Synchronous FIFO

UVM Project

Table of contents

Specs

Parameters

- FIFO_WIDTH: DATA in/out and memory word width (default: 16)
- FIFO_DEPTH: Memory depth (default: 8)

Ports

Port	Direction	Function
data_in	Input	Write Data: The input data bus used when writing the FIFO.
wr_en		Write Enable: If the FIFO is not full, asserting this signal causes data (on data_in) to be written into the FIFO
rd_en		Read Enable: If the FIFO is not empty, asserting this signal causes data (on data_out) to be read from the FIFO
clk		Clock signal
rst_n		Active low asynchronous reset
data_out	Output	Read Data: The sequential output data bus used when reading from the FIFO.
full		Full Flag: When asserted, this combinational output signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
almostfull		Almost Full: When asserted, this combinational output signal indicates that only one more write can be performed before the FIFO is full.
empty		Empty Flag: When asserted, this combinational output signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.
almostempty		Almost Empty: When asserted, this output combinational signal indicates that only one more read can be performed before the FIFO goes to empty.
overflow		Overflow: This sequential output signal indicates that a write request (wr_en) was rejected because the FIFO is full. Overflowing the FIFO is not destructive to the contents of the FIFO.
underflow		Underflow: This sequential output signal Indicates that the read request (rd_en) was rejected because the FIFO is empty. Under flowing the FIFO is not destructive to the FIFO.
wr_ack		Write Acknowledge: This sequential output signal indicates that a write request (wr_en) has succeeded.

Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted check the all signals is zero	Directed at the start of simulation	-	assertion to make sure output is correct
FIFO_2	When the write enable is active and fifo isn't full check that wr_ack=1	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all valuesof wr_ack. Cross wr_en,rd_en,wr_ack. for all combinations	assertion to make sure output is correct .. Assert count up and wr_ptr
FIFO_3	When the count=fifo_depth check that full flag raised	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of full. Cross wr_en,rd_en,full for all combinations	assertion to make sure output is correct
FIFO_4	When the count=fifo_depth-1 check that almostfull flag raised	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of almostfull. Cross wr_en,rd_en,almostfull for all combinations	assertion to make sure output is correct
FIFO_5	When the count=0 check that empty raised	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of empty. Cross wr_en,rd_en,empty for all combinations	assertion to make sure output is correct
	When the count=1 check that almostempty raised	Randomization under constraints that reset ia	cover all values of almostempty.Cross	assertion to make sure output is correct
FIFO_6	When the count=1 check that almostempty raised	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of almostempty.Cross wr_en,rd_en,almostempty for all combinations	assertion to make sure output is correct
FIFO_7	When the fifo is full and write operation is to be done raise overlow	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of overflow. Cross wr_en,rd_en,overflow for all combinations	assertion to make sure output is correct
FIFO_8	When the fifo is empty and read operation is to be done raise underflow	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all values of underflow. Cross wr_en,rd_en,underflow for all combinations	assertion to make sure output is correct
FIFO_9	When the write enable,read enable both are active and fifo is full check that wr_ack=0	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all valuesof wr_ack. Cross wr_en,rd_en,wr_ack. for all combinations	assertion to make sure output is correct .. Assert count down and rd_ptr
FIFO_10	When the write enable,read enable both are active and fifo isn't full or empty check that wr_ack=1	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time	cover all valuesof wr_ack. Cross wr_en,rd_en,wr_ack. for all combinations	assertion to make sure output is correct .. Assert count no_change

			active 30% of time		
2	FIFO_11	when reset is deasserted check for the dataout output	Randomization under constraints that reset ia deasserted most of time ,write enable is active 70% of time and read enable is active 30% of time		golden model to check that output is correct
3	FIFO_12	when reset is deasserted , write enable is active and read enable is inactive check that only write operations happens until the FIFO overflows	Randomization		golden model to check that output is correct
4	FIFO_13	when reset is deasserted , write enable is inactive and read enable is active check that only read operations happens until the FIFO underflows	Randomization		golden model to check that output is correct

Bugs found

File Edit Format View Help

```
1- in the assignment of the " almostfull " signal it sould be when count = fifo depth-1 .
2- the under flow signal is a sequential one but is assigned cominationaly .
3- case of both wr_en =1 and rd_en =1 was not handeled .
4- when reset ia active wr_ack ,overflow and underflow should be zero .
5- count ,read pointer and write pointer should be initialized from zero.
```

Dofile

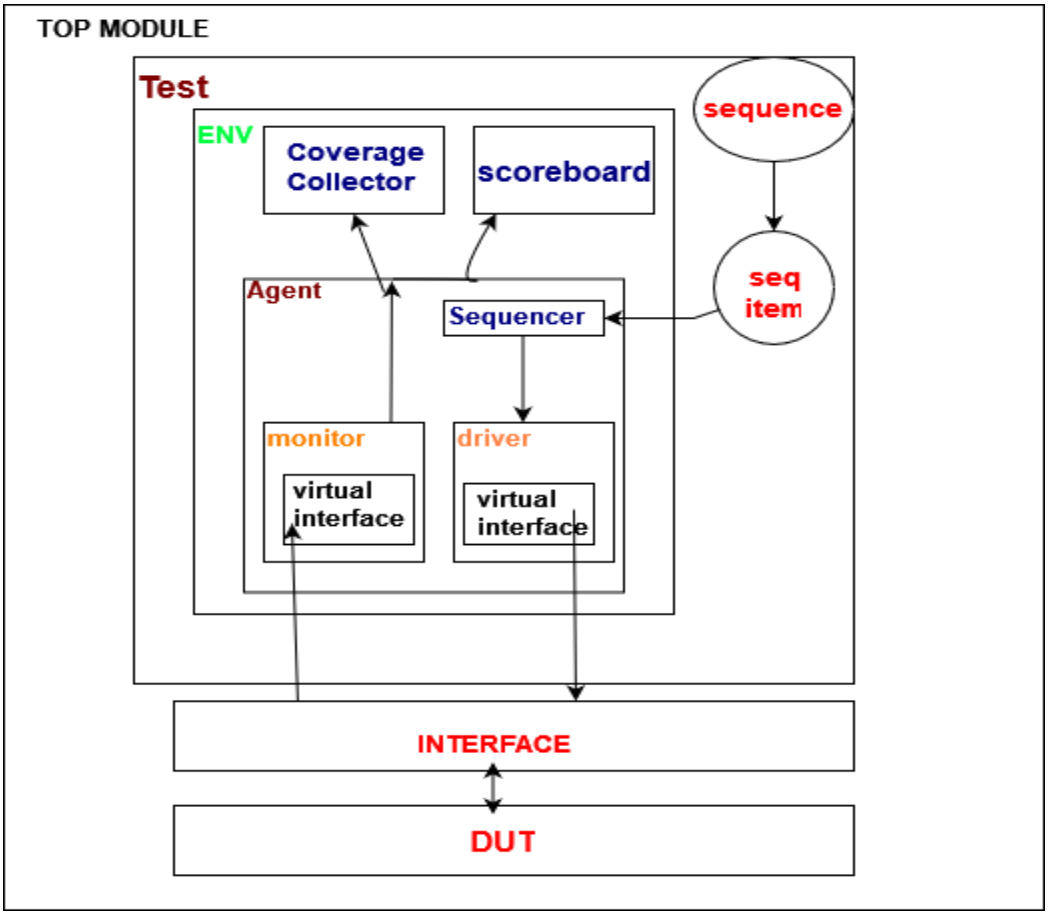
File Edit Format View Help

```
vlib work
vlog -f src_files.txt +cover -covercells
vsim -voptargs=+acc work.FIFO_top -cover
add wave -position insertpoint sim:/FIFO_top/FIFO_if/*
coverage save FIFO_top.ucdb -onexit
run -all
//quit -sim
//vcover report FIFO_top.ucdb -details -all -output coverage_rpt.txt
```

File Edit Format View Help

```
FIFO.sv
FIFO_interface.sv
FIFO_config.sv
FIFO_sequnece_item.sv
FIFO_driver.sv
FIFO_monitor.sv
FIFO_sequencer.sv
FIFO_agent.sv
FIFO_coverage.sv
FIFO_scoreboard.sv
FIFO_env.sv
FIFO_reset_sequence.sv
FIFO_main_sequence.sv
FIFO_test.sv
FIFO_assertions.sv
FIFO_top.sv
```

UVM Hierarchy



Top module

This is where the execution starts .

This block is responsible for generating clock , instantiating the dut , binding assertions , setting the virtual interface into the configuration database and running the test .

UVM TEST

Builds the environment and sequences , getting the virtual interface from database , setting the configuration object and running sequences to the sequencer .

UVM SEQUENCE

Sequences represents the stimulus we generate to test the dut through generating several sequence items .

UVM SEQUENCE ITEMS

These are the data we use to drive the dut and used to generate random sequence stimulus.

UVM SEQUENCER

This is a fifo that registers the generated sequence items to push them to the driver

UVM ENVIRONMENT

Builds and connects the agents and analysis components (coverage collector and scoreboard)

UVM DRIVER

Pulls the registered items from the sequencer and then assigning them to the virtual interface that talks directly with the dut

UVM AGENT

Builds the monitor , driver and sequencer .

Connects the driver to the sequencer .

Gets the config object from data base to assign its virtual interface to ones of driver and monitor .

UVM ANALYSIS COMPONENTS

Scoreboard : receives a sequence item from the monitor and compares it with the reference model to check functionality .

Coverage collector : receives a sequence item from the monitor and samples the data for functional coverage .

