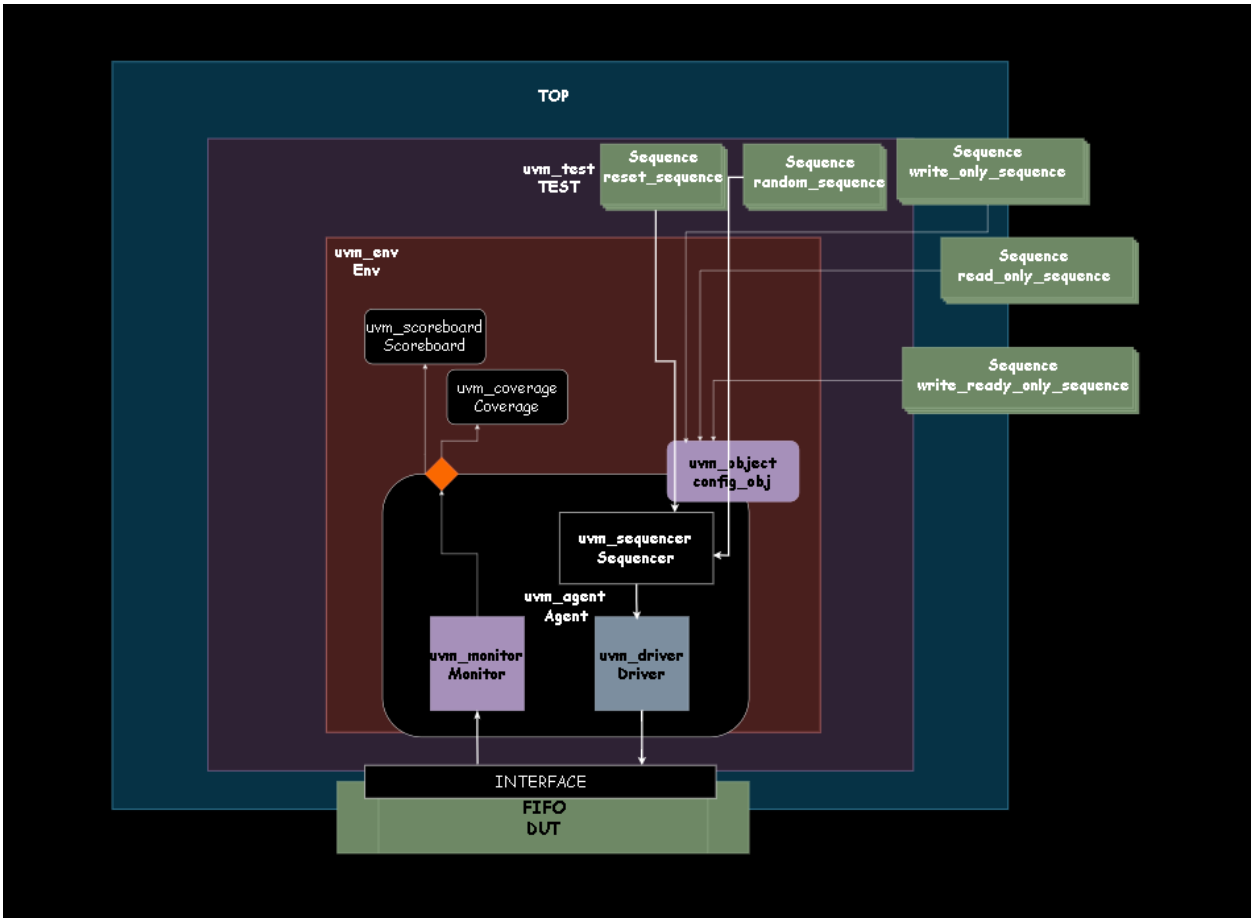
FINAL PROJECT – FIFO UVM



Top Module:

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


module TOP();
bit clk=0;
FIFO_IF IF(clk);
FIFO DUT(
    .data_in(IF.data_in), .wr_en(IF.wr_en), .rd_en(IF.rd_en),
    .clk(IF.clk), .rst_n(IF.rst_n), .full(IF.full),
    .empty(IF.empty), .almostfull(IF.almostfull), .almostempty(IF.almostempty),
    .wr_ack(IF.wr_ack), .overflow(IF.overflow), .underflow(IF.underflow),
    .data_out(IF.data_out)
 );


always  begin
    #5 clk <= ~clk;
end

initial begin

    uvm_config_db#(virtual FIFO_IF)::set(uvm_root::get(),"*","IF",IF);
    //uvm_config_db#(virtual shift_reg_if)::set(uvm_root::get(),"uvm_test_top","IF",srif);
    run_test("Test");
end

endmodule

```

Test:

```
package test_pkg;
`include "uvm_macros.svh"

//`include "main_sequence.sv"
//`include "reset_sequence.sv"

import uvm_pkg::*;
import env_pkg::*;

 import reset_sequence_pkg::*;
 import write_only_sequence_pkg::*;
 import   read_only_sequence_pkg::*;
 import   write_read_sequence_pkg::*;
 import   random_sequence_pkg::*;
 import config_obj_pkg::*;
```

```
class Test extends uvm_test;
`uvm_component_utils(Test)
    config_obj c_obj;
    Env env;
    Reset_Sequence reset_seq;
    Write_Only_Sequence wos;
    Read_Only_Sequence ros;
    Write_Read_Sequence wrs;
    Random_Sequence rs;



