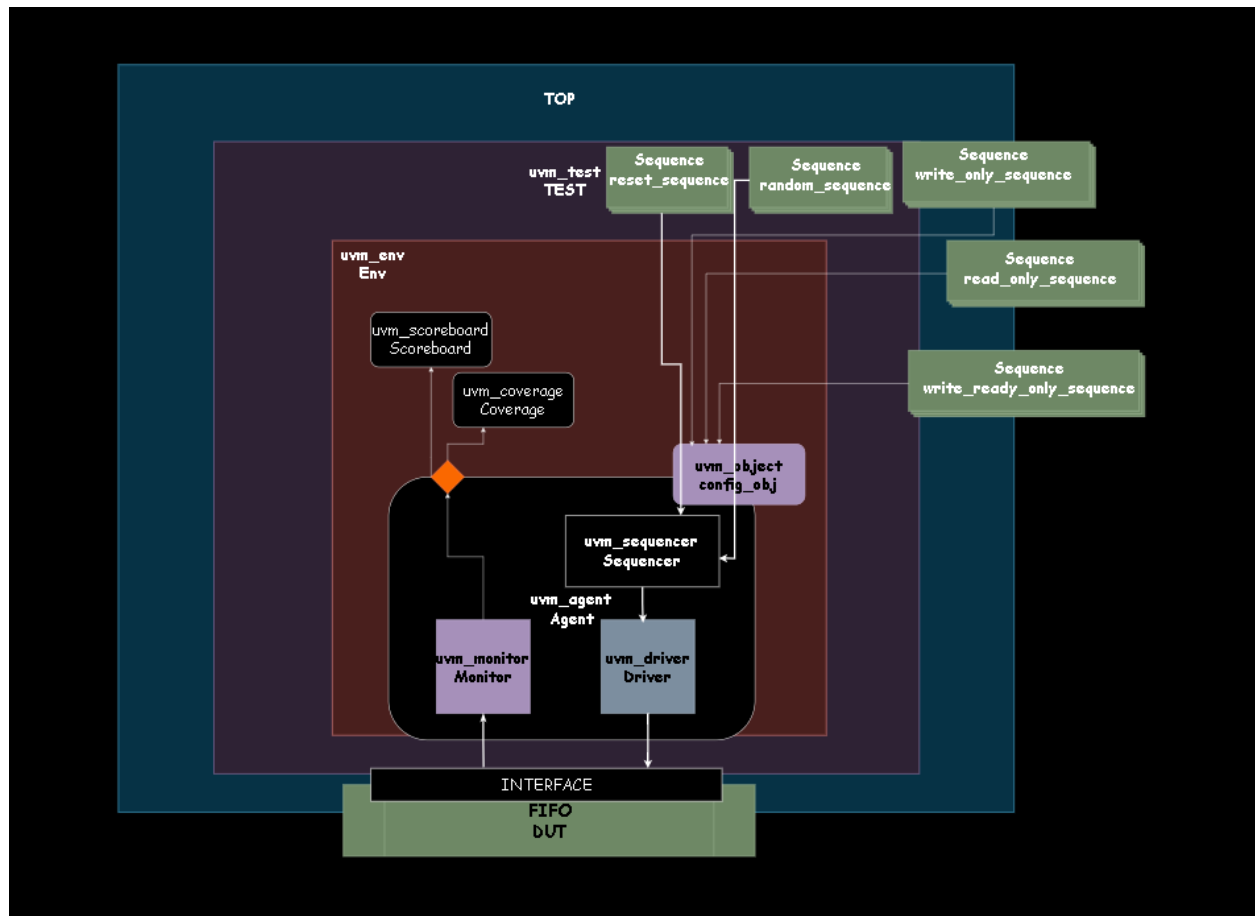


FINAL PROJECT - FIFO UVM



Top Module:

```
1 // include test_pkg.svh
2
3
4
5 `include "uvm_macros.svh"
6 import test_pkg::*;
7 import uvm_pkg::*;
8
9
10 module TOP();
11 bit clk=0;
12 FIFO_IF IF(clk);
13 FIFO DUT(
14     .data_in(IF.data_in), .wr_en(IF.wr_en), .rd_en(IF.rd_en),
15     .clk(IF.clk), .rst_n(IF.rst_n), .full(IF.full),
16     .empty(IF.empty), .almostfull(IF.almostfull), .almostempty(IF.almostempty),
17     .wr_ack(IF.wr_ack), .overflow(IF.overflow), .underflow(IF.underflow),
18     .data_out(IF.data_out)
19 );
20
21
22 always begin
23     #5 clk <= ~clk;
24 end
25
26 initial begin
27
28     uvm_config_db#(virtual FIFO_IF)::set(uvm_root::get(),"*", "IF", IF);
29     //uvm_config_db#(virtual shift_reg_if)::set(uvm_root::get(),"uvm_test_top", "IF", srif);
30     run_test("Test");
31 end
32
33 endmodule
34
```

Test:

```
1
2  package test_pkg;
3  `include "uvm_macros.svh"
4
5  //`include "main_sequence.sv"
6  //`include "reset_sequence.sv"
7
8  import uvm_pkg::*;
9  import env_pkg::*;
10
11  import reset_sequence_pkg::*;
12  import write_only_sequence_pkg::*;
13  import read_only_sequence_pkg::*;
14  import write_read_sequence_pkg::*;
15  import random_sequence_pkg::*;
16  import config_obj_pkg::*;
17
18
```

```
19
20
21  class Test extends uvm_test;
22  `uvm_component_utils(Test)
23      config_obj c_obj;
24      Env env;
25      Reset_Sequence reset_seq;
26      Write_Only_Sequence wos;
27      Read_Only_Sequence ros;
28      Write_Read_Sequence wrs;
29      Random_Sequence rs;
30
31
32
33      function new(string name = "Test", uvm_component parent = null);
34          | super.new(name,parent);
35      endfunction //new()
36
37
```

```

39