Snippets

Sequence item

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


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

function string convert2string ();
  return $sformatf("%s reset =0b%0b ,datain =0h%0h,wr_en =0b%0b,rd_en =0b%0b ,dataout =0h%0h,wr_ack =0b%0b,overflow =0b%0b,
  underflow =0b%0b ,full =0b%0b,empty =0b%0b,almostfull =0b%0b,almostempty =0b%0b",
  super.convert2string(),rst_n,data_in,wr_en,rd_en,data_out,wr_ack,overflow,underflow,full,empty,almostfull,almostempty);
endfunction
function string convert2string_stimulus ();
  return $sformatf(" reset =0b%0b ,datain =0h%0h,wr_en =0b%0b,rd_en =0b%0b",rst_n,data_in,wr_en,rd_en);
endfunction

constraint c1{rst_n dist{0:/5,1:/95};}
constraint c2{wr_en dist{0:/30 ,1:/70 };}
constraint c3{rd_en dist{0:/70,1:/30};}
```

Sequence

FIFO_main_sequence.sv

```
1   package FIFO_main_sequence_pck ;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import FIFO_sequence_item_pck::*;
5   class FIFO_write_read extends uvm_sequence #(FIFO_sequence_item);
6    `uvm_object_utils(FIFO_write_read)
7   FIFO_sequence_item seq_item;
8
9   function new (string name = "FIFO_write_read");
10    super.new(name);
11  endfunction
12
13  task body;
14  repeat(5000) begin
15  seq_item = FIFO_sequence_item::type_id::create("seq_item");
16  start_item(seq_item);
17   assert(seq_item.randomize());
18  finish_item(seq_item); end
19  endtask
20  endclass
21
22  class FIFO_write_only extends uvm_sequence #(FIFO_sequence_item);
23   `uvm_object_utils(FIFO_write_only)
24  FIFO_sequence_item seq_item;
25
26  function new (string name = "FIFO_write_only");
27    super.new(name);
28  endfunction
```

```
task body;
repeat(500) begin
seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.wr_en=1;
seq_item.rd_en=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

class FIFO_read_only extends uvm_sequence #(FIFO_sequence_item);
 `uvm_object_utils(FIFO_read_only)
FIFO_sequence_item seq_item;

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

task body;
repeat(500) begin
seq_item = FIFO_sequence_item::type_id::create("seq_item");
start_item(seq_item);
seq_item.wr_en=0;
seq_item.rd_en=1;
seq_item.wr_en.rand_mode(0);
seq_item.rd_en.rand_mode(0);
 assert(seq_item.randomize());
finish_item(seq_item); end
endtask
```

Reset

FIFO_reset_sequence.sv

```
1   package FIFO_reset_sequence_pck;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import FIFO_sequence_item_pck::*;
5   class FIFO_reset_sequence extends uvm_sequence #(FIFO_sequence_item);
6     `uvm_object_utils(FIFO_reset_sequence)
7   FIFO_sequence_item seq_item;
8
9   function new (string name = "FIFO_reset_sequence");
10    super.new(name);
11  endfunction
12
13  task body;
14  seq_item = FIFO_sequence_item::type_id::create("seq_item");
15  start_item(seq_item);
16  seq_item.rst_n=1'b0;
17  finish_item(seq_item);
18  endtask
19
20  endclass
21  endpackage
```

Sequencer

FIFO_sequencer.sv

```
1   package FIFO_sequencer_pck ;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import FIFO_sequence_item_pck::*;
5   class FIFO_sequencer extends uvm_sequencer#(FIFO_sequence_item);
6     `uvm_component_utils(FIFO_sequencer)
7
8   function new (string name = "FIFO_sequencer",uvm_component parent =null);
9     super.new(name,parent);
10  endfunction
11
12  endclass
13  endpackage
14
```

Driver

FIFO_driver.sv

```
1   package FIFO_driver_pck;
2
3   import FIFO_config_pck::*;
4   import FIFO_sequence_item_pck ::*;
5   import uvm_pkg::*;
6   `include "uvm_macros.svh"
7   class FIFO_driver extends uvm_driver#(FIFO_sequence_item);
8     `uvm_component_utils(FIFO_driver)
9   virtual FIFO_interface FIFO_vif;
10  FIFO_sequence_item stim_seq_item;
11
12  function new (string name = "FIFO_driver",uvm_component parent =null);
13    super.new(name,parent);
14  endfunction
15
16  task run_phase(uvm_phase phase);
17  super.run_phase(phase);
18  forever begin
19  stim_seq_item = FIFO_sequence_item::type_id::create("stim_seq_item");
20  seq_item_port.get_next_item(stim_seq_item);
21  FIFO_vif.rst_n=stim_seq_item.rst_n;
22  FIFO_vif.wr_en=stim_seq_item.wr_en;
23  FIFO_vif.rd_en=stim_seq_item.rd_en;
24  FIFO_vif.data_in=stim_seq_item.data_in;
25  @(negedge FIFO_vif.clk);
26  seq_item_port.item_done();
27  `uvm_info("run_phase",stim_seq_item.convert2string_stimulus (),UVM_HIGH)
28  end
29  endtask
30  endclass
31  endpackage
```

Configuration

FIFO_config.sv

```
1   package FIFO_config_pck;
2
3   import uvm_pkg::*;
4   `include "uvm_macros.svh"
5   class FIFO_config extends uvm_object;
6
7    `uvm_object_utils(FIFO_config)
8   virtual FIFO_interface FIFO_vif;
9   function new (string name = "FIFO_config");
10    super.new(name);
11  endfunction
12  endclass
13  endpackage
```

Monitor

FIFO_monitor.sv

```
1   package FIFO_monitor_pck ;
2   import uvm_pkg::*;
3   import FIFO_sequence_item_pck::*;
4   `include "uvm_macros.svh"
5
6   class FIFO_monitor extends uvm_monitor;
7   `uvm_component_utils(FIFO_monitor)
8
9    virtual FIFO_interface FIFO_vif;
10  FIFO_sequence_item rsp_seq_item;
11  uvm_analysis_port #(FIFO_sequence_item) mon_ap;
12
13  function new (string name = "FIFO_monitor",uvm_component parent =null);
14    super.new(name,parent);
15  endfunction
16
17  function void build_phase (uvm_phase phase);
18   super.build_phase(phase);
19  mon_ap =new("mon_ap",this);
20  endfunction
```

```
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
rsp_seq_item = FIFO_sequence_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.rd_en=FIFO_vif.rd_en;
rsp_seq_item.data_in=FIFO_vif.data_in;
rsp_seq_item.data_out=FIFO_vif.data_out;
rsp_seq_item.overflow=FIFO_vif.overflow;
rsp_seq_item.underflow=FIFO_vif.underflow;
rsp_seq_item.full=FIFO_vif.full;
rsp_seq_item.almostfull=FIFO_vif.almostfull;
rsp_seq_item.empty=FIFO_vif.empty;
rsp_seq_item.almostempty=FIFO_vif.almostempty;
rsp_seq_item.wr_ack=FIFO_vif.wr_ack;
mon_ap.write(rsp_seq_item);
`uvm_info("run_phase",rsp_seq_item.convert2string (),UVM_HIGH)
end

endtask

endclass
endpackage
```

Interface

FIFO_interface.sv

```
1   import uvm_pkg::*;
2   `include "uvm_macros.svh"
3
4   interface FIFO_interface(clk) ;
5   ////ports////
6   parameter FIFO_WIDTH = 16;
7   parameter FIFO_DEPTH = 8;
8   input clk;
9   logic [FIFO_WIDTH-1:0] data_in;
10  bit  rst_n, wr_en, rd_en;
11  logic [FIFO_WIDTH-1:0] data_out;
12  bit wr_ack, overflow;
13  bit full, empty, almostfull, almostempty, underflow;
14
15  ////directions////
16  modport dut (input clk,data_in,rst_n, wr_en, rd_en,
17               output data_out, overflow,wr_ack,full, empty, almostfull, almostempty, underflow);
18  modport monitor (input clk,data_in,rst_n, wr_en, rd_en,
19               data_out, overflow,wr_ack,full, empty, almostfull, almostempty, underflow);
20  endinterface
```

Environment

FIFO_env.sv

```
1   package FIFO_env_pck;
2   import  FIFO_driver_pck::*;
3   import  FIFO_agent_pck::*;
4   import  FIFO_coverage_pck::*;
5   import  FIFO_scoreboard_pck::*;
6   import  FIFO_sequence_item_pck::*;
7   import  FIFO_sequencer_pck::*;
8   import uvm_pkg::*;
9   `include "uvm_macros.svh"
10  class FIFO_env extends uvm_env;
11  `uvm_component_utils(FIFO_env)
12   FIFO_agent agt;
13   FIFO_coverage cov;
14   FIFO_scoreboard sb;
15  function new (string name = "FIFO_env",uvm_component parent =null);
16    super.new(name,parent);
17  endfunction
18
19  function void build_phase (uvm_phase phase);
20   super.build_phase(phase);
21
22   agt = FIFO_agent::type_id::create("agt",this);
23    cov = FIFO_coverage::type_id::create("cov",this);
24     sb = FIFO_scoreboard::type_id::create("sb",this);
25  endfunction
26
27  function void connect_phase (uvm_phase phase);
28   agt.agt_ap.connect(sb.sb_export);
29   agt.agt_ap.connect(cov.cov_export);
30  endfunction
31
32  endclass
```

Agent

FIFO_agent.sv

```
1   package FIFO_agent_pck;
2
3   import uvm_pkg::*;
4   import FIFO_sequencer_pck::*;
5   import FIFO_sequence_item_pck::*;
6   import FIFO_driver_pck::*;
7   import FIFO_monitor_pck::*;
8   import FIFO_config_pck::*;
9   `include "uvm_macros.svh"
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 FIFO_cfg;
17  uvm_analysis_port #(FIFO_sequence_item) agt_ap;
18
19  function new (string name = "FIFO_agent",uvm_component parent =null);
20    super.new(name,parent);
21  endfunction
22
```

```
function void build_phase (uvm_phase phase);
 super.build_phase(phase);
 if(!uvm_config_db #(FIFO_config)::get(this,"","CFG",FIFO_cfg)) begin
   `uvm_fatal("bulid_phase","Driver - Unable to get configuration object") ;end

 drv = FIFO_driver::type_id::create("drv",this);
 sqr = FIFO_sequencer::type_id::create("sqr",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);
 mon.mon_ap.connect(agt_ap);
endfunction

endclass
endpackage
```

Test

```
package FIFO_test_pck;
import FIFO_env_pck::*;
import FIFO_config_pck::*;
import FIFO_main_sequence_pck::*;
import FIFO_reset_sequence_pck::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class FIFO_test extends uvm_test;