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

```
    function void build_phase(uvm_phase phase);
    super.build_phase(phase);

    c_obj = config_obj::type_id::create("config_obj");

    if(!uvm_config_db#(virtual FIFO_IF)::get(this,"","IF",c_obj.vif))
    begin
        `uvm_fatal("build_phase","Test - Unable to get the virtual interface")
    end
    uvm_config_db#(config_obj)::set(this,"*","agent_config_obj",c_obj);
    env = Env::type_id::create("Env",this);

    reset_seq = Reset_Sequence::type_id::create("reset_seq",this);
    wos = Write_Only_Sequence::type_id::create("wos",this);
    ros = Read_Only_Sequence::type_id::create("ros",this);
    wrs = Write_Read_Sequence::type_id::create("wrs",this);
    rs =  Random_Sequence::type_id::create("rs",this);


    endfunction

    task run_phase(uvm_phase phase);
     super.run_phase(phase);
     phase.raise_objection(this);
/*
wos.start(env.agent.sqr);
ros.start(env.agent.sqr);
wrs.start(env.agent.sqr);
rs.start(env.agent.sqr);

*/
```

```
*/

  //   #100; `uvm_info("run_phase","start reset sequence!",UVM_NONE)
     reset_seq.start(env.agent.sqr);
    //  #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
      wos.start(env.agent.sqr);
    //  #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
      ros.start(env.agent.sqr);
     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
      wrs.start(env.agent.sqr);
     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
      rs.start(env.agent.sqr);
      phase.drop_objection(this);
     endtask
endclass //className extends superClass



endpackage

```

Reset Sequence:

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

Random Sequence:

```
package random_sequence_pkg;
`include "uvm_macros.svh"
import uvm_pkg::*;
//`include "Sequence_Item.sv"
import sequence_item_pkg::*;
class Random_Sequence extends uvm_sequence #(Sequence_Item);
`uvm_object_utils(Random_Sequence)
//Sequence_Item seq_item;
function new(string name = "Random_Sequence");
super.new(name);
endfunction

task body;
//seq_item = Sequence_Item::type_id::create("seq_item");

repeat(10000)
begin
Sequence_Item seq_item;
seq_item = Sequence_Item::type_id::create("seq_item");
start_item(seq_item);
seq_item.rst_n = 1;
assert (seq_item.randomize());
finish_item(seq_item);
end

endtask

endclass
endpackage
```

Read Only Sequence

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

Write Only Sequence

Final Project > write_only_sequence.sv

```
1  package write_only_sequence_pkg;
2  `include "uvm_macros.svh"
3  import uvm_pkg::*;
4  //`include "Sequence_Item.sv"
5  import sequence_item_pkg::*;
6  class Write_Only_Sequence extends uvm_sequence #(Sequence_Item);
7  `uvm_object_utils(Write_Only_Sequence)
8  //Sequence_Item seq_item;
9  function new(string name = "Write_Only_Sequence");
10 super.new(name);
11 endfunction
12
13 task body;
14 //seq_item = Sequence_Item::type_id::create("seq_item");
15
16 repeat(1000)
17 begin
18 Sequence_Item seq_item;
19 seq_item = Sequence_Item::type_id::create("seq_item");
20 start_item(seq_item);
21 seq_item.rst_n = 1;
22 assert (seq_item.randomize());
23 seq_item.wr_en=1;
24 seq_item.rd_en=0;
25 finish_item(seq_item);
26 end
27
28 endtask
29
30 endclass
31 endpackage
```

Write Read Only Sequence

```
package write_read_sequence_pkg;
`include "uvm_macros.svh"
import uvm_pkg::*;
//`include "Sequence_Item.sv"
import sequence_item_pkg::*;
class Write_Read_Sequence extends uvm_sequence #(Sequence_Item);
`uvm_object_utils(Write_Read_Sequence)
//Sequence_Item seq_item;
function new(string name = "Write_Read_Sequence");
super.new(name);
endfunction

task body;
//seq_item = Sequence_Item::type_id::create("seq_item");

repeat(1000)
begin
Sequence_Item seq_item;
seq_item = Sequence_Item::type_id::create("seq_item");
start_item(seq_item);
seq_item.rst_n = 1;
seq_item.data_in =0;
seq_item.wr_en=1;
seq_item.rd_en=1;

finish_item(seq_item);
end

repeat(1000)
begin
Sequence_Item seq_item;
seq_item = Sequence_Item::type_id::create("seq_item");
start_item(seq_item);
seq_item.rst_n = 1;
//assert (seq_item.randomize());
seq_item.data_in =0;
seq_item.wr_en=0;
seq_item.rd_en=0;

finish_item(seq_item);
end
endtask
endclass
endpackage
```

Environment:

```
2
3   package env_pkg;
4
5
6   `include "uvm_macros.svh"
7   import uvm_pkg::*;
8   import agent_pkg::*;
9   import coverage_pkg::*;
10  import scoreboard_pkg::*;
11
12  //import driver_pkg::*;
13
14
15  class Env extends uvm_env;
16  `uvm_component_utils(Env)
17
18  Agent agent;
19  Scoreboard sb;
20  Coverage cov;
21  function new(string name = "Env",uvm_component parent = null );
22  super.new(name,parent);
23
24  endfunction
25  function void build_phase(uvm_phase phase);
26  super.build_phase(phase);
27  agent = Agent::type_id::create("agent",this);
28  sb = Scoreboard::type_id::create("sb",this);
29  cov = Coverage::type_id::create("cov",this);
30  endfunction
31
32  function void connect_phase(uvm_phase phase);
33  super.connect_phase(phase);
34
35  agent.agt_ap.connect(sb.sb_export);
36  agent.agt_ap.connect(cov.cov_export);
37
38  endfunction
39
40  endclass
41  endpackage
```