40     function void build_phase(uvm_phase phase);
41         super.build_phase(phase);
42
43         c_obj = config_obj::type_id::create("config_obj");
44
45         if(!uvm_config_db#(virtual FIFO_IF)::get(this,"", "IF",c_obj.vif))
46             begin
47                 `uvm_fatal("build_phase","Test - Unable to get the virtual interface")
48             end
49         uvm_config_db#(config_obj)::set(this,"*", "agent_config_obj",c_obj);
50         env = Env::type_id::create("Env",this);
51
52         reset_seq = Reset_Sequence::type_id::create("reset_seq",this);
53         wos = Write_Only_Sequence::type_id::create("wos",this);
54         ros = Read_Only_Sequence::type_id::create("ros",this);
55         wrs = Write_Read_Sequence::type_id::create("wrs",this);
56         rs = Random_Sequence::type_id::create("rs",this);
57
58
59     endfunction
60
61     task run_phase(uvm_phase phase);
62         super.run_phase(phase);
63         phase.raise_objection(this);
64     /*
65     wos.start(env.agent.sqr);
66     ros.start(env.agent.sqr);
67     wrs.start(env.agent.sqr);
68     rs.start(env.agent.sqr);
69
70     */

```

```

69
70     */
71
72     // #100; `uvm_info("run_phase","start reset sequence!",UVM_NONE)
73     reset_seq.start(env.agent.sqr);
74     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
75     wos.start(env.agent.sqr);
76     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
77     ros.start(env.agent.sqr);
78     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
79     wrs.start(env.agent.sqr);
80     // #100; `uvm_info("run_phase","start main sequence!",UVM_NONE)
81     rs.start(env.agent.sqr);
82     phase.drop_objection(this);
83     endtask
84 endclass //className extends superClass
85
86
87
88 endpackage
89