`uvm_component_utils(FIFO_test);
 virtual FIFO_interface FIFO_vif;
FIFO_config FIFO_cfg;
FIFO_reset_sequence reset_seq;
FIFO_write_read write_read;
FIFO_write_only write_only;
FIFO_read_only read_only;
FIFO_env env;
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::type_id::create("FIFO_cfg",this);
 reset_seq = FIFO_reset_sequence::type_id::create("reset_seq",this);
 write_read = FIFO_write_read::type_id::create("write_read",this);
 write_only = FIFO_write_only::type_id::create("write_only",this);
 read_only = FIFO_read_only::type_id::create("read_only",this);
```

```
function void build_phase (uvm_phase phase);
 read_only = FIFO_read_only::type_id::create("read_only",this);

  if(!uvm_config_db #(virtual FIFO_interface)::get(this,"","FIFO_If",FIFO_cfg.FIFO_vif)) begin
   `uvm_fatal("bulid_phase","Test - Unable to get the virtual interface of the shift_reg from the uvm_config_db") ;end

 uvm_config_db #(FIFO_config)::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_MEDIUM);
  reset_seq.start(env.agt.sqr);
 `uvm_info("run_phase","reset deasserted",UVM_MEDIUM);
`uvm_info("run_phase","stimulas generation for write and read started",UVM_MEDIUM);
  write_read.start(env.agt.sqr);
 `uvm_info("run_phase","stimulas generation for write and read ended",UVM_MEDIUM);
 `uvm_info("run_phase","stimulas generation for write only started",UVM_MEDIUM);
  write_only.start(env.agt.sqr);
 `uvm_info("run_phase","stimulas generation for write only ended",UVM_MEDIUM);
 `uvm_info("run_phase","stimulas generation for read only started",UVM_MEDIUM);
  read_only.start(env.agt.sqr);
 `uvm_info("run_phase","stimulas generation for read only endedd",UVM_MEDIUM);
 phase.drop_objection(this);
endtask

endclass
endpackage
```

Top

FIFO_top.sv

```
1
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import FIFO_test_pck::*;
5
6   module FIFO_top();
7   bit clk;
8
9   initial begin
10      clk=1'b1;
11  forever begin
12  #5 clk=~clk;
13  end
14  end
15
16  FIFO_interface FIFO_if (clk);
17  FIFO DUT (FIFO_if);
18  bind FIFO FIFO_assertions #(FIFO_if.FIFO_WIDTH,FIFO_if.FIFO_DEPTH) INIT(FIFO_if.clk,FIFO_if.data_in,FIFO_if.rst_n,
19   FIFO_if.wr_en, FIFO_if.rd_en,FIFO_if.data_out,FIFO_if.wr_ack,FIFO_if.overflow,FIFO_if.full, FIFO_if.empty,
20   FIFO_if.almostfull, FIFO_if.almostempty, FIFO_if.underflow);
21
22  initial begin
23  uvm_config_db #(virtual FIFO_interface)::set(null,"uvm_test_top","FIFO_If",FIFO_if);
24  run_test("FIFO_test");
25  end
26  endmodule
```

Scoreboard

FIFO_scoreboard.sv

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

class FIFO_scoreboard extends uvm_component;
 `uvm_component_utils(FIFO_scoreboard)
 uvm_analysis_export #(FIFO_sequence_item) sb_export;
 uvm_tlm_analysis_fifo #(FIFO_sequence_item) sb_fifo;
 FIFO_sequence_item seq_item_sb;


int error_count=0;
int correct_count=0;
logic [15:0] data_out_ref;
logic [15:0] mem [$];

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
```

```
task run_phase(uvm_phase phase);
 super.run_phase(phase);
 forever begin
    sb_fifo.get(seq_item_sb);
    ref_model(seq_item_sb);
    if(seq_item_sb.data_out != data_out_ref) begin
        `uvm_error("run_phase",$sformatf("comparison failed,transaction received by the dut %s while the refeence output
         is 0h%0h",seq_item_sb.convert2string(), data_out_ref));
        error_count++; end
    else begin
        `uvm_info("run_phase",$sformatf("correct dataout %s ",seq_item_sb.convert2string()),UVM_HIGH);
        correct_count++;
    end
 end
endtask
```

```
task ref_model( FIFO_sequence_item seq_item_chk);
if(seq_item_chk.rst_n) begin
    if(seq_item_chk.wr_en &&!seq_item_chk.rd_en && count!=8) begin
        mem.push_front(seq_item_chk.data_in);count++;
    end
    else if(!seq_item_chk.wr_en && seq_item_chk.rd_en && count!=0) begin
        data_out_ref=mem.pop_back;count--;

    end

    else if(seq_item_chk.wr_en && seq_item_chk.rd_en && count!=8 && count!=0) begin
        mem.push_front(seq_item_chk.data_in);
        data_out_ref=mem.pop_back;


    end
    else if (seq_item_chk.wr_en && seq_item_chk.rd_en && count==8) begin
        data_out_ref=mem.pop_back;count--;

    end
     else if (seq_item_chk.wr_en && seq_item_chk.rd_en && count==0) begin
            mem.push_front(seq_item_chk.data_in);count++;
    end
end
else if(!seq_item_chk.rst_n) begin
    mem.delete;count=0;
end
```

Coverage collector

FIFO_coverage.sv

```
1   package FIFO_coverage_pck;
2   import FIFO_sequence_item_pck::*;
3   import FIFO_config_pck::*;
4   import uvm_pkg::*;
5   `include "uvm_macros.svh"
6
7   class FIFO_coverage extends uvm_component;
8    `uvm_component_utils(FIFO_coverage)
9    uvm_analysis_export #(FIFO_sequence_item) cov_export;
10   uvm_tlm_analysis_fifo #(FIFO_sequence_item) cov_fifo;
11   FIFO_sequence_item seq_item_cov;
12
13  covergroup cvr_grp ;
14  wr_ack_cp:coverpoint seq_item_cov.wr_ack
15  {
16      bins wr_ack_0={0};
17      bins wr_ack_1={1};
18  }
19  overflow_cp:coverpoint seq_item_cov.overflow
20  {
21      bins overflow_0={0};
22      bins overflow_1={1};
23  }
24  underflow_cp:coverpoint seq_item_cov.underflow
25  {
26      bins underflow_0={0};
27      bins underflow_1={1};
28  }
29  full_cp:coverpoint seq_item_cov.full
30  {
31      bins full_0={0};
32      bins full_1={1};
```

```
almostfull_cp:coverpoint seq_item_cov.almostfull
{
    bins almostfull_0={0};
    bins almostfull_1={1};
}
empty_cp:coverpoint seq_item_cov.empty
{
    bins empty_0={0};
    bins empty_1={1};
}
almostempty_cp:coverpoint seq_item_cov.almostempty
{
    bins almostempty_0={0};
    bins almostempty_1={1};
}
wr_en_cp:coverpoint seq_item_cov.wr_en
{
    bins wr_en_0={0};
    bins wr_en_1={1};
}
rd_en_cp:coverpoint seq_item_cov.rd_en
{
    bins rd_en_0={0};
    bins rd_en_1={1};
}
wr_ack: cross wr_en_cp, rd_en_cp,wr_ack_cp
{
    illegal_bins wr_ack_illegal_1 = binsof (wr_en_cp.wr_en_0)&&binsof (rd_en_cp.rd_en_0) && binsof (wr_ack_cp.wr_ack_1);
    illegal_bins wr_ack_illegal_2 = binsof (wr_en_cp.wr_en_0)&&binsof (rd_en_cp.rd_en_1) && binsof (wr_ack_cp.wr_ack_1) ;

}
```

```
}
overflow: cross wr_en_cp, rd_en_cp,overflow_cp;
underflow: cross wr_en_cp, rd_en_cp,underflow_cp
{
    illegal_bins underflow_illegal_1 = binsof (wr_en_cp.wr_en_0)&&binsof (rd_en_cp.rd_en_0) && binsof (underflow_cp.underflow_1);
}
full: cross wr_en_cp, rd_en_cp,full_cp
{
    illegal_bins full_illegal_1 = binsof (wr_en_cp.wr_en_0)&&binsof (rd_en_cp.rd_en_1) && binsof (full_cp.full_1);
    illegal_bins full_illegal_2 = binsof (wr_en_cp.wr_en_1)&&binsof (rd_en_cp.rd_en_1) && binsof (full_cp.full_1) ;

}
empty: cross wr_en_cp, rd_en_cp,empty_cp;
almostfull: cross wr_en_cp, rd_en_cp,almostfull_cp;
almostempty: cross wr_en_cp, rd_en_cp,almostempty_cp;
endgroup

 function new (string name = "FIFO_coverage",uvm_component parent =null);
  super.new(name,parent);
  cvr_grp=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
```

