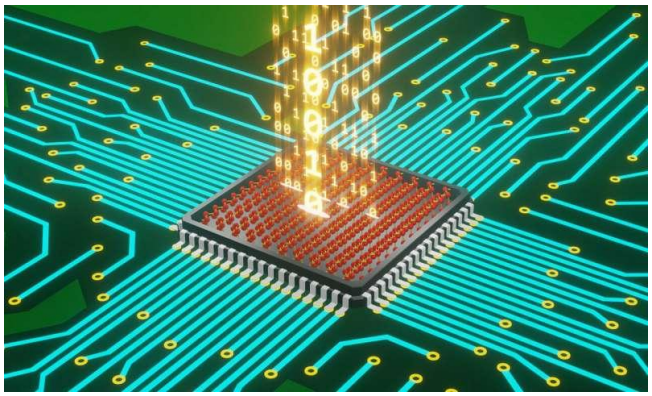


SYNCHRONOUS FIFO(UVM)

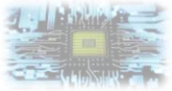
under supervision: Eng. Kareem Waseem

FIFO:

A synchronous FIFO (First-In-First-Out) is a type of data buffer used in digital systems that operates under a single clock domain. This means that both read and write operations occur using the same clock signal, ensuring that data is processed in the order it was received.

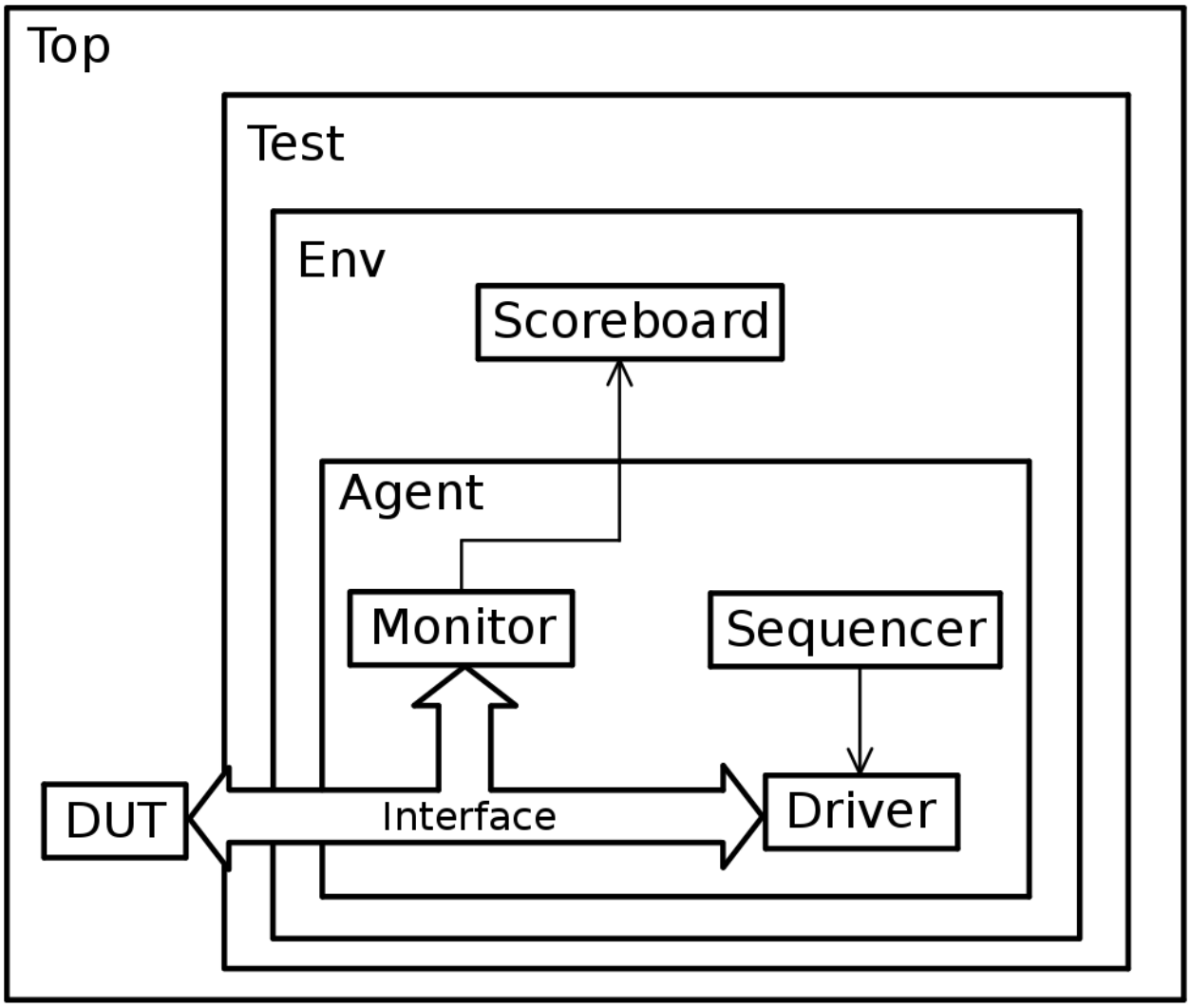
Shams Khaled

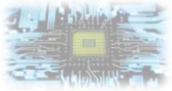
Digital Verification using SV and UVM



Contents

Verification Diagram:

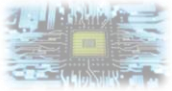




Design:

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

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

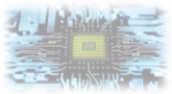
```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
    end
end

assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign underflow = (empty && rd_en)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
assign almostempty = (count == 1)? 1 : 0;

endmodule
```



Modified Design:

FIFO.sv > FIFO > rd_ptr_p

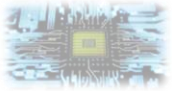
```
8   module FIFO(fifo_interface.DUT fifo_if);
9   parameter FIFO_WIDTH = 16;
10  parameter FIFO_DEPTH = 8;
11
12  localparam max_fifo_addr = $clog2(FIFO_DEPTH);
13  reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
14  reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
15  reg [max_fifo_addr:0] count;
16
17  always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
18      if (!fifo_if.rst_n) begin
19          wr_ptr <= 0;
20          fifo_if.wr_ack<=0;////is added////////
21          fifo_if.overflow<=0;////is added////////
22      end
23      else if (fifo_if.wr_en && count < FIFO_DEPTH) begin//write
24          mem[wr_ptr] <= fifo_if.data_in;
25          fifo_if.wr_ack <= 1;
26          wr_ptr <= wr_ptr + 1;
27          fifo_if.overflow<=0;////is added////////
28
29      end
30      else begin
31          fifo_if.wr_ack <= 0;
32          if (fifo_if.full & fifo_if.wr_en)
33              fifo_if.overflow <= 1;
34          else
35              fifo_if.overflow <= 0;
36      end
37  end
38
39  always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin //read
40      if (!fifo_if.rst_n) begin
41          rd_ptr <= 0;
42          fifo_if.underflow<=0;////is added////////
43
44      end
45      else if (fifo_if.rd_en && count != 0) begin
46          fifo_if.data_out <= mem[rd_ptr];
47          rd_ptr <= rd_ptr + 1;
48          fifo_if.underflow<=0;////is added////////
49      end
50      else if (fifo_if.empty && fifo_if.rd_en) ////is added////////
51          fifo_if.underflow<=1;                ////is added////////
52      else    fifo_if.underflow<=0;            ////is added////////
53
54  end
55
```



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

assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifo_if.empty = (count == 0)? 1 : 0;
assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //corrected 1 not 2////
assign fifo_if.almostempty = (count == 1)? 1 : 0;
```

Bugs in design:

	Design	Modified design
	Underflow is combinational with assign statement	This sequential output signal
	Almostfull signal is set to high when count ==fifo depth -**2**	Almostfull signal is set to high when count ==fifo depth -**1**
	Resetting the signals overflow, wr_ack and underflow was missing	Underflow =0 when reset and fifo is not empty overflow=0 when reset and full fifo wr_ack=0 when reset is asserted
	2 missing cases of wr_en & rd_en	When write and read are both high

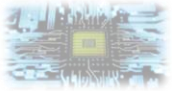


Top module:

```
8   import uvm_pkg::*;
9   import fifo_test_pkg::*;
10  import fifo_env_pkg::*;
11  `include "uvm_macros.svh"
12  module top();
13    bit clk;
14    initial begin
15      forever begin
16        #1 clk=!clk;  // Clock generation
17      end
18    end
19
20  fifo_if fifoif (clk);
21  FIFO DUT(fifoif);
22  bind FIFO fifo_sva fifo_sva_inst(fifoif);
23  initial begin
24    uvm_config_db#(virtual fifo_if)::set(null, "uvm_test_top","fifo_If", fifoif);
25    run_test("fifo_test");
26  end
27
28  endmodule
```

Interface:

```
8   interface fifo_if (clk);
9   parameter FIFO_WIDTH = 16;
10  parameter FIFO_DEPTH = 8;
11  input bit clk;
12  logic [FIFO_WIDTH-1:0] data_in;
13  logic  rst_n, wr_en, rd_en;
14  logic [FIFO_WIDTH-1:0] data_out;
15  logic  wr_ack, overflow;
16  logic full, empty, almostfull, almostempty, underflow;
17  //bit test_finished;
18  modport DUT (input data_in, clk, rst_n, wr_en, rd_en,output data_out,  wr_ack, overflow, full, empty, almostfull, almostempty, underflow);
19
20  endinterface : fifo_if
```

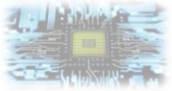


Test package:

```
package fifo_test_pkg;
import fifo_config_pkg::*;
import fifo_sequence_pkg::*;
import fifo_env_pkg::*;
import uvm_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 fifo_config_cfg;
    fifo_sequence fifo_sequence_obj;