```


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:

```
1  package random_sequence_pkg;
2  `include "uvm_macros.svh"
3  import uvm_pkg::*;
4  //`include "Sequence_Item.sv"
5  import sequence_item_pkg::*;
6  class Random_Sequence extends uvm_sequence #(Sequence_Item);
7  `uvm_object_utils(Random_Sequence)
8  //Sequence_Item seq_item;
9  function new(string name = "Random_Sequence");
10  super.new(name);
11  endfunction
12
13  task body;
14  //seq_item = Sequence_Item::type_id::create("seq_item");
15
16  repeat(10000)
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  finish_item(seq_item);
24  end
25
26  endtask
27
28  endclass
29  endpackage
```

Read Only Sequence

Final Project 7 - Read Only Sequence Package

```
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

```
1  package write_read_sequence_pkg;
2  `include "uvm_macros.svh"
3  import uvm_pkg::*;
4  `include "Sequence_Item.sv"
5  import sequence_item_pkg::*;
6  class Write_Read_Sequence extends uvm_sequence #(Sequence_Item);
7  `uvm_object_utils(Write_Read_Sequence)
8  //Sequence_Item seq_item;
9  function new(string name = "Write_Read_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 seq_item.data_in =0;
23 seq_item.wr_en=1;
24 seq_item.rd_en=1;
25
26 finish_item(seq_item);
27 end
28
29 repeat(1000)
30 begin
31 Sequence_Item seq_item;
32 seq_item = Sequence_Item::type_id::create("seq_item");
33 start_item(seq_item);
34 seq_item.rst_n = 1;
35 //assert (seq_item.randomize());
36 seq_item.data_in =0;
37 seq_item.wr_en=0;
38 seq_item.rd_en=0;
39
40 finish_item(seq_item);
41 end
42 endtask
43 endclass
44 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

```
1
2 package scoreboard_pkg;
3 `include "uvm_macros.svh"
4 import uvm_pkg::*;
5 import sequence_item_pkg::*;
6 class Scoreboard extends uvm_scoreboard;
7 `uvm_component_utils(Scoreboard)
8
9 uvm_analysis_export#(Sequence_Item) sb_export;
10 uvm_tlm_analysis_fifo#(Sequence_Item) sb_fifo;
11 Sequence_Item fifo_transaction;
12
13 localparam FIFO_WIDTH = 16;
14 localparam FIFO_DEPTH = 8;
15 localparam max_fifo_addr = $clog2(FIFO_DEPTH);
16
17
18 logic [FIFO_WIDTH-1:0] data_out_ref;
19 logic wr_ack_ref, overflow_ref;
20 logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;
21
22 logic [max_fifo_addr-1:0] wr_ptr_ref;
23 logic [max_fifo_addr-1:0] rd_ptr_ref;
24 logic [max_fifo_addr:0] count_ref;
25
26
27 logic [FIFO_WIDTH-1 : 0] mem_ref [FIFO_DEPTH];
28
29 int failed = 0;
30 int passed = 0;
31
32
33 function new(string name = "Scoreboard", uvm_component parent = null);
34 super.new(name,parent);
35 endfunction
36
37 function void build_phase(uvm_phase phase);
38 super.build_phase(phase);
39 sb_export = new("sb_export",this);
40 sb_fifo = new("sb_fifo", this);
41 endfunction
42
43 function void connect_phase(uvm_phase phase);
44 super.connect_phase(phase);
```

```

4  super.connect_phase(phase);
5  sb_export.connect(sb_fifo.analysis_export);
6
7
8  endfunction
9
10 function void reference_model(Sequence_Item trans);
11     if (!trans.rst_n) begin
12         wr_ptr_ref = 0;
13         wr_ack_ref=0;
14         overflow_ref=0;//fault was detected here
15         data_out_ref= 0;//fault was detected here
16     end
17     else if (trans.wr_en && count_ref < 8) begin
18         mem_ref[wr_ptr_ref] = trans.data_in;
19         wr_ack_ref = 1;
20         wr_ptr_ref = wr_ptr_ref + 1;
21         overflow_ref = 0;//fault was detected here
22     end
23     else begin
24         wr_ack_ref = 0;
25         if (full_ref & trans.wr_en)
26             overflow_ref = 1;
27         else
28             overflow_ref = 0;
29     end
30
31     if (!trans.rst_n) begin
32         rd_ptr_ref = 0;
33         data_out_ref= 0;//fault was detected here
34         underflow_ref=0;//fault was detected here
35     end
36     else if (trans.rd_en && count_ref != 0) begin
37         data_out_ref = mem_ref[rd_ptr_ref];
38         rd_ptr_ref = rd_ptr_ref + 1;
39         underflow_ref=0;//fault was detected here
40     end
41     else begin
42         if(empty_ref&&trans.rd_en)
43             underflow_ref=1;
44         else