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

Assertions

FIFO_assertions.sv

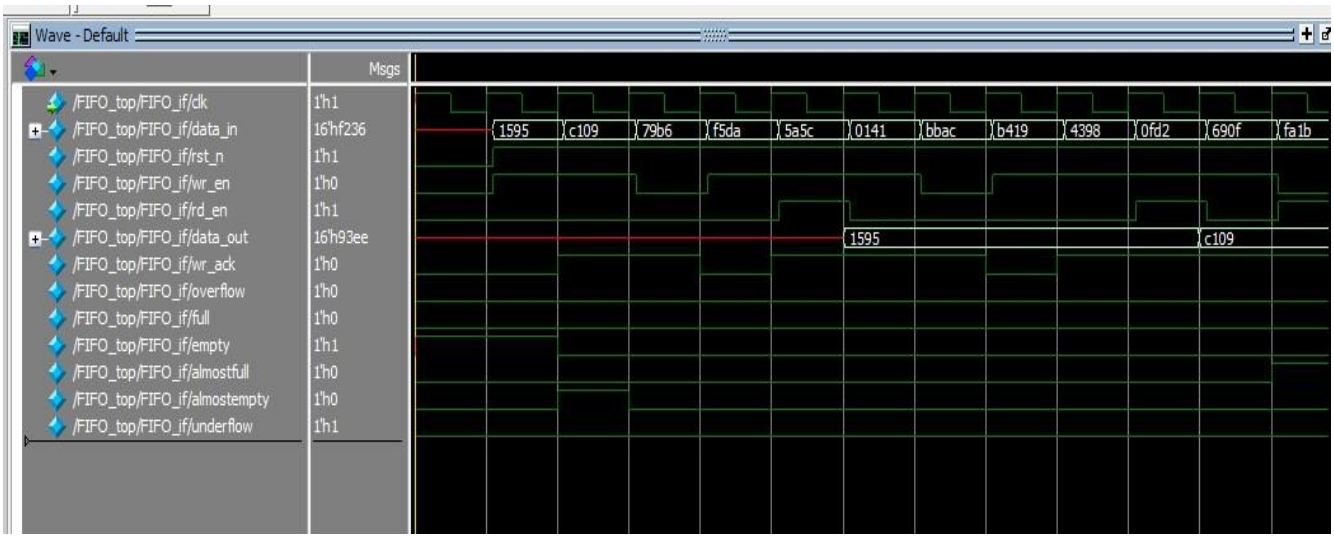
```
1
2   module FIFO_assertions (clk,data_in,rst_n, wr_en, rd_en,data_out,wr_ack, overflow,full, empty, almostfull, almostempty, und
3
4   parameter FIFO_WIDTH = 16;
5   parameter FIFO_DEPTH = 8;
6   input clk;
7   input [FIFO_WIDTH-1:0] data_in;
8   input  rst_n, wr_en, rd_en;
9   input [FIFO_WIDTH-1:0] data_out;
10  input wr_ack, overflow;
11  input full, empty, almostfull, almostempty, underflow;
12
13  always_comb begin : reset_check
14      if(!rst_n)
15      reset: assert final (DUT.count==0 && DUT.rd_ptr==0 && DUT.wr_ptr==0 && empty && !full && !almostfull && !almostempty);
16      reset_cover:cover (DUT.count==0 && DUT.rd_ptr==0 && DUT.wr_ptr==0 && empty && !full && !almostfull && !almostempty);
17  end
18
19  always_comb begin : comb_checks
20      if(rst_n) begin
21          if(DUT.count==FIFO_DEPTH)begin
22              full_check:assert(full==1'b1);
23              full_cover:cover(full==1'b1);
24          end
25          if(DUT.count==inter.FIFO_DEPTH-1)begin
26              almostfull_check:assert(almostfull==1'b1);
27              almostfull_cover:cover(almostfull==1'b1);
28          end
29          if(DUT.count==0)begin
30              empty_check:assert(empty==1'b1);
31              empty_cover:cover(empty==1'b1);
32          end
```

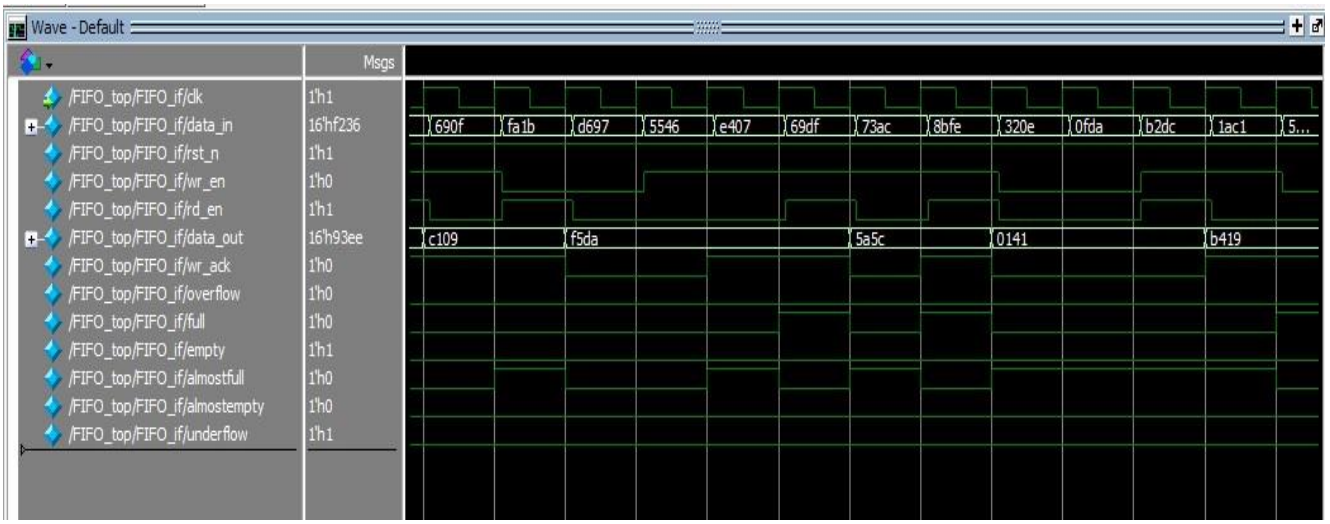
```
        if(DUT.count==1)begin
            almostempty_check:assert(almostempty==1'b1);
            almostempty_cover :cover(almostempty==1'b1);
        end
    end
end


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

writing_assert:assert property (writing);
writing_cover:cover property (writing);

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

reading_assert:assert property (reading);
reading_cover:cover property (reading);

property WriteNotRead;
@(posedge clk) disable iff(!rst_n) (wr_en && rd_en && empty) |=> (DUT.wr_ptr==$past(DUT.wr_ptr)+1'b1);
endproperty

WriteNotRead_assert:assert property (WriteNotRead);
WriteNotRead_cover:cover property (WriteNotRead);
```

```
property ReadNotWrite;
@(posedge clk) disable iff(!rst_n) (wr_en && rd_en && full) |=> (DUT.rd_ptr==$past(DUT.rd_ptr)+1'b1);
endproperty

ReadNotWrite_assert:assert property (ReadNotWrite);
ReadNotWrite_cover:cover property (ReadNotWrite);

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

accept_writing_assert:assert property (accept_writing);
accept_writing_cover:cover property (accept_writing);

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

refuse_writing_assert:assert property (refuse_writing);
refuse_writing_cover:cover property (refuse_writing);

property count_no_change;
@(posedge inter.clk) disable iff(!rst_n) (!wr_en && !rd_en) |=> ($stable(DUT.count));
endproperty

count_no_change_assert:assert property (count_no_change);
count_no_change_cover:cover property (count_no_change);

property count_up;
@(posedge inter.clk) disable iff(!inter.rst_n) ((wr_en && !rd_en && !full)||(wr_en && rd_en && empty))
|=> (DUT.count==$past(DUT.count)+1'b1);
endproperty
```

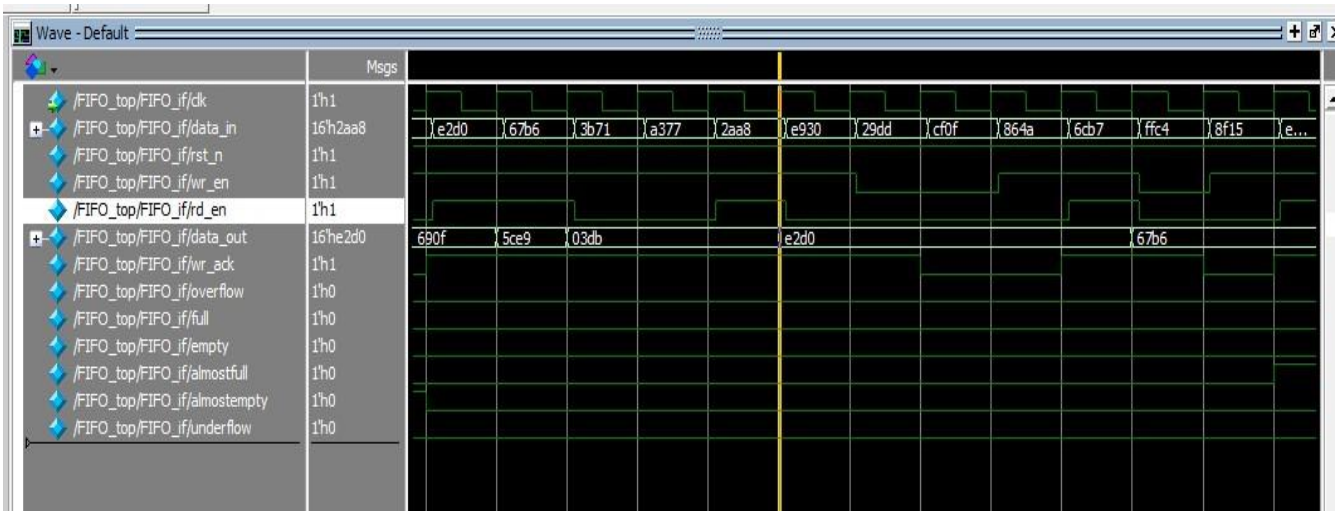
```
count_up_assert:assert property (count_up);
count_up_cover:cover property (count_up);

property count_down;
@(posedge clk) disable iff(!rst_n) ((!wr_en && rd_en && !empty)||(wr_en && rd_en && full))
|=> (DUT.count==$past(DUT.count)-1'b1);
endproperty

count_down_assert:assert property (count_down);
count_down_cover:cover property (count_down);

property count_above;
@(posedge clk) (DUT.count < 4'b1001) ;
endproperty

count_above_assert:assert property (count_above);
count_above_cover:cover property (count_above);

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

over_flow_assert:assert property (over_flow);
over_flow_cover:cover property (over_flow);

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

under_flow_assert:assert property (under_flow);
under_flow_cover:cover property (under_flow);
```

Simulation(read and write)