    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_cfg = fifo_config::type_id::create("fifo_config_cfg");
        fifo_sequence_obj = fifo_sequence::type_id::create("fifo_sequence_obj",this);//////////////////

        if(!uvm_config_db#(virtual fifo_if)::get(this,"","fifo_If" , fifo_config_cfg.fifo_config_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)::set(this,"*","CFG" , fifo_config_cfg);
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        `uvm_info("run_phase", "stimulus Generation started",UVM_LOW);
        fifo_sequence_obj.start(env.agent.sqr);
        `uvm_info("run_phase", "stimulus Generation ended",UVM_LOW);
        phase.drop_objection(this);
    endtask

endclass
endpackage
```

FIFO Sequence Item:

```
fifo_seq_item.sv > {} fifo_seq_item_pkg > fifo_seq_item
package fifo_seq_item_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class fifo_seq_item extends uvm_sequence_item;
`uvm_object_utils(fifo_seq_item)
  rand logic rst_n, wr_en, rd_en;
  rand logic [15:0] data_in;
  logic [15:0] data_out;//
  bit wr_ack, overflow, full, empty, almostfull, almostempty, underflow;

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

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

    function string convert2string_stimulus();
        return $sformatf ("rst_n = %0b ,wr_en= %0b, rd_en= %0b, data_in=%0b , data_out=%0b, wr_ack = %0b ,full= %0b, empty= %0b, almostfull=%0b almostempty = %0b ,overflow= %0b, underfl
        ,rst_n,wr_en, rd_en, data_in,data_out,wr_ack,full,empty,almostfull,almostempty,overflow,underflow);
    endfunction

////////////////////////////////////////////
    int RD_EN_ON_DIST =30;
    int WR_EN_ON_DIST =70;

    constraint reset_enable{
        rst_n dist{0:/ 5 , 1:/95};
    }
    constraint write_enable{
        wr_en dist{1:/ WR_EN_ON_DIST , 0:/(100-WR_EN_ON_DIST)};
    }
    constraint read_enable{
        rd_en dist{1:/ RD_EN_ON_DIST , 0:/(100-RD_EN_ON_DIST)};
    }

endclass

endpackage
```



FIFO Sequence:

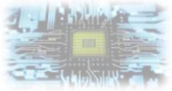
fifo_sequence.sv > {} fifo_sequence_pkg

```
1   package fifo_sequence_pkg;
2   import uvm_pkg::*;
3   import fifo_seq_item_pkg::*;
4   `include "uvm_macros.svh"
5       class fifo_sequence extends uvm_sequence #(fifo_seq_item);
6       `uvm_object_utils(fifo_sequence)
7           fifo_seq_item seq_item;
8
9           function new(string name ="fifo_sequence");
10              super.new(name);
11          endfunction //new()
12
13          task body;
14              seq_item = fifo_seq_item::type_id::create("seq_item");
15              start_item(seq_item);
16              seq_item.rst_n=0;
17              seq_item.data_in=0;
18              seq_item.wr_en=0;
19              seq_item.rd_en=0;
20              finish_item(seq_item);
21
22          repeat(2000)begin
23              seq_item = fifo_seq_item::type_id::create("seq_item");
24              start_item(seq_item);
25              assert(seq_item.randomize());
26              finish_item(seq_item);
27              end
28          endtask
29      endclass
30
31  endpackage
```

FIFO Sequencer:

fifo_sequencer.sv > {} fifo_sequencer_pkg

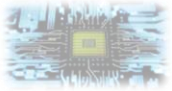
```
1   package fifo_sequencer_pkg;
2   import uvm_pkg::*;
3   import fifo_seq_item_pkg::*;
4   `include "uvm_macros.svh"
5   class fifo_sequencer extends uvm_sequencer #(fifo_seq_item);
6       `uvm_component_utils(fifo_sequencer)
7
8       function new(string name ="fifo_sequencer",uvm_component parent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



FIFO Driver:

fifo_driver.sv > {} fifo_driver_pkg

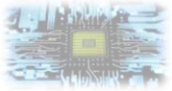
```
1   package fifo_driver_pkg;
2   import fifo_seq_item_pkg::*;
3   import uvm_pkg::*;
4   `include "uvm_macros.svh"
5
6   class fifo_driver extends uvm_driver #(fifo_seq_item);
7       `uvm_component_utils(fifo_driver)
8
9       virtual fifo_if fifo_vif;
10      fifo_seq_item stim_seq_item;
11
12      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_seq_item::type_id::create("stim_seq_item");
20              seq_item_port.get_next_item(stim_seq_item);
21
22              fifo_vif.data_in = stim_seq_item.data_in;
23              fifo_vif.wr_en = stim_seq_item.wr_en;
24              fifo_vif.rd_en = stim_seq_item.rd_en;
25              fifo_vif.rst_n = stim_seq_item.rst_n;
26
27              @(negedge fifo_vif.clk);
28              seq_item_port.item_done();
29              `uvm_info("run_phase",stim_seq_item.convert2string_stimulus(), UVM_HIGH)
30          end
31      endtask
32  endclass
33
34  endpackage
```