```

```

85     else
86         underflow_ref=0;
87     end
88
89
90     if (!trans.rst_n) begin
91         count_ref = 0;
92         data_out_ref= 0; //fault was detected here
93     end
94     else begin
95         if ( ({trans.wr_en, trans.rd_en} == 2'b10) && !full_ref) // if wr_en=1 and count_ref != 8
96             count_ref = count_ref + 1;
97         else if ( ({trans.wr_en, trans.rd_en} == 2'b01) && !empty_ref) // if rd_en=1 and count_ref != 0
98             count_ref = count_ref - 1;
99         else if ({trans.wr_en,trans.rd_en}==2'b11) begin //fault was detected here
100             if(full_ref)
101                 count_ref = count_ref-1;
102             else if (empty_ref)
103                 count_ref = count_ref+1;
104             end
105         end
106
107         full_ref = (count_ref == FIFO_DEPTH)? 1 : 0;
108         empty_ref = (count_ref == 0)? 1 : 0;
109         almostfull_ref = (count_ref == FIFO_DEPTH-1)? 1 : 0; //fault was detected here
110         almostempty_ref = (count_ref == 1)? 1 : 0;
111
112         //assign underflow_ref = (empty_ref && rd_en)? 1 : 0; //fault was detected here
113
114     endfunction
115
116
117     function void check_data(Sequence_Item trans);
118
119     reference_model(trans);
120
121
122     assert(trans.data_out === data_out_ref && trans.wr_ack === wr_ack_ref && trans.overflow === overflow_ref && trans.full === full_ref && trans
123     && trans.almostempty === almostempty_ref && underflow_ref === trans.underflow)
124     begin
125         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


```

1  package coverage_pkg;
2  `include "uvm_macros.svh"
3  import sequence_item_pkg::*;
4  import uvm_pkg::*;
5  class Coverage extends uvm_component;
6  `uvm_component_utils(Coverage)
7  uvm_analysis_export #(Sequence_Item) cov_export;
8  uvm_tlm_analysis_fifo #(Sequence_Item) cov_fifo;
9  Sequence_Item F_cvg_txn;
10
11  coveragegroup cg;
12  option.per_instance = 1;
13  wr_en_cp: coverpoint F_cvg_txn.wr_en;
14  rd_en_cp: coverpoint F_cvg_txn.rd_en;
15
16  wr_ack_cp: coverpoint F_cvg_txn.wr_ack;
17
18  overflow_cp: coverpoint F_cvg_txn.overflow;
19  underflow_cp: coverpoint F_cvg_txn.underflow;
20
21  full_cp: coverpoint F_cvg_txn.full;
22  empty_cp: coverpoint F_cvg_txn.empty;
23
24  almostfull_cp: coverpoint F_cvg_txn.almostfull;
25  almostempty_cp: coverpoint F_cvg_txn.almostempty;
26
27  wr__rd__ack: cross wr_en_cp,rd_en_cp,wr_ack_cp
28  {
29      ignore_bins ign_ack_without_wr = binsof(wr_en_cp) intersect {0} && binsof(wr_ack_cp) intersect{1};
30  }
31
32  wr__rd__overflow:cross wr_en_cp,rd_en_cp,overflow_cp
33  {
34      ignore_bins ign_overflow_without_wr = binsof(wr_en_cp) intersect {0} && binsof(overflow_cp) intersect{1};
35  }
36
37  wr__rd__underflow:cross wr_en_cp,rd_en_cp,underflow_cp
38  {
39      ignore_bins ign_underflow_without_rd = binsof(rd_en_cp) intersect {0} && binsof(underflow_cp) intersect{1};
40  }
41
42  wr__rd__full:cross wr_en_cp,rd_en_cp,full_cp
43  {
44      ignore_bins ign_full_without_wr = binsof(wr_en_cp) intersect {0} && binsof(full_cp) intersect{1};
45      ignore_bins ign_full_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(full_cp) intersect{1};
46  }

```