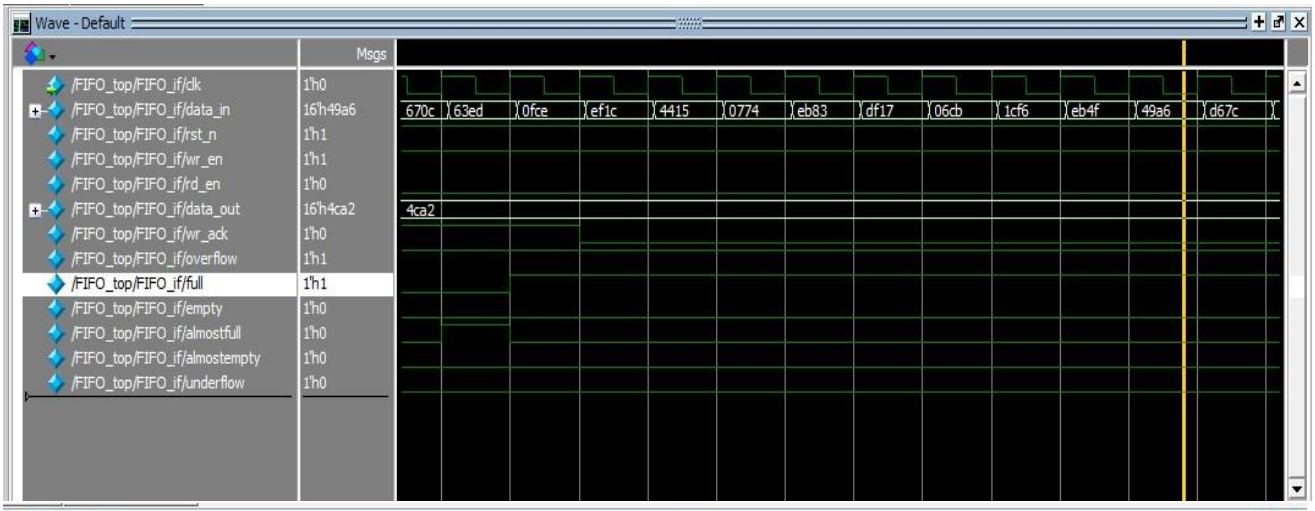


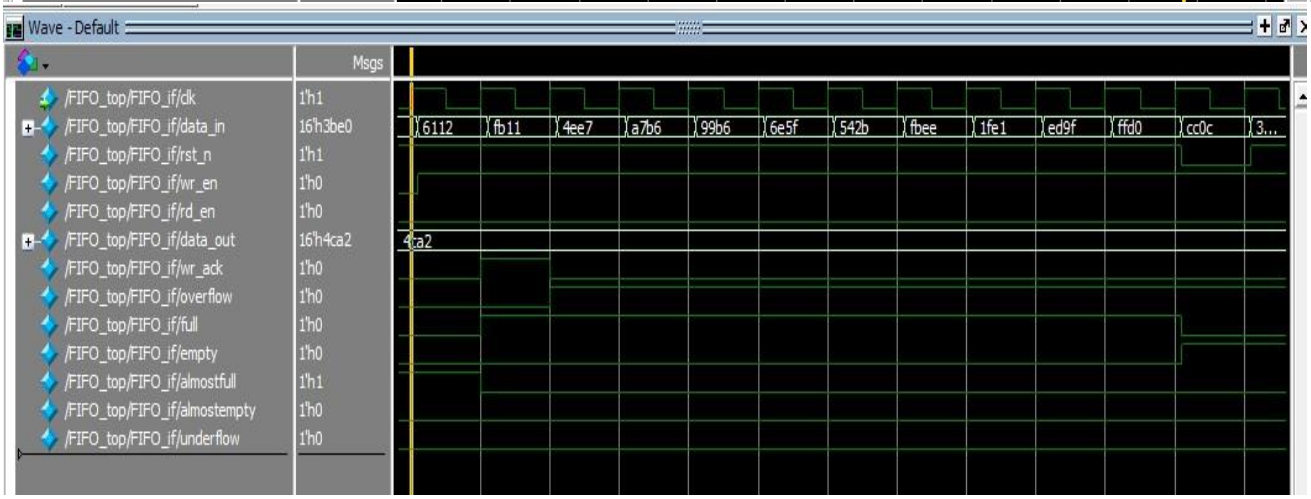

Wave - Default
Msgs
/FIFO_top/FIFO_if/clk
/FIFO_top/FIFO_if/data_in
/FIFO_top/FIFO_if/rst_n
/FIFO_top/FIFO_if/wr_en
/FIFO_top/FIFO_if/rd_en
/FIFO_top/FIFO_if/data_out
/FIFO_top/FIFO_if/wr_ack
/FIFO_top/FIFO_if/overflow
/FIFO_top/FIFO_if/full
/FIFO_top/FIFO_if/empty
/FIFO_top/FIFO_if/almostfull
/FIFO_top/FIFO_if/almostempty
/FIFO_top/FIFO_if/underflow
1'h1
16'hf236
1'h1
1'h0
1'h1
16'h93ee
1'h0
1'h0
1'h0
1'h1
1'h0
1'h0
1'h1
690f
fa1b
d697
5546
e407
69df
73ac
8bfe
320e
0fda
b2dc
1ac1
c109
f5da
5a5c
0141
b419


Wave - Default
Msgs
/FIFO_top/FIFO_if/clk
/FIFO_top/FIFO_if/data_in
/FIFO_top/FIFO_if/rst_n
/FIFO_top/FIFO_if/wr_en
/FIFO_top/FIFO_if/rd_en
/FIFO_top/FIFO_if/data_out
/FIFO_top/FIFO_if/wr_ack
/FIFO_top/FIFO_if/overflow
/FIFO_top/FIFO_if/full
/FIFO_top/FIFO_if/empty
/FIFO_top/FIFO_if/almostfull
/FIFO_top/FIFO_if/almostempty
/FIFO_top/FIFO_if/underflow
1'h1
16'hf236
1'h1
1'h0
1'h1
16'h93ee
1'h0
1'h0
1'h0
1'h1
1'h0
1'h0
1'h1
185e
a076
a73f
cc4f
268e
3b36
9f48
23e6
48a7
2b4e
5ce9
2b2d
03db
b419
4398
0fd2
690f


Wave - Default
Msgs
/FIFO_top/FIFO_if/clk
/FIFO_top/FIFO_if/data_in
/FIFO_top/FIFO_if/rst_n
/FIFO_top/FIFO_if/wr_en
/FIFO_top/FIFO_if/rd_en
/FIFO_top/FIFO_if/data_out
/FIFO_top/FIFO_if/wr_ack
/FIFO_top/FIFO_if/overflow
/FIFO_top/FIFO_if/full
/FIFO_top/FIFO_if/empty
/FIFO_top/FIFO_if/almostfull
/FIFO_top/FIFO_if/almostempty
/FIFO_top/FIFO_if/underflow
1'h1
16'h2aa8
1'h1
1'h1
1'h1
16'he2d0
1'h1
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
e2d0
67b6
3b71
a377
2aa8
e930
29dd
cf0f
864a
6cb7
ffc4
8f15
690f
5ce9
03db
e2d0
67b6