Scoreboard

```
package scoreboard_pkg;
`include "uvm_macros.svh"
import uvm_pkg::*;
import sequence_item_pkg::*;
class Scoreboard extends uvm_scoreboard;
`uvm_component_utils(Scoreboard)

uvm_analysis_export#(Sequence_Item) sb_export;
uvm_tlm_analysis_fifo#(Sequence_Item) sb_fifo;
Sequence_Item fifo_transaction;

   localparam FIFO_WIDTH = 16;
   localparam FIFO_DEPTH = 8;
   localparam max_fifo_addr = $clog2(FIFO_DEPTH);


    logic [FIFO_WIDTH-1:0] data_out_ref;
    logic wr_ack_ref, overflow_ref;
    logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;

    logic [max_fifo_addr-1:0] wr_ptr_ref;
    logic [max_fifo_addr-1:0] rd_ptr_ref;
    logic [max_fifo_addr:0] count_ref;


    logic [FIFO_WIDTH-1 : 0] mem_ref [FIFO_DEPTH];

    int failed = 0;
    int passed = 0;


function new(string name = "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);
```

```
  super.connect_phase(phase);
  sb_export.connect(sb_fifo.analysis_export);


  endfunction

  function void reference_model(Sequence_Item trans);
      if (!trans.rst_n) begin
          wr_ptr_ref = 0;
          wr_ack_ref=0;
          overflow_ref=0;//fault was dectected here
          data_out_ref= 0;//fault was dectected here
      end
      else if (trans.wr_en && count_ref < 8) begin
          mem_ref[wr_ptr_ref] = trans.data_in;
          wr_ack_ref = 1;
          wr_ptr_ref = wr_ptr_ref + 1;
          overflow_ref = 0;//fault was dectected here
      end
      else begin
          wr_ack_ref = 0;
          if (full_ref & trans.wr_en)
              overflow_ref = 1;
          else
              overflow_ref = 0;
      end


      if (!trans.rst_n) begin
          rd_ptr_ref = 0;
          data_out_ref= 0;//fault was dectected here
          underflow_ref=0;//fault was dectected here
      end
      else if (trans.rd_en && count_ref != 0) begin
          data_out_ref = mem_ref[rd_ptr_ref];
          rd_ptr_ref = rd_ptr_ref + 1;
          underflow_ref=0;//fault was dectected here
      end
      else begin
      if(empty_ref&&trans.rd_en)
          underflow_ref=1;
      else
```

```
    else
        underflow_ref=0;
    end


    if (!trans.rst_n) begin
        count_ref = 0;
        data_out_ref= 0;//fault was dectected here
    end
    else begin
        if  ( ({trans.wr_en, trans.rd_en} == 2'b10) && !full_ref) // if wren=1 and count_ref != 8
            count_ref = count_ref + 1;
        else if ( ({trans.wr_en, trans.rd_en} == 2'b01) && !empty_ref)// if rden=1 and count_ref != 0
            count_ref = count_ref - 1;
            else if ({trans.wr_en,trans.rd_en}==2'b11) begin//fault was dectected here
            if(full_ref)
            count_ref = count_ref-1;
            else if (empty_ref)
            count_ref = count_ref+1;
            end
    end

 full_ref = (count_ref == FIFO_DEPTH)? 1 : 0;
 empty_ref = (count_ref == 0)? 1 : 0;
 almostfull_ref = (count_ref == FIFO_DEPTH-1)? 1 : 0; //fault was dectected here
 almostempty_ref = (count_ref == 1)? 1 : 0;

//assign underflow_ref = (empty_ref && rd_en)? 1 : 0; //fault was dectected here

endfunction


function void check_data(Sequence_Item trans);

reference_model(trans);


assert(trans.data_out === data_out_ref && trans.wr_ack === wr_ack_ref && trans.overflow === overflow_ref && trans.full === full_ref && trans
&& trans.almostempty === almostempty_ref && underflow_ref === trans.underflow)
begin
    passed++;
```

```
126   end
127   else
128   begin
129   $error("test failed!");
130       failed++;
131   end
132
133
134
135   endfunction
136
137   task run_phase(uvm_phase phase);
138   super.run_phase(phase);
139   forever begin
140       sb_fifo.get(fifo_transaction);
141       check_data(fifo_transaction);
142   end
143   endtask
144
145   function void phase_ready_to_end(uvm_phase phase);
146   super.phase_ready_to_end(phase);
147   $display("No. Of Passed Cases: %d, No. Of Failed Cases: %d",passed,failed);
148
149   endfunction
150
151   endclass
152
153   endpackage
```