```

46 }
47 wr_rd_empty:cross wr_en_cp,rd_en_cp,empty_cp
48 {
49     ignore_bins ign_empty_without_rd = binsof(rd_en_cp) intersect {0} && binsof(empty_cp) intersect{1};
50     ignore_bins ign_empty_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(empty_cp) intersect{1};
51 }
52
53
54 wr_rd_almostfull:cross wr_en_cp,rd_en_cp,almostfull_cp;
55 wr_rd_almostempty:cross wr_en_cp,rd_en_cp,almostempty_cp;
56
57
58 endgroup
59
60 //CG cg;
61
62 function new (string name = "coverage",uvm_component parent = null);
63 super.new(name,parent);
64 cg = new();
65 endfunction
66
67 function void build_phase(uvm_phase phase);
68 super.build_phase(phase);
69     cov_export = new("cov_export", this);
70     cov_fifo = new ("cov_fifo",this);
71 endfunction
72
73 function void connect_phase(uvm_phase phase);
74
75     super.connect_phase(phase);
76     cov_export.connect(cov_fifo.analysis_export);
77
78 endfunction
79
80
81 task run_phase(uvm_phase phase);
82 super.run_phase(phase);
83
84 forever
85 begin

```

```

84     forever
85     begin
86         cov_fifo.get(F_cvg_txn);
87         cg.sample();
88
89     end
90
91 endtask
92 endclass
93 endpackage

```

Agent

```
1 package agent_pkg;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4
5 import sequence_item_pkg::*;
6 import driver_pkg::*;
7 import monitor_pkg::*;
8 import config_obj_pkg::*;
9 import sequencer_pkg::*;
10
11
12 class Agent extends uvm_agent;
13 `uvm_component_utils(Agent)
14 config_obj c_obj;
15 Sequencer sqr;
16 Driver driver;
17 Monitor monitor;
18 uvm_analysis_port #(Sequence_Item) agt_ap;
19 function new(string name = "Agent", uvm_component parent = null);
20 super.new(name,parent);
21 endfunction
22
23
24 function void build_phase(uvm_phase phase);
25 super.build_phase(phase);
26 sqr = Sequencer::type_id::create("sqr",this);
27 if(!uvm_config_db#(config_obj)::get(this,"","agent_config_obj",c_obj))//got from
28 begin
29 `uvm_fatal("build_phase","Driver - Unable to get the config object")
30 end
31
32 driver = Driver::type_id::create("driver",this);
33 monitor = Monitor::type_id::create("monitor",this);
34 agt_ap = new("agt_ap",this);
35 endfunction
36
37 task run_phase(uvm_phase phase);
38 super.run_phase(phase);
39 endtask
40
41 function void connect_phase(uvm_phase phase);
42 super.connect_phase(phase);
43 driver.seq_item_port.connect(sqr.seq_item_export);
44 monitor.IF = c_obj.vif;
```

```
48     endfunction
```

```
49
```

```
50
```

```
51
```

```
52
```

```
53     endclass
```

```
54
```

```
55
```

```
56     endpackage
```

```
57
```

```
58
```

Monitor

```
FinalProject > = 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.wr_en = IF.wr_en;
```