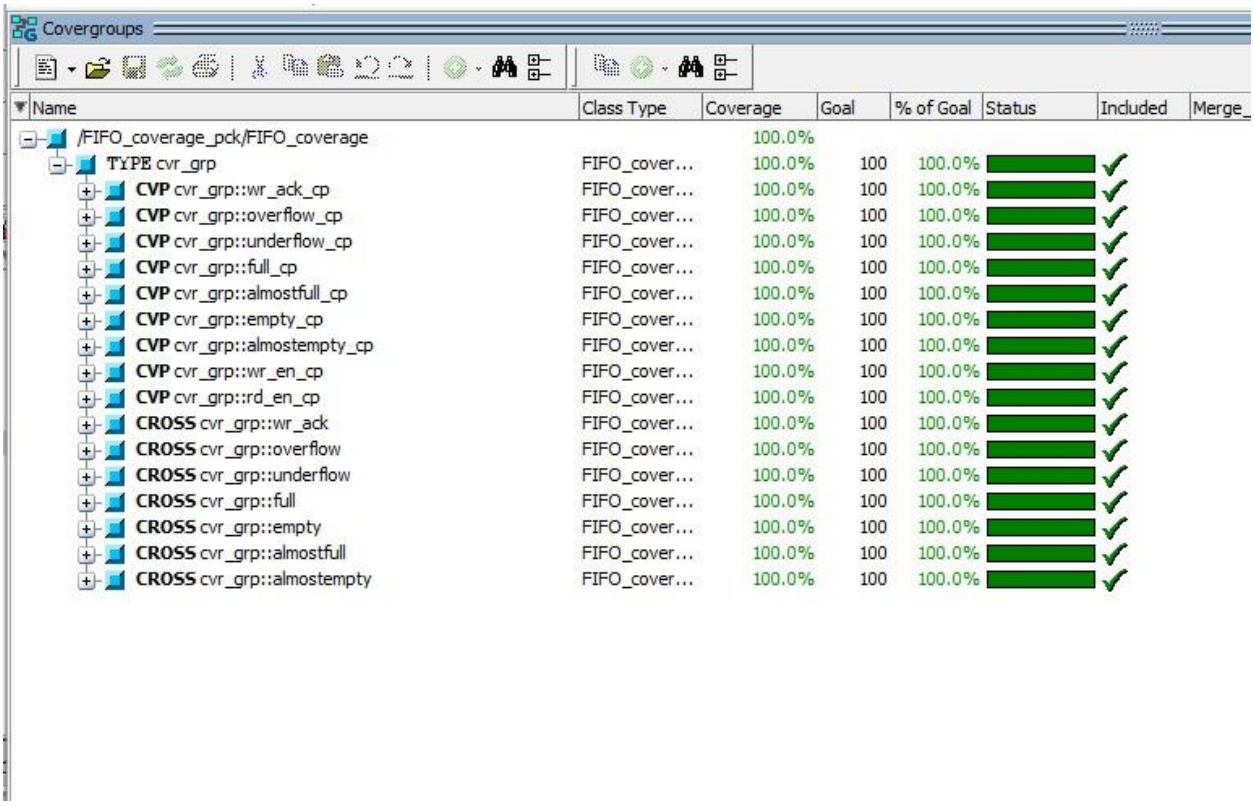
Wrire only sequence





Read only

Functional coverage




```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------
Covergroup                                                          Metric       Goal     Status

----------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pck/FIFO_coverage/cvr_grp                     100.0%        100     Covered
    covered/total bins:                                               74         74
    missing/total bins:                                               0         74
    % Hit:                                                        100.0%        100
    Coverpoint cvr_grp::wr_ack_cp                                 100.0%        100     Covered
        covered/total bins:                                            2          2
        missing/total bins:                                            0          2
        % Hit:                                                    100.0%        100
        bin wr_ack_0                                                3449          1     Covered
        bin wr_ack_1                                                2552          1     Covered
    Coverpoint cvr_grp::overflow_cp                               100.0%        100     Covered
        covered/total bins:                                            2          2
        missing/total bins:                                            0          2
        % Hit:                                                    100.0%        100
        bin overflow_0                                              4829          1     Covered
        bin overflow_1                                              1172          1     Covered
    Coverpoint cvr_grp::underflow_cp                              100.0%        100     Covered
        covered/total bins:                                            2          2
        missing/total bins:                                            0          2
        % Hit:                                                    100.0%        100
        bin underflow_0                                             5474          1     Covered
        bin underflow_1                                              527          1     Covered
    Coverpoint cvr_grp::full_cp                                   100.0%        100     Covered
        covered/total bins:                                            2          2
        missing/total bins:                                            0          2
        % Hit:                                                    100.0%        100
        bin full_0                                                  4507          1     Covered
        bin full_1                                                  1494          1     Covered
    Coverpoint cvr_grp::almostfull_cp                             100.0%        100     Covered
```

Coverpoint cvr_grp::almostfull_cp	100.0%	100	Covered
covered/total bins:	2	2	
missing/total bins:	0	2	
% Hit:	100.0%	100	
bin almostfull_0	5132	1	Covered
bin almostfull_1	869	1	Covered
Coverpoint cvr_grp::empty_cp	100.0%	100	Covered
covered/total bins:	2	2	
missing/total bins:	0	2	
% Hit:	100.0%	100	
bin empty_0	4818	1	Covered
bin empty_1	1183	1	Covered
Coverpoint cvr_grp::almostempty_cp	100.0%	100	Covered
covered/total bins:	2	2	
missing/total bins:	0	2	
% Hit:	100.0%	100	
bin almostempty_0	5475	1	Covered
bin almostempty_1	526	1	Covered
Coverpoint cvr_grp::wr_en_cp	100.0%	100	Covered
covered/total bins:	2	2	
missing/total bins:	0	2	
% Hit:	100.0%	100	
bin wr_en_0	1986	1	Covered
bin wr_en_1	4015	1	Covered
Coverpoint cvr_grp::rd_en_cp	100.0%	100	Covered
covered/total bins:	2	2	
missing/total bins:	0	2	
% Hit:	100.0%	100	
bin rd_en_0	3960	1	Covered
bin rd_en_1	2041	1	Covered
Cross cvr_grp::wr_ack	100.0%	100	Covered
covered/total bins:	8	8	
missing/total bins:	0	8	
% Hit:	100.0%	100	
bin <wr_en_0,rd_en_0,wr_ack_0>	561	1	Covered
bin <wr_en_1,rd_en_0,wr_ack_0>	1569	1	Covered

bin <wr_en_1,rd_en_1,wr_ack_1>	508	1	Covered
Cross cvr_grp::overflow	100.0%	100	Covered
covered/total bins:	8	8	
missing/total bins:	0	8	
% Hit:	100.0%	100	
bin <wr_en_0,rd_en_0,overflow_0>	909	1	Covered
bin <wr_en_1,rd_en_0,overflow_0>	2090	1	Covered
bin <wr_en_0,rd_en_1,overflow_0>	879	1	Covered
bin <wr_en_1,rd_en_1,overflow_0>	951	1	Covered
bin <wr_en_0,rd_en_0,overflow_1>	134	1	Covered
bin <wr_en_1,rd_en_0,overflow_1>	827	1	Covered
bin <wr_en_0,rd_en_1,overflow_1>	64	1	Covered
bin <wr_en_1,rd_en_1,overflow_1>	147	1	Covered
Cross cvr_grp::underflow	100.0%	100	Covered
covered/total bins:	8	8	
missing/total bins:	0	8	
% Hit:	100.0%	100	
bin <wr_en_0,rd_en_0,underflow_0>	1035	1	Covered
bin <wr_en_1,rd_en_0,underflow_0>	2900	1	Covered
bin <wr_en_0,rd_en_1,underflow_0>	448	1	Covered
bin <wr_en_1,rd_en_1,underflow_0>	1091	1	Covered
bin <wr_en_0,rd_en_0,underflow_1>	8	1	Covered
bin <wr_en_1,rd_en_0,underflow_1>	17	1	Covered
bin <wr_en_0,rd_en_1,underflow_1>	495	1	Covered
bin <wr_en_1,rd_en_1,underflow_1>	7	1	Covered
Cross cvr_grp::full	100.0%	100	Covered
covered/total bins:	8	8	
missing/total bins:	0	8	
% Hit:	100.0%	100	
bin <wr_en_0,rd_en_0,full_0>	790	1	Covered
bin <wr_en_1,rd_en_0,full_0>	2019	1	Covered
bin <wr_en_0,rd_en_1,full_0>	843	1	Covered
bin <wr_en_1,rd_en_1,full_0>	855	1	Covered
bin <wr_en_0,rd_en_0,full_1>	253	1	Covered
bin <wr_en_1,rd_en_0,full_1>	898	1	Covered
bin <wr_en_0,rd_en_1,full_1>	100	1	Covered

```
      bin <wr_en_0,rd_en_0,empty_0>                897          1    Covered
      bin <wr_en_1,rd_en_0,empty_0>               2577          1    Covered
      bin <wr_en_0,rd_en_1,empty_0>                391          1    Covered
      bin <wr_en_1,rd_en_1,empty_0>                953          1    Covered
      bin <wr_en_0,rd_en_0,empty_1>                146          1    Covered
      bin <wr_en_1,rd_en_0,empty_1>                340          1    Covered
      bin <wr_en_0,rd_en_1,empty_1>                552          1    Covered
      bin <wr_en_1,rd_en_1,empty_1>                145          1    Covered
  Cross cvr_grp::almostfull                      100.0%        100    Covered
      covered/total bins:                             8          8
      missing/total bins:                             0          8
      % Hit:                                     100.0%        100
      bin <wr_en_0,rd_en_0,almostfull_0>           857          1    Covered
      bin <wr_en_1,rd_en_0,almostfull_0>          2501          1    Covered
      bin <wr_en_0,rd_en_1,almostfull_0>           871          1    Covered
      bin <wr_en_1,rd_en_1,almostfull_0>           903          1    Covered
      bin <wr_en_0,rd_en_0,almostfull_1>           186          1    Covered
      bin <wr_en_1,rd_en_0,almostfull_1>           416          1    Covered
      bin <wr_en_0,rd_en_1,almostfull_1>            72          1    Covered
      bin <wr_en_1,rd_en_1,almostfull_1>           195          1    Covered
  Cross cvr_grp::almostempty                     100.0%        100    Covered
      covered/total bins:                             8          8
      missing/total bins:                             0          8
      % Hit:                                     100.0%        100
      bin <wr_en_0,rd_en_0,almostempty_0>          937          1    Covered
      bin <wr_en_1,rd_en_0,almostempty_0>         2647          1    Covered
      bin <wr_en_0,rd_en_1,almostempty_0>          896          1    Covered
      bin <wr_en_1,rd_en_1,almostempty_0>          995          1    Covered
      bin <wr_en_0,rd_en_0,almostempty_1>          106          1    Covered
      bin <wr_en_1,rd_en_0,almostempty_1>          270          1    Covered
      bin <wr_en_0,rd_en_1,almostempty_1>           47          1    Covered
      bin <wr_en_1,rd_en_1,almostempty_1>          103          1    Covered
LASS FIFO_coverage

TAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```

Code coverage

```
================================================================================
=== File: FIFO.sv
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           25        25         0     100.0

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

Statement Coverage for file FIFO.sv --

    1                                       ////////////////////////////////////////////////////////////
    2                                       // Author: Kareem Waseem
    3                                       // Course: Digital Verification using SV & UVM
    4                                       //
    5                                       // Description: FIFO Design
    6                                       //
    7                                       ////////////////////////////////////////////////////////////
    8                                       import uvm_pkg::*;
    9                                       `include "uvm_macros.svh"
    10                                      module FIFO(FIFO_interface.dut inter);
    11
    12                                      localparam max_fifo_addr = $clog2(inter.FIFO_DEPTH);
    13
    14                                      reg [inter.FIFO_WIDTH-1:0] mem [inter.FIFO_DEPTH-1:0];
    15
    16                                      reg [max_fifo_addr-1:0] wr_ptr=0, rd_ptr=0;
    17                                      reg [max_fifo_addr:0] count=0;//
    18
    19              1                  6279     always @(posedge inter.clk or negedge inter.rst_n) begin
    20                                             if (!inter.rst_n) begin
    21              1                   575                wr_ptr <= 0;
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        25        25         0     100.0

==============================Branch Details==============================
```

```
Condition Coverage:
    Enabled Coverage            Active    Covered    Misses % Covered
    ----------------            ------    -------    ------ ---------
    FEC Condition Terms              22         22         0     100.0

==============================Condition Details==============================
```

```
================================Toggle Details================================

Toggle Coverage for File FIFO.sv --

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

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

Toggle Coverage      =      100.0% (20 of 20 bins)