FIFO Agent package:

fifo_agent.sv > {} fifo_agent_pkg > fifo_agent

```
package fifo_agent_pkg;

import uvm_pkg::*;
import fifo_driver_pkg::*;
import fifo_config_pkg::*;
import fifo_monitor_pkg::*;
import fifo_sequencer_pkg::*;
import fifo_seq_item_pkg::*;

`include "uvm_macros.svh"

    class fifo_agent extends uvm_agent;
        `uvm_component_utils(fifo_agent)
        fifo_sequencer sqr;
        fifo_monitor mon;
        fifo_driver fifo_driv;
        fifo_config fifo_config_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)::get(this, "","CFG" , fifo_config_cfg))begin
            `uvm_fatal("build_phase" , "Unable to get configuration object");
            end

            fifo_driv=fifo_driver::type_id::create("fifo_driv",this);
            sqr = fifo_sequencer::type_id::create("driver",this);
            mon=fifo_monitor::type_id::create("mon",this);
            agt_ap = new("agt_ap",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            fifo_driv.fifo_vif= fifo_config_cfg.fifo_config_vif;
            mon.fifo_mon_vif = fifo_config_cfg.fifo_config_vif;
            fifo_driv.seq_item_port.connect(sqr.seq_item_export);
            mon.mon_ap.connect(agt_ap);
        endfunction
    endclass
endpackage
```



FIFO ENVIRONMENT:

```
8   package fifo_env_pkg;
9   import fifo_agent_pkg::*;
10  import fifo_scoreboard_pkg::*;
11  import fifo_coverage_pkg::*;
12  import uvm_pkg::*;
13  `include "uvm_macros.svh"
14
15      class fifo_env extends uvm_env;
16
17        `uvm_component_utils(fifo_env)
18
19
20        fifo_scoreboard fifo_sb;
21        fifo_coverage fifo_cvrg;
22        fifo_agent agent;
23
24          function new (string name = "fifo_env",uvm_component parent = null);
25            super.new(name,parent);
26          endfunction
27
28          function void build_phase(uvm_phase phase);
29            super.build_phase(phase);
30            fifo_sb=fifo_scoreboard::type_id::create("fifo_sb",this);
31            agent = fifo_agent::type_id::create("agent",this);
32            fifo_cvrg = fifo_coverage::type_id::create("fifo_cvrg",this);
33          endfunction
34
35          function void connect_phase(uvm_phase phase);
36            super.connect_phase(phase);
37            agent.agt_ap.connect(fifo_sb.sb_export);
38            agent.agt_ap.connect(fifo_cvrg.cov_export);
39
40          endfunction
41      endclass
42  endpackage
```

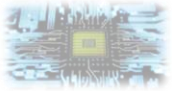
FIFO Config:

fifo_config.sv > { } fifo_config_pkg > fifo_config

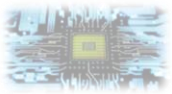
```
1   package fifo_config_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4
5       class fifo_config extends uvm_object;
6           `uvm_object_utils(fifo_config)
7
8           virtual fifo_if fifo_config_vif;
9
10          function new (string name = "fifo_config");
11              super.new(name);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

FIFO Coverage package:

```
1   package fifo_coverage_pkg;
2   import fifo_seq_item_pkg::*;
3   import uvm_pkg::*;
4   `include "uvm_macros.svh"
5
6       class fifo_coverage extends uvm_component;
7           `uvm_component_utils(fifo_coverage);
8
9           uvm_analysis_export #(fifo_seq_item) cov_export;
10          uvm_tlm_analysis_fifo #(fifo_seq_item) cov_fifo;
11          fifo_seq_item item_cov;
12
13
14          covergroup cvr_gp ;
15
16              write_enable    :coverpoint item_cov.wr_en{
17              bins write_1  = {1};
18              bins write_0  = {0};
19              }
20              read_enable     :coverpoint item_cov.rd_en{
21              bins read_1  = {1};
22              bins read_0  = {0};
23              }
24              full_flag       :coverpoint item_cov.full {
25              bins full_1  = {1};
26              bins full_0  = {0};
27              }
28              empty_flag      :coverpoint item_cov.empty{
29              bins empty_1  = {1};
30              bins empty_0  = {0};
31              }
32              almostfull_flag :coverpoint item_cov.almostfull{
33              bins almostfull_1  = {1};
34              bins almostfull_0  = {0};
35              }
36              almostempty_flag:coverpoint item_cov.almostempty {
37              bins almostempty_1  = {1};
38              bins almostempty_0  = {0};
39              }
40              overflow_flag   :coverpoint item_cov.overflow    {
41              bins overflow_1  = {1};
42              bins overflow_0  = {0};
43              }
44              underflow_flag  :coverpoint item_cov.underflow   {
45              bins underflow_1  = {1};
46              bins underflow_0  = {0};
47              }
48              wr_ack_flag     :coverpoint item_cov.wr_ack  {
49              bins wr_ack_1  = {1};
50              bins wr_ack_0  = {0};
51              }
```

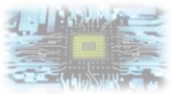