```

45  ▾ @(negedge IF.clk) ;
46      fifo_transaction.wr_ack = IF.wr_ack;
47
48      fifo_transaction.overflow = IF.overflow;
49      fifo_transaction.underflow = IF.underflow;
50
51      fifo_transaction.data_out = IF.data_out;
52      fifo_transaction.full = IF.full;
53      fifo_transaction.empty = IF.empty;
54
55      fifo_transaction.almostfull = IF.almostfull;
56      fifo_transaction.almostempty = IF.almostempty;
57
58      fifo_transaction.underflow = IF.underflow;
59      mon_ap.write(fifo_transaction);
60
61
62      // fork
63
64      //      fifo__coverage.sample_data(fifo_transaction);
65      //      begin
66
67      //      fifo_scoreboard.check_data(fifo_transaction);
68      //      end
69  ▾ // join
70
71
72      | //`uvm_info("run_phase","monitor running!",UVM_NONE)
73
74  end
75  endtask
76
77  endclass
78
79
80  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 ;
  end
endtask
endclass
```

```
6  //@(negedge vif.clk)
7  seq_item_port.item_done();
8
9  //`uvm_info("run_phase","driver running!",UVM_NONE)
0  //`$display("%s",seq_item.convert2string());
1
2  end
3  endtask
4
5  endclass
6  endpackage
7
8
```


SV Assertion

```
Final Project > SV_A.sv
1  module SVA(FIFO_IF,DUT IF);
2
3  a_wr_ack:assert property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.wr_en && (!IF.full)) | => IF.wr_ack );
4  c_wr_ack:cover property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.wr_en && (!IF.full)) | => IF.wr_ack );
5
6  a_overflow:assert property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.wr_en && IF.full) | => IF.overflow );
7  c_overflow:cover property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.wr_en && IF.full) | => IF.overflow );
8
9  a_underflow:assert property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.rd_en && IF.empty) | => IF.underflow );
10 c_underflow:cover property (@(posedge IF.clk) disable iff(!IF.rst_n) (IF.rd_en && IF.empty) | => IF.underflow );
11 endmodule
```

Config_Obj

```
Top.sv write_read_sequence.sv SWA.sv config_obj_pkg.sv X
Final Project > config_obj_pkg.sv
1 package config_obj_pkg;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 class config_obj extends uvm_object;
5 `uvm_object_utils(config_obj)
6 virtual FIFO_IF vif;
7 function new(string name = "config_obj");
8 | super.new(name);
9 endfunction
10 endclass
11 endpackage
```

FIFO_IF

