
rd_en 1'h0
data_out 16'h506d
wr_ack 1'h0
overflow 1'h0
full 1'h0
empty 1'h0
almostfull 1'h0
almostempty 1'h0
underflow 1'h0
mem 16'h506d 16'h8ed...
[7] 16'h506d
[6] 16'h8ed5
[5] 16'h183c
[4] 16'h61e3
[3] 16'h1a2c
[2] 16'h6a00
[1] 16'hdef7
[0] 16'h891a
wr_ptr 3'h5
rd_ptr 3'h0
count 4'h5


Wave - Default
Msgs
clk 1'h0
data_in 16'h948c
rst_n 1'h1
wr_en 1'h0
rd_en 1'h0
data_out 16'h86ef
wr_ack 1'h0
overflow 1'h0
full 1'h0
empty 1'h0
almostfull 1'h0
almostempty 1'h1
underflow 1'h0
mem 16'h0afe 16'h483...
[7] 16'h0afe
[6] 16'h4838
[5] 16'hb562
[4] 16'hfc7d
[3] 16'h86ef
[2] 16'h8a7a
[1] 16'h3547
[0] 16'h95f6
wr_ptr 3'h5
rd_ptr 3'h4
count 4'h1


Wave - Default
Msgs
clk 1'h1
data_in 16'h8a24
rst_n 1'h1
wr_en 1'h0
rd_en 1'h0
data_out 16'h1ab2
wr_ack 1'h0
overflow 1'h0
full 1'h0
empty 1'h0
almostfull 1'h0
almostempty 1'h1
underflow 1'h0
mem 16'h1ab2 16'hdc2...
[7] 16'h1ab2
[6] 16'hdc23
[5] 16'hd3d7
[4] 16'ha1e0
[3] 16'had5b
[2] 16'h4011
[1] 16'h8f29
[0] 16'h1be3
wr_ptr 3'h1
rd_ptr 3'h0
count 4'h1


Wave - Default
Msgs
clk 1'h1
data_in 16'hec76
rst_n 1'h1
wr_en 1'h0
rd_en 1'h0
data_out 16'hc51a
wr_ack 1'h0
overflow 1'h0
full 1'h0
empty 1'h0
almostfull 1'h0
almostempty 1'h0
underflow 1'h0
mem 16'hf30b 16'h07c...
[7] 16'hf30b
[6] 16'h07c2
[5] 16'hc51a
[4] 16'h9449
[3] 16'hef0d
[2] 16'h5d6b
[1] 16'h24d9
[0] 16'h740a
wr_ptr 3'h2
rd_ptr 3'h6
count 4'h4

Transcript:

Transcript

File Edit View Bookmarks Window Help

Transcript

```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test FIFO_test...
# UVM_INFO FIFO_test.sv(43) @ 0: uvm_test_top [run_phase] reset Asserted
# UVM_INFO FIFO_test.sv(45) @ 2: uvm_test_top [run_phase] reset Deasserted
# UVM_INFO FIFO_test.sv(47) @ 2: uvm_test_top [run_phase] full write operation started
# UVM_INFO FIFO_test.sv(49) @ 42: uvm_test_top [run_phase] full write operation  ended
# UVM_INFO FIFO_test.sv(51) @ 42: uvm_test_top [run_phase] full read operation  started
# UVM_INFO FIFO_test.sv(53) @ 82: uvm_test_top [run_phase] full read operation  ended
# UVM_INFO FIFO_test.sv(55) @ 82: uvm_test_top [run_phase] Stimulus Generation started
# UVM_INFO FIFO_test.sv(57) @ 20082: uvm_test_top [run_phase] Stimulus Generation ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20082: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_Scoreboard.sv(153) @ 20082: uvm_test_top.env.sb [report_phase] Total successful transactions: 10041
# UVM_INFO FIFO_Scoreboard.sv(154) @ 20082: 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    : F:/questasim/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 20082 ns  Iteration: 61  Instance: /FIFO_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at F:/questasim/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```