Coverage

```
package coverage_pkg;
`include "uvm_macros.svh"
import sequence_item_pkg::*;
import uvm_pkg::*;
class Coverage extends uvm_component;
`uvm_component_utils(Coverage)
uvm_analysis_export #(Sequence_Item) cov_export;
uvm_tlm_analysis_fifo #(Sequence_Item) cov_fifo;
Sequence_Item F_cvg_txn;

covergroup cg;
option.per_instance = 1;
wr_en_cp: coverpoint F_cvg_txn.wr_en;
rd_en_cp: coverpoint F_cvg_txn.rd_en;

wr_ack_cp: coverpoint F_cvg_txn.wr_ack;

overflow_cp: coverpoint F_cvg_txn.overflow;
underflow_cp: coverpoint F_cvg_txn.underflow;

full_cp: coverpoint F_cvg_txn.full;
empty_cp: coverpoint F_cvg_txn.empty;

almostfull_cp: coverpoint F_cvg_txn.almostfull;
almostempty_cp: coverpoint F_cvg_txn.almostempty;

wr__rd__ack: cross wr_en_cp,rd_en_cp,wr_ack_cp
{
    ignore_bins ign_ack_without_wr = binsof(wr_en_cp) intersect {0} && binsof(wr_ack_cp) intersect{1};
}

wr__rd__overflow:cross wr_en_cp,rd_en_cp,overflow_cp
{
    ignore_bins ign_overflow_without_wr = binsof(wr_en_cp) intersect {0} && binsof(overflow_cp) intersect{1};
}

wr__rd__underflow:cross wr_en_cp,rd_en_cp,underflow_cp
{
    ignore_bins ign_underflow_without_rd = binsof(rd_en_cp) intersect {0} && binsof(underflow_cp) intersect{1};
}

wr__rd__full:cross wr_en_cp,rd_en_cp,full_cp
{
    ignore_bins ign_full_without_wr = binsof(wr_en_cp) intersect {0} && binsof(full_cp) intersect{1};
    ignore_bins ign_full_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(full_cp) intersect{1};
```

```
}
wr__rd__empty:cross wr_en_cp,rd_en_cp,empty_cp
{
        ignore_bins ign_empty_without_rd = binsof(rd_en_cp) intersect {0} && binsof(empty_cp) intersect{1};
        ignore_bins ign_empty_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(empty_cp) intersect{1};
}


wr__rd__almostfull:cross wr_en_cp,rd_en_cp,almostfull_cp;
wr__rd__almostempty:cross wr_en_cp,rd_en_cp,almostempty_cp;


endgroup

//CG cg;

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

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

function void connect_phase(uvm_phase phase);

    super.connect_phase(phase);
    cov_export.connect(cov_fifo.analysis_export);

endfunction


task run_phase(uvm_phase phase);
super.run_phase(phase);

forever
begin
```

```
  forever
  begin
      cov_fifo.get(F_cvg_txn);
      cg.sample();

  end

  endtask
  endclass
  endpackage
```

Agent

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

import sequence_item_pkg::*;
import driver_pkg::*;
import monitor_pkg::*;
import config_obj_pkg::*;
import sequencer_pkg::*;


class Agent extends uvm_agent;
`uvm_component_utils(Agent)
config_obj c_obj;
Sequencer sqr;
Driver driver;
Monitor monitor;
uvm_analysis_port #(Sequence_Item) agt_ap;
function new(string name = "Agent", uvm_component parent = null);
super.new(name,parent);
endfunction


function void build_phase(uvm_phase phase);
super.build_phase(phase);
sqr = Sequencer::type_id::create("sqr" ,this);
if(!uvm_config_db#(config_obj)::get(this,"","agent_config_obj",c_obj))//got from
begin
`uvm_fatal("build_phase","Driver - Unable to get the config object")
end

driver = Driver::type_id::create("driver",this);
monitor = Monitor::type_id::create("monitor",this);
agt_ap = new("agt_ap",this);
endfunction

task run_phase(uvm_phase phase);
super.run_phase(phase);
endtask

function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
driver.seq_item_port.connect(sqr.seq_item_export);
monitor.IF = c_obj.vif;
```

```
48  endfunction
49
50
51
52
53  endclass
54
55
56  endpackage
57
58
```

Monitor

```
Final Project > = monitor_pkg.sv
1   package monitor_pkg;
2   `include "uvm_macros.svh"
3   import uvm_pkg::*;
4   import sequence_item_pkg::*;
5   //import config_obj_pkg::*;
6   class Monitor extends uvm_monitor;
7   `uvm_component_utils(Monitor)
8   virtual FIFO_IF IF;
9   //config_obj c_obj;
10  Sequence_Item fifo_transaction;
11
12  uvm_analysis_port#(Sequence_Item) mon_ap;
13  function new(string name = "monitor", uvm_component parent = null);
14
15  super.new(name,parent);
16
17  endfunction
18
19
20  function void build_phase(uvm_phase phase);
21  super.build_phase(phase);
22  mon_ap = new("mon_ap",this);
23
24
25  endfunction
26
27  function void connect_phase(uvm_phase phase);
28  super.connect_phase(phase);
29  endfunction
30
31  task run_phase(uvm_phase phase);
32  super.run_phase(phase);
33  forever
34  begin
35      fifo_transaction = Sequence_Item::type_id::create("fifo_transaction");
36      @(posedge IF.clk)
37     fifo_transaction.data_in  = IF.data_in;
38
39     fifo_transaction.rst_n = IF.rst_n;
40
41     fifo_transaction.wr_en = IF.wr_en;
42     fifo_transaction.rd_en = IF.rd_en;
43
44
45   @(negedge IF.clk) ;
46     fifo_transaction.up_ack = IF.up_ack;
```