```
51              }
52
53              full:cross write_enable,read_enable,full_flag {
54              ignore_bins ignored_1_will_not_happend =binsof (read_enable.read_1) && binsof (full_flag.full_1);
55              }
56              empty:cross write_enable,read_enable,empty_flag  {
57              ignore_bins ignored_2_will_not_happend =binsof (write_enable.write_1) && binsof (empty_flag.empty_1);
58              }
59              almostfull:cross write_enable,read_enable,almostfull_flag ;
60              almostempty:cross write_enable,read_enable,almostempty_flag;
61              overflow:cross write_enable,read_enable,overflow_flag{
62              ignore_bins ignored_3_will_not_happend =binsof (write_enable.write_0) && binsof (overflow_flag.overflow_1);
63              }
64              underflow:cross write_enable,read_enable,underflow_flag{
65              ignore_bins ignored_4_will_not_happend =binsof (read_enable.read_0) && binsof (underflow_flag.underflow_1);
66              }
67              ack:cross write_enable,read_enable,wr_ack_flag   {
68              ignore_bins ignored_5_will_not_happend =binsof (write_enable.write_0) && binsof (wr_ack_flag.wr_ack_1);
69              }
70          endgroup
71
72          function new (string name = "fifo_coverage",uvm_component parent = null);
73              super.new(name,parent);
74              cvr_gp=new();
75          endfunction
76
77          function void build_phase(uvm_phase phase);
78              super.build_phase(phase);
79              cov_export = new("cov_export",this);
80              cov_fifo = new("cov_fifo",this);
81          endfunction
82
83          function void connect_phase(uvm_phase phase);
84                  super.connect_phase(phase);
85                  cov_export.connect(cov_fifo.analysis_export);
86          endfunction
87
88          task run_phase(uvm_phase phase);
89              super.run_phase(phase);
90              forever begin
91                  cov_fifo.get(item_cov);
92                  cvr_gp.sample();
93              end
94          endtask
95      endclass
96  endpackage
97
```



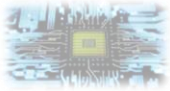
FIFO Scoreboard package:

```
package fifo_scoreboard_pkg;
import uvm_pkg::*;
import fifo_seq_item_pkg::*;
`include "uvm_macros.svh"
    class fifo_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(fifo_scoreboard)
        uvm_analysis_port #(fifo_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(fifo_seq_item) sb_fifo;
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
        fifo_seq_item item_sb;
        logic [FIFO_WIDTH-1:0]  data_out_ref;
        logic full_ref, wr_ack_ref, overflow_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;
        localparam max_fifo_addr = $clog2(FIFO_DEPTH);//----------
        logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;//----------
        logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
        logic [max_fifo_addr:0] count;//----------
        int error_count,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
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(item_sb);
                ref_model(item_sb);
                if((item_sb.rd_en &&(data_out_ref !== item_sb.data_out)) || (full_ref !== item_sb.full) ||(wr_ack_ref !== item_sb.wr_ack)|| (overflow_ref!==item_sb.overflow) ||
                (empty_ref!==item_sb.empty)|| (almostfull_ref!==item_sb.almostfull)|| (almostempty_ref!==item_sb.almostempty)|| (underflow_ref!==item_sb.underflow))begin
                    `uvm_error("run_phase", $sformatf("comparison failed, transaction received by the dut:  %s while the reference out :  data_out_ref=%0b, wr_ack_ref=%0b,overflow_ref=%0b, underflow_ref=%0b,
                    ,item_sb.convert2string(),data_out_ref, wr_ack_ref, overflow_ref, underflow_ref, full_ref, empty_ref, almostfull_ref, almostempty_ref,overflow_ref,underflow_ref));
                    error_count++;
                end
                else correct_count++;
            end
        endtask
```



```
        endtask
        task ref_model(fifo_seq_item item_chk);
            if (!item_chk.rst_n) begin
                rd_ptr = 0;
                wr_ptr = 0;
                count = 0;

                wr_ack_ref = 0;
                underflow_ref = 0;
                overflow_ref = 0;

                full_ref = 0;
                empty_ref = 1;
                almostfull_ref = 0;
                almostempty_ref = 0;
            end
            else begin
                // Sread and write
                if(item_chk.wr_en && !full_ref)begin //write
                    mem[wr_ptr] = item_chk.data_in;
                    wr_ack_ref = 1;
                    wr_ptr ++;
                    count++;
                    overflow_ref=0;
                end
                else if(item_chk.wr_en && full_ref) begin
                        overflow_ref=1;
                        wr_ack_ref = 0;
                end
                else begin
                        wr_ack_ref = 0;
                        overflow_ref=0;
                end

                if(item_chk.rd_en && !empty_ref)begin //read
                        data_out_ref = mem[rd_ptr];
                        rd_ptr++;
                        underflow_ref = 0;
                        count--;
                end
                else if(item_chk.rd_en && empty_ref) underflow_ref=1;
                else underflow_ref=0;

            full_ref = (count == FIFO_DEPTH)? 1 : 0;
            empty_ref = (count == 0)? 1 : 0;
            almostfull_ref = (count == FIFO_DEPTH-1)? 1 : 0;
            almostempty_ref = (count == 1)? 1 : 0;
            end
        endtask
```

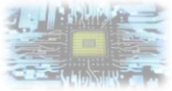
```