================================================================================
```

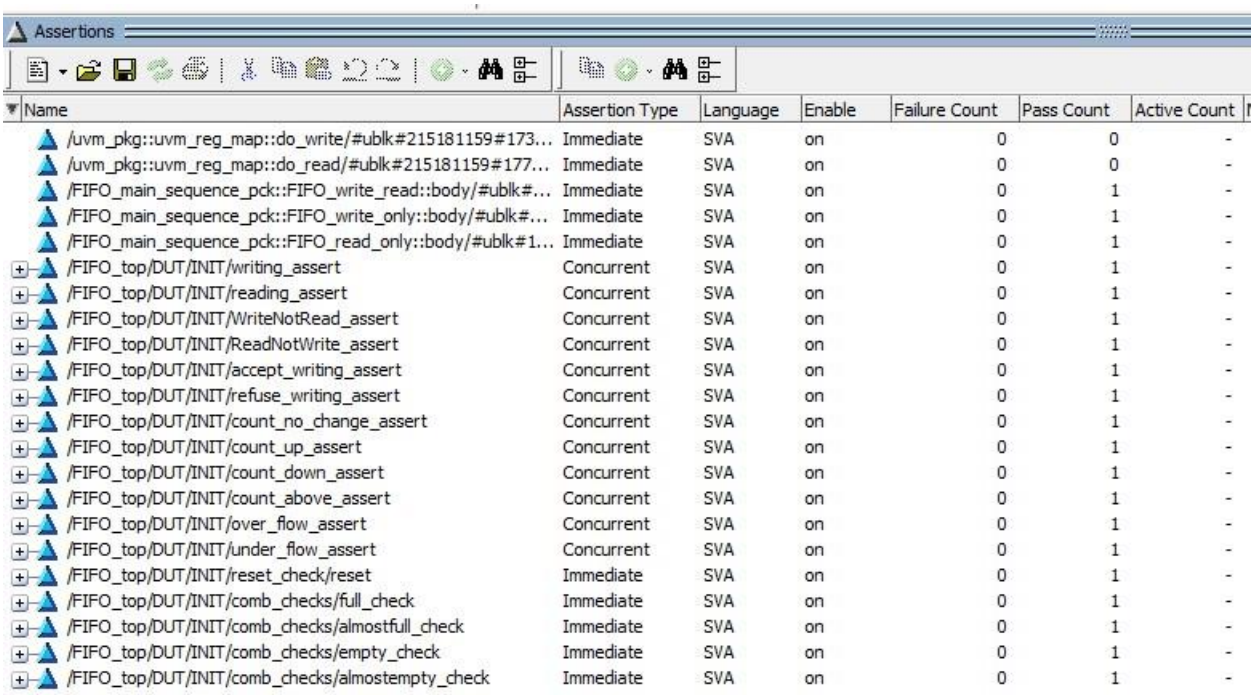
```
================================Toggle Details================================

Toggle Coverage for File FIFO_interface.sv --

       Line                                  Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          8                                   clk           1           1      100.00
          9                              data_in[9]           1           1      100.00
          9                              data_in[8]           1           1      100.00
          9                              data_in[7]           1           1      100.00
          9                              data_in[6]           1           1      100.00
          9                              data_in[5]           1           1      100.00
          9                              data_in[4]           1           1      100.00
          9                              data_in[3]           1           1      100.00
          9                              data_in[2]           1           1      100.00
          9                              data_in[1]           1           1      100.00
          9                             data_in[15]           1           1      100.00
          9                             data_in[14]           1           1      100.00
          9                             data_in[13]           1           1      100.00
          9                             data_in[12]           1           1      100.00
          9                             data_in[11]           1           1      100.00
          9                             data_in[10]           1           1      100.00
          9                              data_in[0]           1           1      100.00
         10                                   wr_en           1           1      100.00
         10                                   rst_n           1           1      100.00
         10                                   rd_en           1           1      100.00
         11                             data_out[9]           1           1      100.00
         11                             data_out[8]           1           1      100.00
         11                             data_out[7]           1           1      100.00
         11                             data_out[6]           1           1      100.00
         11                             data_out[5]           1           1      100.00
         11                             data_out[4]           1           1      100.00
         11                             data_out[3]           1           1      100.00
         11                             data_out[2]           1           1      100.00
```

```
     10                          rst_n        1          1      100.00
     10                          rd_en        1          1      100.00
     11                    data_out[9]        1          1      100.00
     11                    data_out[8]        1          1      100.00
     11                    data_out[7]        1          1      100.00
     11                    data_out[6]        1          1      100.00
     11                    data_out[5]        1          1      100.00
     11                    data_out[4]        1          1      100.00
     11                    data_out[3]        1          1      100.00
     11                    data_out[2]        1          1      100.00
     11                    data_out[1]        1          1      100.00
     11                   data_out[15]        1          1      100.00
     11                   data_out[14]        1          1      100.00
     11                   data_out[13]        1          1      100.00
     11                   data_out[12]        1          1      100.00
     11                   data_out[11]        1          1      100.00
     11                   data_out[10]        1          1      100.00
     11                    data_out[0]        1          1      100.00
     12                         wr_ack        1          1      100.00
     12                       overflow        1          1      100.00
     13                      underflow        1          1      100.00
     13                           full        1          1      100.00
     13                          empty        1          1      100.00
     13                     almostfull        1          1      100.00
     13                    almostempty        1          1      100.00

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

Toggle Coverage      =      100.0% (86 of 86 bins)

=================================================================================
```

Assertion coverage

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory
/FIFO_top/DUT/INIT/reset_check/reset_cover	SVA	✓	Off	1407	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/comb_checks/full_cover	SVA	✓	Off	565	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/comb_checks/almostfull_cover	SVA	✓	Off	700	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/comb_checks/empty_cover	SVA	✓	Off	750	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/comb_checks/almostempty_cover	SVA	✓	Off	564	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/writing_cover	SVA	✓	Off	3991	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/reading_cover	SVA	✓	Off	368	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/WriteNotRead_cover	SVA	✓	Off	78	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/ReadNotWrite_cover	SVA	✓	Off	227	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/accept_writing_cover	SVA	✓	Off	3991	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/refuse_writing_cover	SVA	✓	Off	3685	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/count_no_change_cover	SVA	✓	Off	947	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/count_up_cover	SVA	✓	Off	3316	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/count_down_cover	SVA	✓	Off	595	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/count_above_cover	SVA	✓	Off	15002	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/over_flow_cover	SVA	✓	Off	3458	1	Unli...	1	100%		✓	0
/FIFO_top/DUT/INIT/under_flow_cover	SVA	✓	Off	4563	1	Unli...	1	100%		✓	0



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#173...	Immediate	SVA	on	0	0	-
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#177...	Immediate	SVA	on	0	0	-
/FIFO_main_sequence_pck::FIFO_write_read::body/#ublk#...	Immediate	SVA	on	0	1	-
/FIFO_main_sequence_pck::FIFO_write_only::body/#ublk#...	Immediate	SVA	on	0	1	-
/FIFO_main_sequence_pck::FIFO_read_only::body/#ublk#1...	Immediate	SVA	on	0	1	-
/FIFO_top/DUT/INIT/writing_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/reading_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/WriteNotRead_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/ReadNotWrite_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/accept_writing_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/refuse_writing_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/count_no_change_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/count_up_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/count_down_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/count_above_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/over_flow_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/under_flow_assert	Concurrent	SVA	on	0	1	-
/FIFO_top/DUT/INIT/reset_check/reset	Immediate	SVA	on	0	1	-
/FIFO_top/DUT/INIT/comb_checks/full_check	Immediate	SVA	on	0	1	-
/FIFO_top/DUT/INIT/comb_checks/almostfull_check	Immediate	SVA	on	0	1	-
/FIFO_top/DUT/INIT/comb_checks/empty_check	Immediate	SVA	on	0	1	-
/FIFO_top/DUT/INIT/comb_checks/almostempty_check	Immediate	SVA	on	0	1	-

```
------------------------------------------------------------
Name                  File(Line)           Failure Pass
                                           Count   Count
------------------------------------------------------------
/FIFO_top/DUT/INIT/reset_check/reset
                      FIFO_assertions.sv(15)       0     1
/FIFO_top/DUT/INIT/comb_checks/full_check
                      FIFO_assertions.sv(22)       0     1
/FIFO_top/DUT/INIT/comb_checks/almostfull_check
                      FIFO_assertions.sv(26)       0     1
/FIFO_top/DUT/INIT/comb_checks/empty_check
                      FIFO_assertions.sv(30)       0     1
/FIFO_top/DUT/INIT/comb_checks/almostempty_check
                      FIFO_assertions.sv(34)       0     1
/FIFO_top/DUT/INIT/writing_assert
                      FIFO_assertions.sv(45)       0     1
/FIFO_top/DUT/INIT/reading_assert
                      FIFO_assertions.sv(52)       0     1
/FIFO_top/DUT/INIT/WriteNotRead_assert
                      FIFO_assertions.sv(59)       0     1
/FIFO_top/DUT/INIT/ReadNotWrite_assert
                      FIFO_assertions.sv(66)       0     1
/FIFO_top/DUT/INIT/accept_writing_assert
                      FIFO_assertions.sv(73)       0     1
/FIFO_top/DUT/INIT/refuse_writing_assert
                      FIFO_assertions.sv(80)       0     1
/FIFO_top/DUT/INIT/count_no_change_assert
                      FIFO_assertions.sv(87)       0     1
/FIFO_top/DUT/INIT/count_up_assert
                      FIFO_assertions.sv(95)       0     1
/FIFO_top/DUT/INIT/count_down_assert
                      FIFO_assertions.sv(103)      0     1
/FIFO_top/DUT/INIT/count_above_assert
                      FIFO_assertions.sv(110)      0     1
/FIFO_top/DUT/INIT/over_flow_assert
                      FIFO_assertions.sv(117)      0     1
```