```
@(negedge IF.clk) ;
    fifo_transaction.wr_ack = IF.wr_ack;

    fifo_transaction.overflow  = IF.overflow;
    fifo_transaction.underflow = IF.underflow;

    fifo_transaction.data_out = IF.data_out;
    fifo_transaction.full  = IF.full;
    fifo_transaction.empty = IF.empty;

    fifo_transaction.almostfull = IF.almostfull;
    fifo_transaction.almostempty = IF.almostempty;

    fifo_transaction.underflow  = IF.underflow;
    mon_ap.write(fifo_transaction);


// fork

//     fifo__coverage.sample_data(fifo_transaction);
//     begin

//     fifo_scoreboard.check_data(fifo_transaction);
//     end
// join


    //`uvm_info("run_phase","monitor running!",UVM_NONE)

end
endtask

endclass


endpackage
```

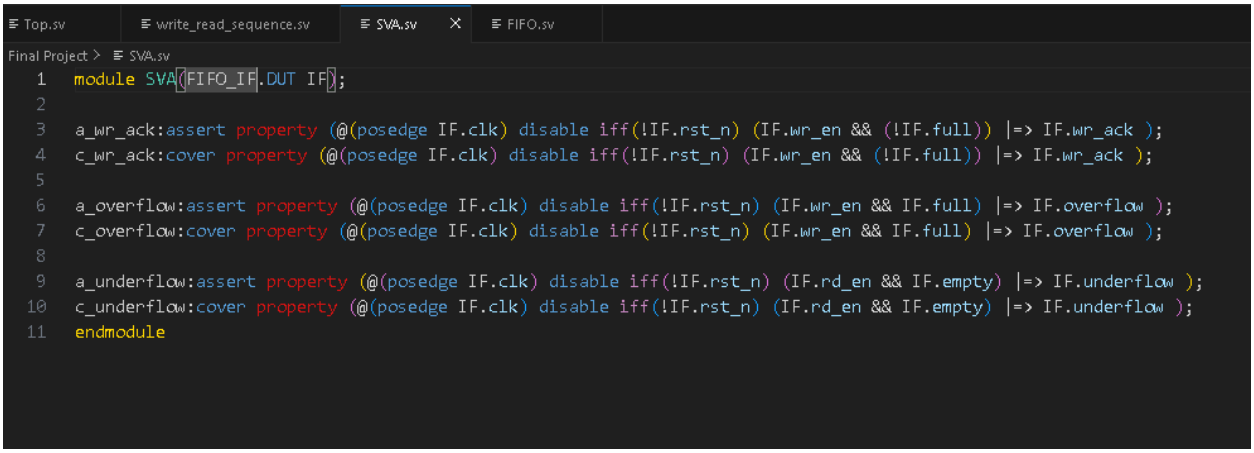
```
    mon_ap.write(item);




    //mon_ap.write(item);
end
endtask

endclass
endpackage
```

Driver

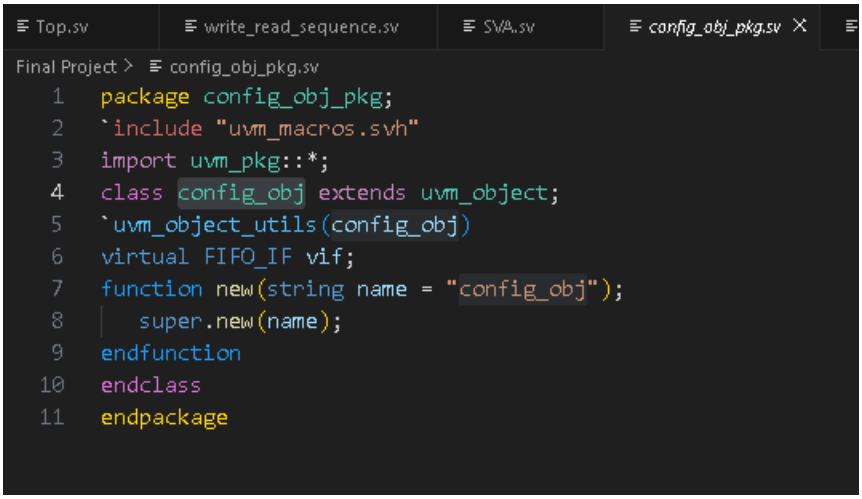
```
package driver_pkg;
`include "uvm_macros.svh"
//`include "Sequence_Item.sv"
import sequence_item_pkg::*;
import uvm_pkg::*;
class Driver extends uvm_driver #(Sequence_Item);
`uvm_component_utils(Driver)
virtual FIFO_IF IF;
//config_obj c_obj;
Sequence_Item fifo_transaction;
function new(string name = "Driver", uvm_component parent = null);
super.new(name,parent);
endfunction

function void build_phase(uvm_phase phase);
super.build_phase(phase);
//c_obj = config_obj::type_id::create("config_obj");

endfunction


function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
//vif = c_obj.vif;
endfunction


task run_phase(uvm_phase phase);
super.run_phase(phase);
    IF.data_in =   0 ;
    IF.rst_n   =   0 ;
    IF.wr_en   =   0 ;
    IF.rd_en   =   0 ;
forever begin
fifo_transaction = Sequence_Item::type_id::create("fifo_transaction");
seq_item_port.get_next_item(fifo_transaction);
@(negedge IF.clk)
    //fifo_transaction.randomize();

    IF.data_in =   fifo_transaction.data_in ;
    IF.rst_n   =   fifo_transaction.rst_n ;
    IF.wr_en   =   fifo_transaction.wr_en ;
    IF.rd_en   =   fifo_transaction.rd_en ;
```