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase",$sformatf("correct count = %0d",correct_count),UVM_MEDIUM);
            `uvm_info("report_phase",$sformatf("error count = %0d",error_count),UVM_MEDIUM);

        endfunction
    endclass
endpackage
```



Monitor:

fifo_monitor.sv > {} fifo_monitor_pkg > fifo_monitor

```
package fifo_monitor_pkg;

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

    class fifo_monitor extends uvm_monitor;
        `uvm_component_utils(fifo_monitor)
        virtual fifo_if fifo_mon_vif;
        fifo_seq_item fifo_mon_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
                fifo_mon_item = fifo_seq_item::type_id::create("fifo_mon_item");
                @(negedge fifo_mon_vif.clk);
                fifo_mon_item.data_in = fifo_mon_vif.data_in;
                fifo_mon_item.wr_en = fifo_mon_vif.wr_en;
                fifo_mon_item.rst_n = fifo_mon_vif.rst_n;
                fifo_mon_item.rd_en = fifo_mon_vif.rd_en;

                fifo_mon_item.data_out = fifo_mon_vif.data_out;
                fifo_mon_item.wr_ack = fifo_mon_vif.wr_ack;
                fifo_mon_item.full = fifo_mon_vif.full;
                fifo_mon_item.almostfull = fifo_mon_vif.almostfull;

                fifo_mon_item.empty = fifo_mon_vif.empty;
                fifo_mon_item.almostempty = fifo_mon_vif.almostempty;
                fifo_mon_item.overflow = fifo_mon_vif.overflow;
                fifo_mon_item.underflow = fifo_mon_vif.underflow;

                mon_ap.write(fifo_mon_item);
                `uvm_info("run_phase",fifo_mon_item.convert2string(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```



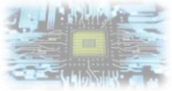
Assertions:

fifo_sva.sv > fifo_sva

```
module fifo_sva(fifo_if.DUT fifoif);
parameter FIFO_DEPTH = 8;

always_comb begin
    if(!fifoif.rst_n)begin
        assert final (!DUT.count && !DUT.wr_ptr && !DUT.rd_ptr && !fifoif.wr_ack && !fifoif.overflow && !fifoif.underflow);
        end
    assert final (fifoif.full == (DUT.count == FIFO_DEPTH)? 1 : 0);
    assert final (fifoif.empty == (DUT.count == 0)? 1 : 0);
    assert final (fifoif.almostfull == (DUT.count == FIFO_DEPTH-1)? 1 : 0);
    assert final (fifoif.almostempty == (DUT.count == 1)? 1 : 0);

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

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

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

    property wr_ptr_p;
        @(posedge fifoif.clk)  disable iff(!fifoif.rst_n) (fifoif.wr_en && !fifoif.full) |=> (DUT.wr_ptr == (($past(DUT.wr_ptr)+1)%FIFO_DEPTH));
    endproperty
assert property(wr_ptr_p); cover property(wr_ptr_p);

    property rd_ptr_p;
        @(posedge fifoif.clk)  disable iff(!fifoif.rst_n) (fifoif.rd_en && !fifoif.empty) |=> (DUT.rd_ptr == (($past(DUT.rd_ptr)+1)%FIFO_DEPTH));
    endproperty
assert property(rd_ptr_p); cover property(rd_ptr_p);



endmodule

```

Verification Plan:

Label	Description	Stimulus generation	Functional coverage	Functionality check
Reset	When reset is asserted, all signals are resetted	rst_n =0	Rst_n active 5% inactive95%	Output is correct
write	When reset is deasserted, wr_en is enabled	Rst_n=1, wr_en=1	Cross coverage	Output is correct
read	When reset is deasserted, rd_en is enabled	Rst_n=1, rd_en=1	Cross coverage	Output is correct
Read_write	When reset is deasserted, wr_en & rd_en are enabled	Rst_n=1, rd_en=1 & wr_en=1	Cross coverage	Output is correct



Do file:

run.do

```
vlib work
vlog -f src_files.list +cover
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover

add wave -position insertpoint  \
sim:/top/DUT/FIFO_WIDTH \
sim:/top/DUT/FIFO_DEPTH \
sim:/top/DUT/max_fifo_addr \
sim:/top/DUT/mem \
sim:/top/DUT/wr_ptr \
sim:/top/DUT/rd_ptr \
sim:/top/DUT/count

add wave -position insertpoint  \
sim:/top/fifoif/almostempty \
sim:/top/fifoif/almostfull \
sim:/top/fifoif/clk \
sim:/top/fifoif/data_in \
sim:/top/fifoif/data_out \
sim:/top/fifoif/empty \
sim:/top/fifoif/FIFO_DEPTH \
sim:/top/fifoif/FIFO_WIDTH \
sim:/top/fifoif/full \
sim:/top/fifoif/overflow \
sim:/top/fifoif/rd_en \
sim:/top/fifoif/rst_n \
sim:/top/fifoif/underflow \
sim:/top/fifoif/wr_ack \
sim:/top/fifoif/wr_en
coverage save top.ucdb -onexit
run -all
vcover report top.ucdb -details -annotate -all -output coverage_report.txt
```