```
Top.sv write_read_sequence.sv SWA.sv FIFO_IF.sv X FIFO.sv
Final Project > FIFO_IF.sv
1 interface FIFO_IF(input bit clk);
2 parameter FIFO_WIDTH = 16;
3 parameter FIFO_DEPTH = 8;
4
5 logic [FIFO_WIDTH-1:0] data_in;
6 logic rst_n;
7 logic wr_en, rd_en;
8
9
10 logic [FIFO_WIDTH-1:0] data_out;
11 logic wr_ack, overflow;
12 logic full, empty, almostfull, almostempty, underflow;
13
14 // modport TB (input clk, output data_in, output rst_n, output wr_en, output rd_en);
15 //modport Monitor (input clk, input rst_n, input data_in, input wr_en, input rd_en, input data_out, input wr_ack, input overflow, input full, input empty, input almostfull, input almostempty, input underflow);
16 modport DUT (input clk, input rst_n, input data_in, input wr_en, input rd_en, output data_out, wr_ack, overflow, full, empty, almostfull, almostempty, underflow);
17
18
19 endinterface //
20
21
```

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/clock
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

Instance	Design unit	Design unit type	Total coverage	Covergroup bin
/TOP/DUT/sva	SVA	Module	100.00%	
/TOP/DUT	FIFO	Module	100.00%	
/TOP	TOP	Module	100.00%	
/TOP/IF	FIFO_IF	Module	98.84%	

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_cover
/coverage_pkg/Co...		100.00%	100	100.00...				
TYPE cg		100.00%	100	100.00...				auto(1)
CVP cg:wr...		100.00%	100	100.00...				
CVP cg:rd...		100.00%	100	100.00...				
CVP cg:wr...		100.00%	100	100.00...				
CVP cg:ov...		100.00%	100	100.00...				
CVP cg:un...		100.00%	100	100.00...				
CVP cg:ful...		100.00%	100	100.00...				
CVP cg:...		100.00%	100	100.00...				
CVP cg:st...		100.00%	100	100.00...				
CVP cg:st...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
CROSS cg:...		100.00%	100	100.00...				
INST Vcov...		100.00%	100	100.00...				

Assertion Coverage

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory
/TOP/DUT/c_wr_ptr	SVA	✓	Off	4027	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/c_rd_ptr	SVA	✓	Off	4019	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/c_count_up	SVA	✓	Off	2154	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/c_count_down	SVA	✓	Off	2146	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/sva/c_wr_ack	SVA	✓	Off	4027	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/sva/c_overflow	SVA	✓	Off	5012	1	Unlimited	1	100%		✓	0	0
/TOP/DUT/sva/c_underflow	SVA	✓	Off	1993	1	Unlimited	1	100%		✓	0	0

Assertions						
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Act
▲ /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	
▲ /random_sequence_pkg::Random_Sequence::body/#ublk#176381175#17/immed__22	Immediate	SVA	on	0	1	
▲ /read_only_sequence_pkg::Read_Only_Sequence::body/#ublk#140916279#17/immed__22	Immediate	SVA	on	0	1	
▲ /write_only_sequence_pkg::Write_Only_Sequence::body/#ublk#154865639#17/immed__22	Immediate	SVA	on	0	1	
▲ /scoreboard_pkg::Scoreboard::check_data/immed__122	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_reset	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_full	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_empty	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_almostfull	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_almostempty	Immediate	SVA	on	0	1	
▲ /TOP/DUT/a_wr_ptr	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/a_rd_ptr	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/a_count_up	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/a_count_down	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/sva/a_wr_ack	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/sva/a_overflow	Concurrent	SVA	on	0	1	
▲ /TOP/DUT/sva/a_underflow	Concurrent	SVA	on	0	1	

Reports

Code Coverage

● ● ●					
1	Branch Coverage:				
2	Enabled Coverage	Bins	Hits	Misses	Coverage
3	-----	----	----	-----	-----
4	Branches	29	29	0	100.00%
5					
6	=====Branch Details=====				
7	=				
8	Branch Coverage for instance /TOP/DUT				



```
1 Condition Coverage:
2     Enabled Coverage           Bins   Covered   Misses   Coverage
3     -----
4     Conditions                 20      20       0    100.00%
5
6     =====Condition Details=====
7     ==
8 Condition Coverage for instance /TOP/DUT --
9
10    File FIFO.sv
```



```
1 Toggle Coverage:
2     Enabled Coverage          Bins    Hits    Misses  Coverage
3     -----
4     Toggles                   20     20      0    100.00%
5
6 =====Toggle Details=====
7
8 Toggle Coverage for instance /TOP/DUT --
9
10                                     Node    1H->0L    0L->1H  "Coverag
11     e"                                     -----
12     -
13 Total Node Count      =      10
14 Toggled Node Count   =      10
15 Untoggled Node Count =       0
16
17 Toggle Coverage      =    100.00% (20 of 20 bins)
```

Functional Coverage



1	Covergroup Coverage:					
2	Covergroups	1	na	na	100.00%	
3	Coverpoints/Crosses	16	na	na	na	
4	Covergroup Bins	62	62	0	100.00%	
5	-----					
6	Covergroup			Metric	Goal	Bins Status
7						
8	-----					
9	TYPE /coverage_pkg/Coverage/cg			100.00%	100	- Covered
10	covered/total bins:			62	62	-
11	missing/total bins:			0	62	-
12	% Hit:			100.00%	100	-
13	Coverpoint wr_en_cp			100.00%	100	- Covered
14	covered/total bins:			2	2	-
15	missing/total bins:			0	2	-
16	% Hit:			100.00%	100	-
17	bin auto[0]			5961	1	- Covered
18	bin auto[1]			9040	1	- Covered
19	Coverpoint rd_en_cp			100.00%	100	- Covered
20	covered/total bins:			2	2	-
21	missing/total bins:			0	2	-
22	% Hit:			100.00%	100	-
23	bin auto[0]			8988	1	- Covered
24	bin auto[1]			6013	1	- Covered
25	Coverpoint wr_ack_cp			100.00%	100	- Covered
26	covered/total bins:			2	2	-
27	missing/total bins:			0	2	-
28	% Hit:			100.00%	100	-
29	bin auto[0]			10974	1	- Covered
30	bin auto[1]			4027	1	- Covered
31	Coverpoint overflow_cp			100.00%	100	- Covered
32	covered/total bins:			2	2	-
33						

Assertion Coverage