```
  //@(negedge vif.clk)
  seq_item_port.item_done();

  //`uvm_info("run_phase","driver running!",UVM_NONE)
  //$display("%s",seq_item.convert2string());

  end
  endtask

  endclass
  endpackage
```

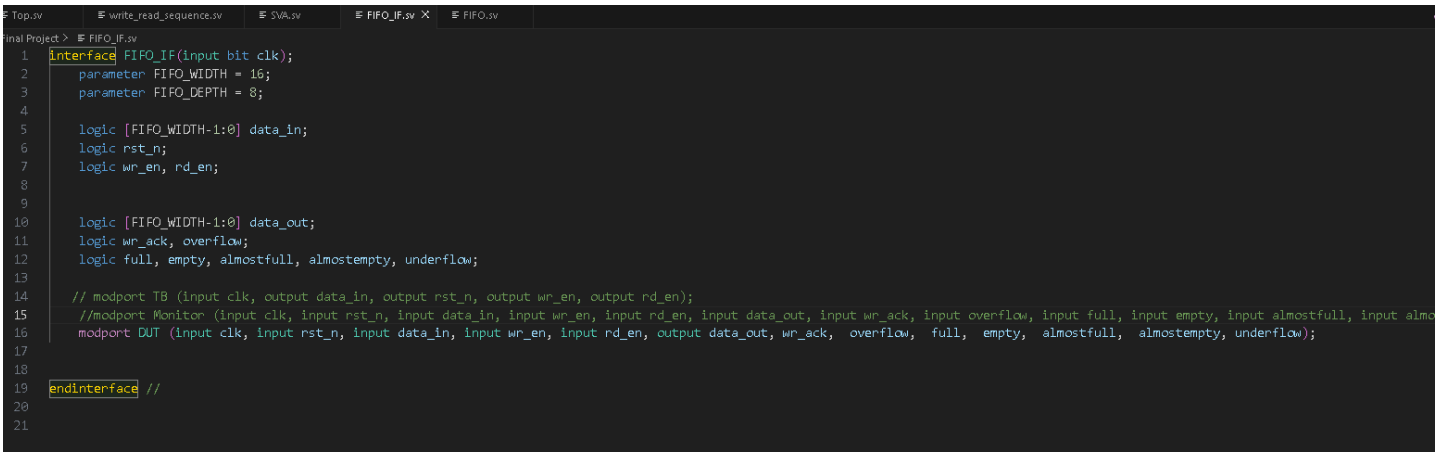
SV Assertion



Config_Obj



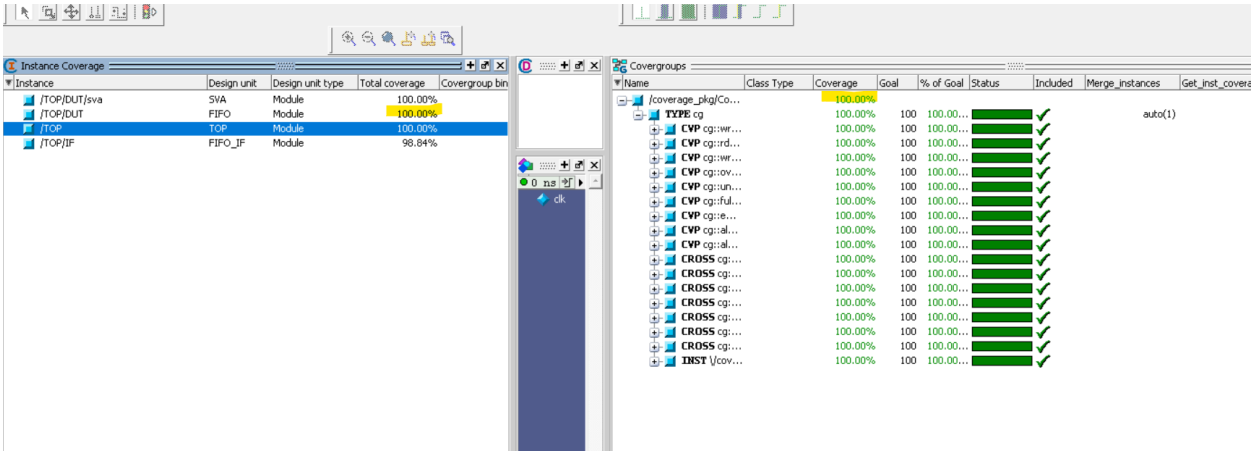
FIFO_IF



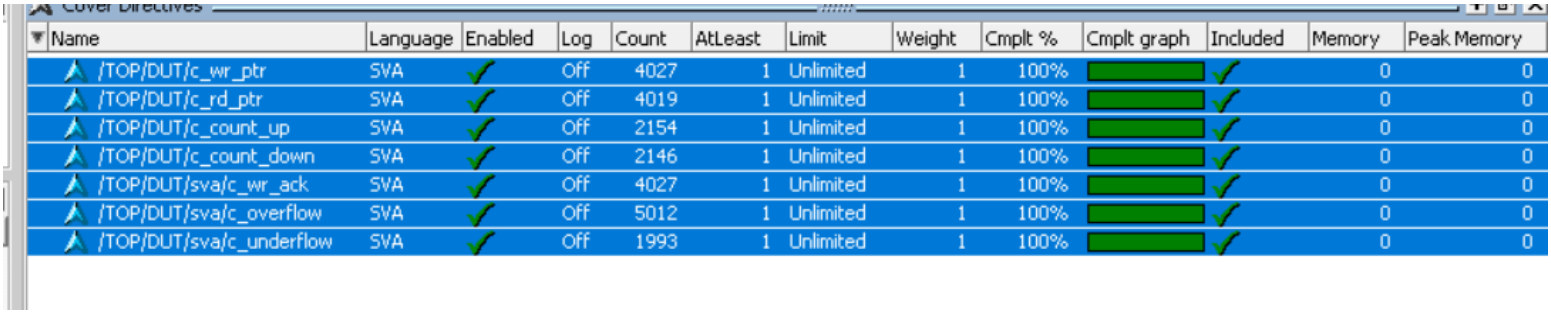
RUN.DO

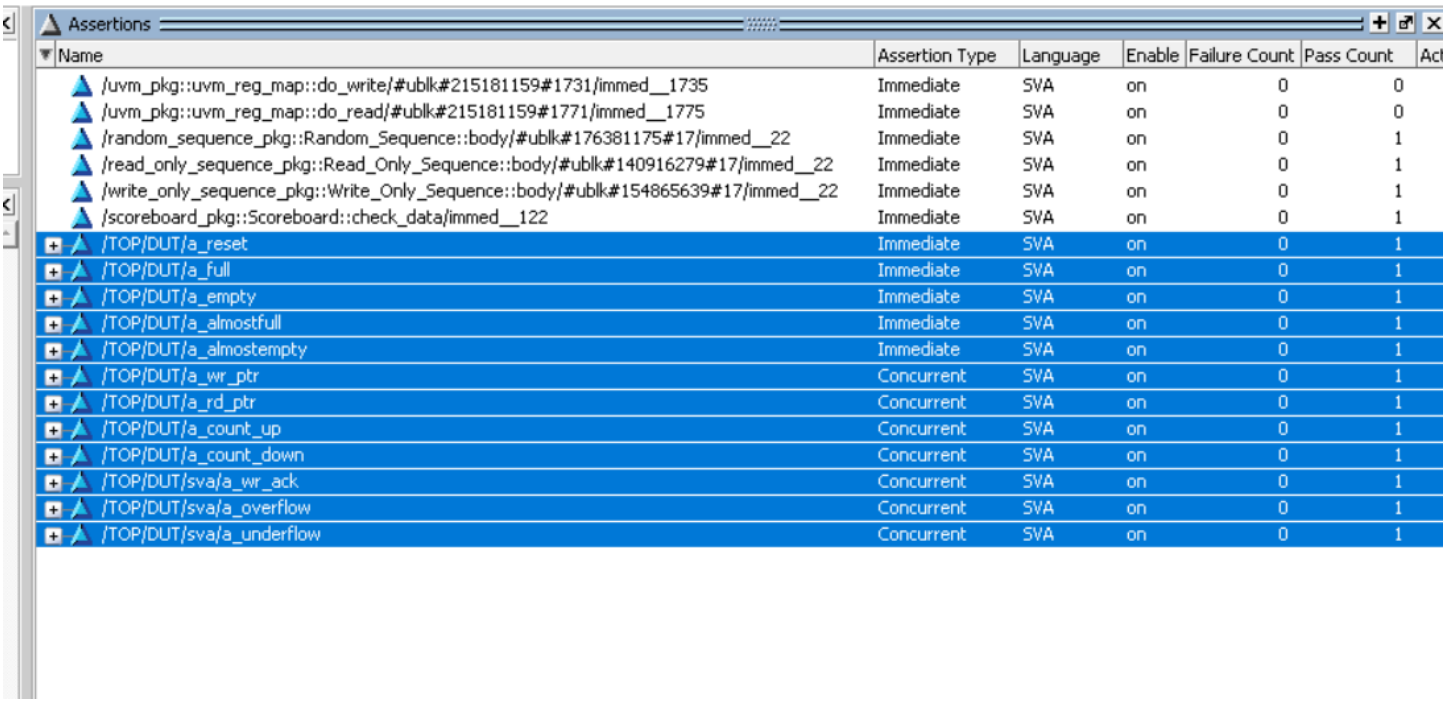
```
vlib work
vlog +fcover +cover +define+sva +acc -f source.list -covercells
vsim -voptargs=+acc -cover work.TOP
coverage save report.ucdb -onexit -du FIFO
add wave -noupdate /TOP/IF/clk
add wave -noupdate /TOP/IF/data_in
add wave -noupdate /TOP/IF/rst_n
add wave -noupdate /TOP/IF/wr_en
add wave -noupdate /TOP/IF/rd_en
add wave -noupdate /TOP/IF/data_out
add wave -noupdate /TOP/IF/wr_ack
add wave -noupdate /TOP/IF/overflow
add wave -noupdate /TOP/IF/full
add wave -noupdate /TOP/IF/empty
add wave -noupdate /TOP/IF/almostfull
add wave -noupdate /TOP/IF/almostempty
add wave -noupdate /TOP/IF/underflow
add wave -noupdate /TOP/DUT/count
add wave -noupdate -expand -group Coverage /TOP/DUT/c_wr_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_rd_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_up
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_down
add wave -noupdate -expand -group Coverage /TOP/DUT/sva/c_wr_ack
add wave -noupdate -expand -group Coverage /TOP/DUT/sva/c_overflow
add wave -noupdate -expand -group Coverage /TOP/DUT/sva/c_underflow