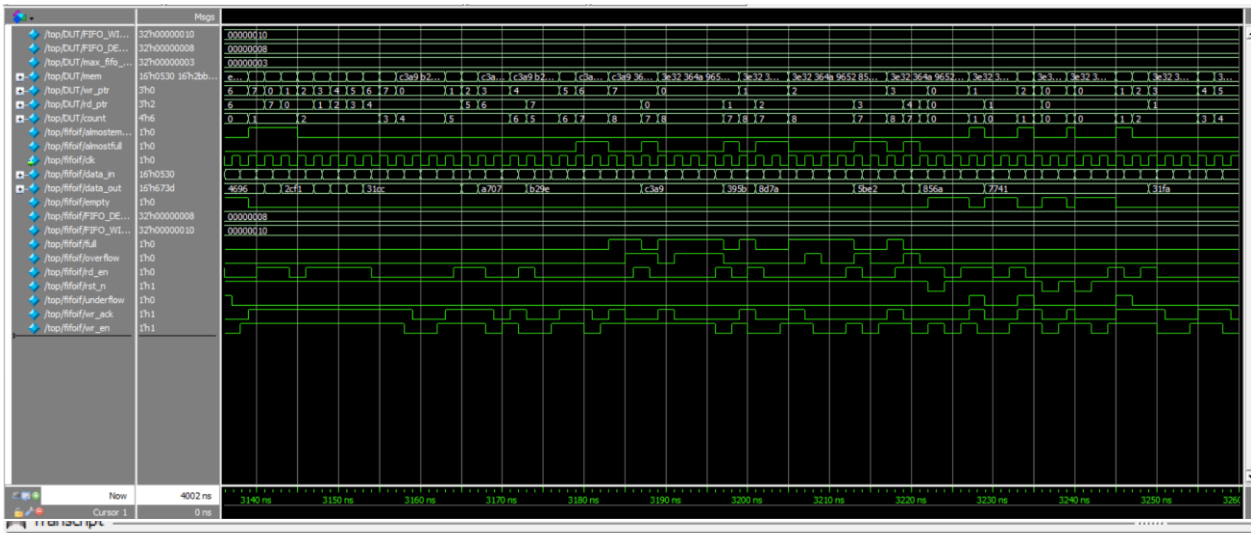
Source files:

src_files.list

```
fifo_if.sv
FIFO.sv
fifo_seq_item.sv
fifo_sequence.sv
fifo_sequencer.sv
fifo_config.sv
fifo_driver.sv
fifo_agent.sv
fifo_coverage_uvm.sv
fifo_scoreboard_uvm.sv
fifo_env.sv
fifo_test.sv
fifo_sva.sv
top.sv
```

Questasim:

Waveform:



```
# UVM_INFO fifo_scoreboard_uvm.sv(102) @ 4002: uvm_test_top.env.fifo_sb [report_phase] correct count = 2001
# UVM_INFO fifo_scoreboard_uvm.sv(103) @ 4002: uvm_test_top.env.fifo_sb [report_phase] error count = 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    8
# 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]     2
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 4002 ns  Iteration: 61  Instance: /top
```

Code coverage:

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      29        29         0   100.00%

--------------------------------Statement Details--------------------------------
```

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

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

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10         2         8    20.00%

  --------------------------------Branch Details--------------------------------
```

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      2         0         2     0.00%

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

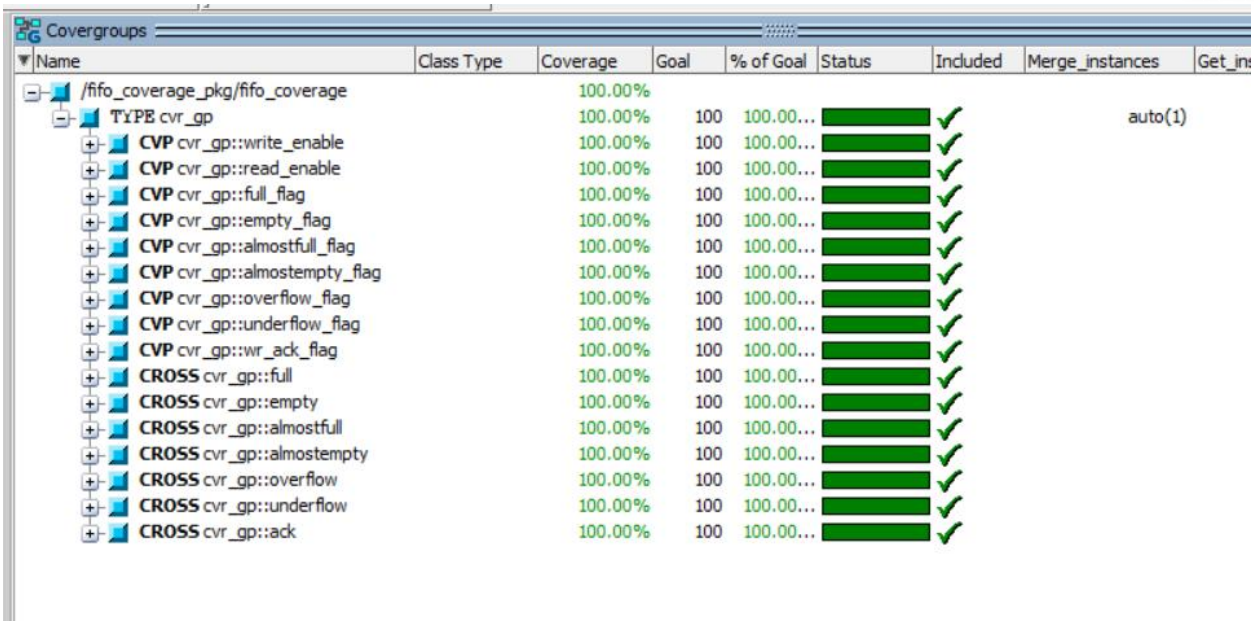
Functional coverage:

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memc
/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	-	
/fifo_sequence_pkg::fifo_sequence::body/#ublk#40571367#22/immed__25	Immediate	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/assert__write_ack_p	Concurrent	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/assert__overflow_p	Concurrent	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/assert__underflow_p	Concurrent	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/assert__wr_ptr_p	Concurrent	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/assert__rd_ptr_p	Concurrent	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__8	Immediate	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__9	Immediate	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__10	Immediate	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__11	Immediate	SVA	on	0	1	-	
/top/DUT/fifo_sva_inst/#ublk#265645057#4/#ublk#265645057#5/immed__6	Immediate	SVA	on	0	1	-	

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/top/DUT/fifo_sva_inst/cover__overflow_p	SVA	✓	Off	316	1	Unlimited	1	100%		✓
/top/DUT/fifo_sva_inst/cover__rd_ptr_p	SVA	✓	Off	479	1	Unlimited	1	100%		✓
/top/DUT/fifo_sva_inst/cover__underflow_p	SVA	✓	Off	47	1	Unlimited	1	100%		✓
/top/DUT/fifo_sva_inst/cover__wr_ptr_p	SVA	✓	Off	932	1	Unlimited	1	100%		✓
/top/DUT/fifo_sva_inst/cover__write_ack_p	SVA	✓	Off	932	1	Unlimited	1	100%		✓



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_ins
/fifo_coverage_pkg/fifo_coverage		100.00%						
TYPE cvr_gp		100.00%	100	100.00...			auto(1)	
CVP cvr_gp::write_enable		100.00%	100	100.00...				
CVP cvr_gp::read_enable		100.00%	100	100.00...				
CVP cvr_gp::full_flag		100.00%	100	100.00...				
CVP cvr_gp::empty_flag		100.00%	100	100.00...				
CVP cvr_gp::almostfull_flag		100.00%	100	100.00...				
CVP cvr_gp::almostempty_flag		100.00%	100	100.00...				
CVP cvr_gp::overflow_flag		100.00%	100	100.00...				
CVP cvr_gp::underflow_flag		100.00%	100	100.00...				
CVP cvr_gp::wr_ack_flag		100.00%	100	100.00...				
CROSS cvr_gp::full		100.00%	100	100.00...				
CROSS cvr_gp::empty		100.00%	100	100.00...				
CROSS cvr_gp::almostfull		100.00%	100	100.00...				
CROSS cvr_gp::almostempty		100.00%	100	100.00...				
CROSS cvr_gp::overflow		100.00%	100	100.00...				
CROSS cvr_gp::underflow		100.00%	100	100.00...				
CROSS cvr_gp::ack		100.00%	100	100.00...				

```
Assertion Coverage:
    Assertions                      10        10         0   100.00%
  ----------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
  ----------------------------------------------------------------------
/top/DUT/fifo_sva_inst/assert__rd_ptr_p
                     fifo_sva.sv(37)                     0          1
/top/DUT/fifo_sva_inst/assert__wr_ptr_p
                     fifo_sva.sv(32)                     0          1
/top/DUT/fifo_sva_inst/assert__underflow_p
                     fifo_sva.sv(27)                     0          1
/top/DUT/fifo_sva_inst/assert__overflow_p
                     fifo_sva.sv(22)                     0          1
/top/DUT/fifo_sva_inst/assert__write_ack_p
                     fifo_sva.sv(17)                     0          1
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__11
                     fifo_sva.sv(11)                     0          1
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__10
                     fifo_sva.sv(10)                     0          1
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__9
                     fifo_sva.sv(9)                      0          1
/top/DUT/fifo_sva_inst/#ublk#265645057#4/immed__8
                     fifo_sva.sv(8)                      0          1
/top/DUT/fifo_sva_inst/#ublk#265645057#4/#ublk#265645057#5/immed__6
                     fifo_sva.sv(6)                      0          1

Directive Coverage:
    Directives                      5         5         0   100.00%
```