```
                    FIFO_assertions.sv(73)       0     1
/FIFO_top/DUT/INIT/refuse_writing_assert
                    FIFO_assertions.sv(80)       0     1
/FIFO_top/DUT/INIT/count_no_change_assert
                    FIFO_assertions.sv(87)       0     1
/FIFO_top/DUT/INIT/count_up_assert
                    FIFO_assertions.sv(95)       0     1
/FIFO_top/DUT/INIT/count_down_assert
                    FIFO_assertions.sv(103)       0     1
/FIFO_top/DUT/INIT/count_above_assert
                    FIFO_assertions.sv(110)       0     1
/FIFO_top/DUT/INIT/over_flow_assert
                    FIFO_assertions.sv(117)       0     1
/FIFO_top/DUT/INIT/under_flow_assert
                    FIFO_assertions.sv(124)       0     1
/FIFO_main_sequence_pck/FIFO_write_read/body/#ublk#123879083#14/immed__17
                    FIFO_main_sequence.sv(17)       0     1
/FIFO_main_sequence_pck/FIFO_write_only/body/#ublk#123879083#31/immed__38
                    FIFO_main_sequence.sv(38)       0     1
/FIFO_main_sequence_pck/FIFO_read_only/body/#ublk#123879083#52/immed__59
                    FIFO_main_sequence.sv(59)       0     1
```

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                      Design Design   Lang File(Line)      Count Status
                                          Unit   UnitType
--------------------------------------------------------------------------------------------
/FIFO_top/DUT/INIT/reset_check/reset_cover
                                          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(16)
                                                                                557 Covered
/FIFO_top/DUT/INIT/comb_checks/full_cover
                                          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(23)
                                                                                416 Covered
/FIFO_top/DUT/INIT/comb_checks/almostfull_cover
                                          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(27)
                                                                                537 Covered
/FIFO_top/DUT/INIT/comb_checks/empty_cover
                                          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(31)
                                                                                325 Covered
/FIFO_top/DUT/INIT/comb_checks/almostempty_cover
                                          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(35)
                                                                                341 Covered
/FIFO_top/DUT/INIT/writing_cover          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(46)
                                                                               2552 Covered
/FIFO_top/DUT/INIT/reading_cover          FIFO_assertions Verilog  SVA  FIFO_assertions.sv(53)
                                                                                368 Covered
/FIFO_top/DUT/INIT/WriteNotRead_cover     FIFO_assertions Verilog  SVA  FIFO_assertions.sv(60)
                                                                                 78 Covered
/FIFO_top/DUT/INIT/ReadNotWrite_cover     FIFO_assertions Verilog  SVA  FIFO_assertions.sv(67)
                                                                                227 Covered
/FIFO_top/DUT/INIT/accept_writing_cover   FIFO_assertions Verilog  SVA  FIFO_assertions.sv(74)
                                                                               2552 Covered
/FIFO_top/DUT/INIT/refuse_writing_cover   FIFO_assertions Verilog  SVA  FIFO_assertions.sv(81)
                                                                               1082 Covered
/FIFO_top/DUT/INIT/count_no_change_cover FIFO_assertions Verilog  SVA  FIFO_assertions.sv(88)
                                                                                947 Covered
/FIFO_top/DUT/INIT/count_up_cover         FIFO_assertions Verilog  SVA  FIFO_assertions.sv(96)
```

```
                                                                                227 Covered
/FIFO_top/DUT/INIT/accept_writing_cover   FIFO_assertions Verilog  SVA  FIFO_assertions.sv(74)
                                                                               2552 Covered
/FIFO_top/DUT/INIT/refuse_writing_cover   FIFO_assertions Verilog  SVA  FIFO_assertions.sv(81)
                                                                               1082 Covered
/FIFO_top/DUT/INIT/count_no_change_cover FIFO_assertions Verilog  SVA  FIFO_assertions.sv(88)
                                                                                947 Covered
/FIFO_top/DUT/INIT/count_up_cover         FIFO_assertions Verilog  SVA  FIFO_assertions.sv(96)
                                                                               1877 Covered
/FIFO_top/DUT/INIT/count_down_cover       FIFO_assertions Verilog  SVA  FIFO_assertions.sv(104)
                                                                                595 Covered
/FIFO_top/DUT/INIT/count_above_cover      FIFO_assertions Verilog  SVA  FIFO_assertions.sv(111)
                                                                               6002 Covered
/FIFO_top/DUT/INIT/over_flow_cover        FIFO_assertions Verilog  SVA  FIFO_assertions.sv(118)
                                                                                855 Covered
/FIFO_top/DUT/INIT/under_flow_cover       FIFO_assertions Verilog  SVA  FIFO_assertions.sv(125)
                                                                                477 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 17
```

UVM report

```
#
#   ***********       IMPORTANT RELEASE NOTES         ************
#
#   You are using a version of the UVM library that has been compiled
#   with `UVM_NO_DEPRECATED undefined.
#   See http://www.eda.org/svdb/view.php?id=3313 for more details.
#
#   You are using a version of the UVM library that has been compiled
#   with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR undefined.
#   See http://www.eda.org/svdb/view.php?id=3770 for more details.
#
#       (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test FIFO_test...
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(40) @ 0: uvm_test_top [run_phase] reset asserted
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(42) @ 11: uvm_test_top [run_phase] reset deasserted
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(43) @ 11: uvm_test_top [run_phase] stimulas generation for write and read started
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(45) @ 50011: uvm_test_top [run_phase] stimulas generation for write and read ended
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(46) @ 50011: uvm_test_top [run_phase] stimulas generation for write only started
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(48) @ 55011: uvm_test_top [run_phase] stimulas generation for write only ended
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(49) @ 55011: uvm_test_top [run_phase] stimulas generation for read only started
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_test.sv(51) @ 60011: uvm_test_top [run_phase] stimulas generation for read only endedd
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1268) @ 60011: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_scoreboard.sv(81) @ 60011: uvm_test_top.env.sb [run_phase] total successful transactions :       6001
# UVM_INFO C:/Users/DELL/Desktop/System Veri;og-UVM Diploma/UVM_project/FIFO_scoreboard.sv(82) @ 60011: uvm_test_top.env.sb [run_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
# [run_phase]    10
# ** Note: $finish    : C:/questasim64_10.6c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
```

Assertion details

Immediate Assertions

Feature	Assertion
When reset is asserted all internal signals and outputs should be zero except the empty flag and data out	if(!rst_n) reset: assert final (DUT.count==0 && DUT.rd_ptr==0 && DUT.wr_ptr==0 && empty && !full && !almostfull && !almostempty);

When reset is deasserted and count=FIFO_DEPTH ,the full flag rises.	if(rst_n) begin
	if(DUT.count==FIFO_DEPTH)
	full_check:assert(full==1'b1);

When reset is deasserted and count=FIFO_DEPTH-1 ,the almostfull flag rises.	if(rst_n) begin
	if(DUT.count==FIFO_DEPTH-1)
	full_check:assert(almostfull==1'b1);

When reset is deasserted and count= 1 ,the almostempty flag rises.	if(rst_n) begin
	if(DUT.count==1'b1)
	full_check:assert(almostempty ==1'b1);

When reset is deasserted and count= 0 ,the empty flag rises.	if(rst_n) begin
	if(DUT.count==1'b0)
	full_check:assert(empty ==1'b1);

Concurrent Assertions

Feature	Assertion
At posedge clk if write enable is high and fifo isn't full the write pointer increments in a cycle	@(posedge clk) disable iff(!rst_n) (wr_en && !full) |=> (DUT.wr_ptr==$past(DUT.wr_ptr)+1'b1);
At posedge clk if write enable is low,read enable is high and fifo isn't empty the read pointer increments in a cycle	@(posedge clk) disable iff(!rst_n) (!wr_en && rd_en && !empty) |=> (DUT.rd_ptr==$past(DUT.rd_ptr)+1'b1);

At posedge clk if write enable is high,read enable is high and fifo is empty the write pointer increments in a cycle	@(posedge clk) disable iff(!rst_n) (wr_en && rd_en && empty) |=> (DUT.wr_ptr==$past(DUT.wr_ptr)+1'b1);
At posedge clk if write enable is high,read enable is high and fifo is full the read pointer increments in a cycle	@(posedge clk) disable iff(!rst_n) (wr_en && rd_en && full) |=> (DUT.rd_ptr==$past(DUT.rd_ptr)+1'b1);
At posedge clk if write enable is high and fifo is full the wr_ack will be low in a cycle(failing to write)	@(posedge clk) disable iff(!rst_n) (wr_en && full) |=> (!wr_ack);
At posedge clk if write enable is high and fifo isn't full the wr_ack will be high in a cycle(manage to write)	@(posedge clk) disable iff(!rst_n) (wr_en && !full) |=> (wr_ack);
At posedge clk if write enable is low,read enable is low the count will not change	@(posedge inter.clk) disable iff(!rst_n) (!wr_en && !rd_en) |=> ($stable(DUT.count));
At posedge clk if write enable is high,read enable is low and fifo isn't full or write enable is high,read enable is high and fifo is empty the counter increments in a cycle	@(posedge inter.clk) disable iff(!inter.rst_n) ((wr_en && !rd_en && !full)||(wr_en && rd_en && empty)) |=> (DUT.count==$past(DUT.count)+1'b1);
At posedge clk if write enable is high,read enable is high and fifo is full or write enable is low,read enable is high and fifo isn't empty the counter decrements in a cycle	@(posedge clk) disable iff(!rst_n) ((!wr_en && rd_en && !empty)||(wr_en && rd_en && full)) |=> (DUT.count==$past(DUT.count)-1'b1);
Count shouldn't exceed the fifo depth at any time	@(posedge clk) (DUT.count < 4'b1001) ;
At posedge clk if write enable is high,read enable is low and fifo is full the overflow rises in a cycle	@(posedge clk) disable iff(!rst_n) (wr_en && !rd_en && full) |=> (overflow);
At posedge clk if write enable is low,read enable is high and fifo is empty the underflow rises in a cycle	@(posedge clk) disable iff(!rst_n) (!wr_en && rd_en && empty) |=> (underflow);