add wave -noupdate -expand -group Assertions /TOP/DUT/a_reset
add wave -noupdate -expand -group Assertions /TOP/DUT/a_full
add wave -noupdate -expand -group Assertions /TOP/DUT/a_empty
add wave -noupdate -expand -group Assertions /TOP/DUT/a_almostfull
add wave -noupdate -expand -group Assertions /TOP/DUT/a_almostempty
add wave -noupdate -expand -group Assertions /TOP/DUT/a_wr_ptr
add wave -noupdate -expand -group Assertions /TOP/DUT/a_rd_ptr
add wave -noupdate -expand -group Assertions /TOP/DUT/a_count_up
add wave -noupdate -expand -group Assertions /TOP/DUT/a_count_down
add wave -noupdate -expand -group Assertions /TOP/DUT/sva/a_wr_ack
add wave -noupdate -expand -group Assertions /TOP/DUT/sva/a_overflow
add wave -noupdate -expand -group Assertions /TOP/DUT/sva/a_underflow
run -all
add wave -position insertpoint  \
sim:/TOP/DUT/count
coverage exclude -du FIFO -togglenode rst_n
#vcover report report.ucdb -details -annotate -all -output coverage_report.txt
```

Code Coverage and Functional Coverage



Assertion Coverage





Reports

Code Coverage

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        29        29         0   100.00%

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

Branch Coverage for instance /TOP/DUT
```

```
Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Conditions                      20         20         0   100.00%

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

Condition Coverage for instance /TOP/DUT --

  File FIFO.sv
```