```
1 Directive Coverage:
2     Directives           3         3         0    100.00%
3
4 DIRECTIVE COVERAGE:
5 -----
6 Name                      Design Design  Lang File(Line)    Hits Status
7                           Unit   UnitType
8 -----
9 /TOP/DUT/sva/c_wr_ack      SVA    Verilog  SVA  SVA.sv(4)         4027 Covered
10 /TOP/DUT/sva/c_overflow    SVA    Verilog  SVA  SVA.sv(7)         5012 Covered
11 /TOP/DUT/sva/c_underflow   SVA    Verilog  SVA  SVA.sv(10)        1993 Covered
12
```



```
1 Directive Coverage:
2     Directives           4         4         0  100.00%
3
4 DIRECTIVE COVERAGE:
5 -----
6 Name                      Design Design  Lang File(Line)      Hits Status
7                               Unit   UnitType
8 -----
9 /TOP/DUT/c_wr_ptr         FIFO   Verilog  SVA  FIFO.sv(105)      4027 Covered
10 /TOP/DUT/c_rd_ptr         FIFO   Verilog  SVA  FIFO.sv(108)      4019 Covered
11 /TOP/DUT/c_count_up       FIFO   Verilog  SVA  FIFO.sv(111)      2154 Covered
12 /TOP/DUT/c_count_down     FIFO   Verilog  SVA  FIFO.sv(114)      2146 Covered
13
```



```
1  =====
2  ==
3  === Instance: /TOP/DUT
4  === Design Unit: work.FIFO
5  =====
6  ==
7
8  Assertion Coverage:
9
10  Assertions          9          9          0  100.00%
11  -----
12  Name                File(Line)                Failure    Pass
13                                     Count        Count
14  -----
15  /TOP/DUT/a_reset     FIFO.sv(95)                0          1
16  /TOP/DUT/a_full      FIFO.sv(97)                0          1
17  /TOP/DUT/a_empty     FIFO.sv(98)                0          1
18  /TOP/DUT/a_almostfull
19                                     FIFO.sv(99)                0          1
20  /TOP/DUT/a_almostempty
21                                     FIFO.sv(100)               0          1
22  /TOP/DUT/a_wr_ptr     FIFO.sv(104)               0          1
23  /TOP/DUT/a_rd_ptr     FIFO.sv(107)               0          1
24  /TOP/DUT/a_count_up   FIFO.sv(110)               0          1
25  /TOP/DUT/a_count_down
26                                     FIFO.sv(113)               0          1
```



```
1  =====
2  ==
3  === Instance: /TOP/DUT/sva
4  === Design Unit: work.SVA
5  =====
6  ==
7
8  Assertion Coverage:
9
10     Assertions                3          3          0    100.00%
11  -----
12
13  Name                          File(Line)                      Failure    Pass
14                                     Count          Count
15  -----
16  /TOP/DUT/sva/a_wr_ack
17                                     SVA.sv(3)                0          1
18  /TOP/DUT/sva/a_overflow
19                                     SVA.sv(6)                0          1
20  /TOP/DUT/sva/a_underflow
21                                     SVA.sv(9)                0          1
22
23  Directive Coverage:
24
25     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
```