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

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

Toggle Coverage for instance /TOP/DUT --

                                            Node      1H->0L      0L->1H  "Coverag
e"
                                            ------------------------------------
-

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

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

Functional Coverage

```
Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        16        na        na        na
            Covergroup Bins        62        62         0   100.00%
--------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------
 TYPE /coverage_pkg/Coverage/cg                   100.00%        100          -    Covered
    covered/total bins:                                62         62          -
    missing/total bins:                                 0         62          -
    % Hit:                                         100.00%        100          -
    Coverpoint wr_en_cp                            100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                  5961          1          -    Covered
        bin auto[1]                                  9040          1          -    Covered
    Coverpoint rd_en_cp                            100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                  8988          1          -    Covered
        bin auto[1]                                  6013          1          -    Covered
    Coverpoint wr_ack_cp                           100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                 10974          1          -    Covered
        bin auto[1]                                  4027          1          -    Covered
    Coverpoint overflow_cp                         100.00%        100          -    Covered
        covered/total bins:                             2          2          -

```

Assertion Coverage

```
Directive Coverage:
    Directives                      3         3         0   100.00%

DIRECTIVE COVERAGE:
----------------------------------------------------------------------------------------
Name                                Design Design   Lang File(Line)      Hits Status
                                    Unit   UnitType
----------------------------------------------------------------------------------------
/TOP/DUT/sva/c_wr_ack               SVA    Verilog  SVA  SVA.sv(4)       4027 Covered
/TOP/DUT/sva/c_overflow             SVA    Verilog  SVA  SVA.sv(7)       5012 Covered
/TOP/DUT/sva/c_underflow            SVA    Verilog  SVA  SVA.sv(10)      1993 Covered
```

```
Directive Coverage:
    Directives                       4         4         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
--------------------------------------------------------------------------------
/TOP/DUT/c_wr_ptr                    FIFO   Verilog  SVA  FIFO.sv(105)    4027 Covered
/TOP/DUT/c_rd_ptr                    FIFO   Verilog  SVA  FIFO.sv(108)    4019 Covered
/TOP/DUT/c_count_up                  FIFO   Verilog  SVA  FIFO.sv(111)    2154 Covered
/TOP/DUT/c_count_down                FIFO   Verilog  SVA  FIFO.sv(114)    2146 Covered

```

```
================================================================================
=== Instance: /TOP/DUT
=== Design Unit: work.FIFO
================================================================================

Assertion Coverage:
    Assertions                       9         9         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/TOP/DUT/a_reset     FIFO.sv(95)                        0         1
/TOP/DUT/a_full      FIFO.sv(97)                        0         1
/TOP/DUT/a_empty     FIFO.sv(98)                        0         1
/TOP/DUT/a_almostfull
                     FIFO.sv(99)                        0         1
/TOP/DUT/a_almostempty
                     FIFO.sv(100)                       0         1
/TOP/DUT/a_wr_ptr    FIFO.sv(104)                       0         1
/TOP/DUT/a_rd_ptr    FIFO.sv(107)                       0         1
/TOP/DUT/a_count_up  FIFO.sv(110)                       0         1
/TOP/DUT/a_count_down
                     FIFO.sv(113)                       0         1
```

```
================================================================================
=== Instance: /TOP/DUT/sva
=== Design Unit: work.SVA
================================================================================

Assertion Coverage:
    Assertions                       3         3         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/TOP/DUT/sva/a_wr_ack
                     SVA.sv(3)                        0          1
/TOP/DUT/sva/a_overflow
                     SVA.sv(6)                        0          1
/TOP/DUT/sva/a_underflow
                     SVA.sv(9)                        0          1

Directive Coverage:
    Directives                       3         3         0   100.00%
```

Results:

```
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 150010: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# No. Of Passed Cases:       15001, No. Of Failed Cases:           0
# No. Of Passed Cases:       15001, No. Of Failed Cases:           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    4
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 150010 ns  Iteration: 61  Instance: /TOP
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```