

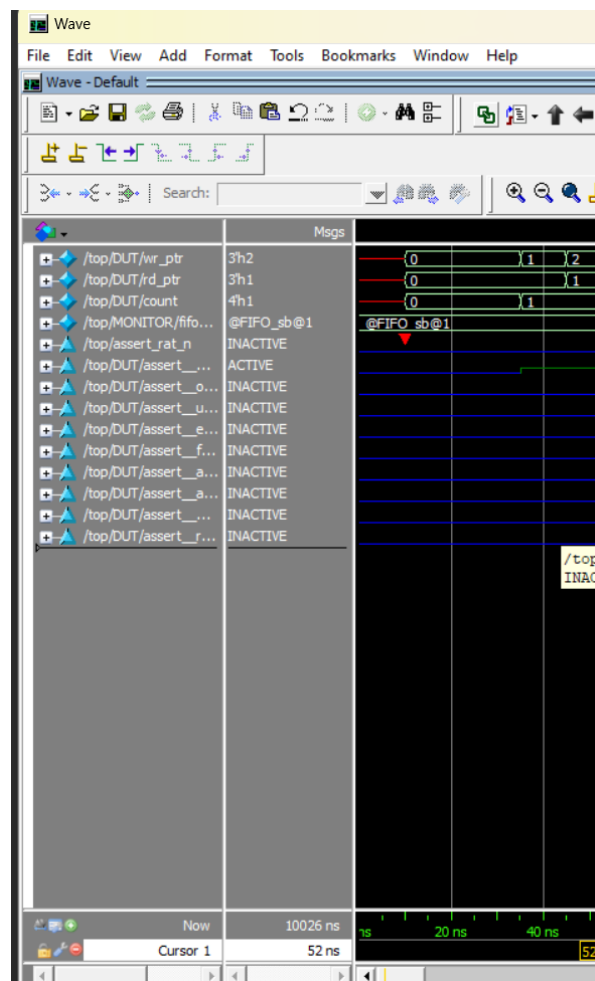
Project 1

⇒ Verification plan

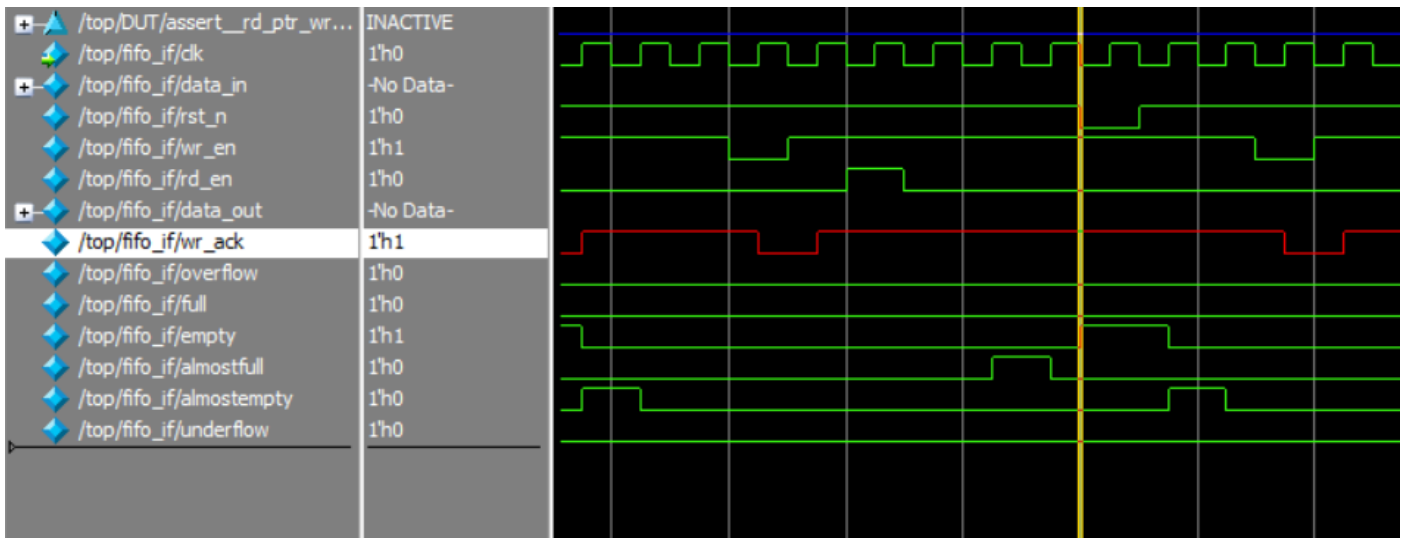
	Lable	Design requirement discription	Stimulus generation	Functional coverage	Functionality check
1					
2	FIFO_1	When reset is asserted all external pointers and output shall return to default	Directed at the start of the simulation	-	Immediate assertion to check the async reset functionality
3	FIFO_2	When wr_en is asserted data_in shall be written to the fifo if not full	Randomized during the simulation	Cover all values of wr_en with full flag	A checker compaires between the output and the reference model
4	FIFO_3	When rd_en is asserted the first data in the fifo shall be written to the data_out if not empty	Randomized during the simulation	Cover all values of rd_en with empty flag	A checker compaires between the output and the reference model
5	FIFO_4	When wr_en is asserted and the fifo is not full wr_ack shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with wr_ack flag	Concurrent assertion to check the functionality of wr_ack
6	FIFO_5	When wr_en is asserted and the fifo is full the overflow shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with over flow flag	Concurrent assertion to check the functionality of over flow
7	FIFO_6	When rd_en is asserted and the fifo is empty the underflow shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with underflow flag	Concurrent assertion to check the functionality of underflow
8	FIFO_7	When wr_en and rd_en are asserted at the same time if fifo is empty the underflow and wr_ack shall be high for one clock cycle	Randomized during the simulation	Cover all values of wr_en with rd_en and underflow and wr_ack	A checker compaires between the output and the reference model
9	FIFO_8	When the internal counter reaches FIFO_DEPTH the full flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and full flag	Concurrent assertion to check the functionality of full
10	FIFO_9	When the internal counter reaches FIFO_DEPTH-2 the almostfull flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and almostfull flag	Concurrent assertion to check the functionality of almostfull
11	FIFO_10	When the internal counter reaches ZERO the emptyl flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and empty	Concurrent assertion to check the functionality of empty
12	FIFO_11	When the internal counter reaches 1 the almostemptyl flag shall beasserte	Randomized during the simulation	Cover all values of wr_en with rd_en and almostempty flag	Concurrent assertion to check the functionality of almostempty
13					

⇒ Bugs detected

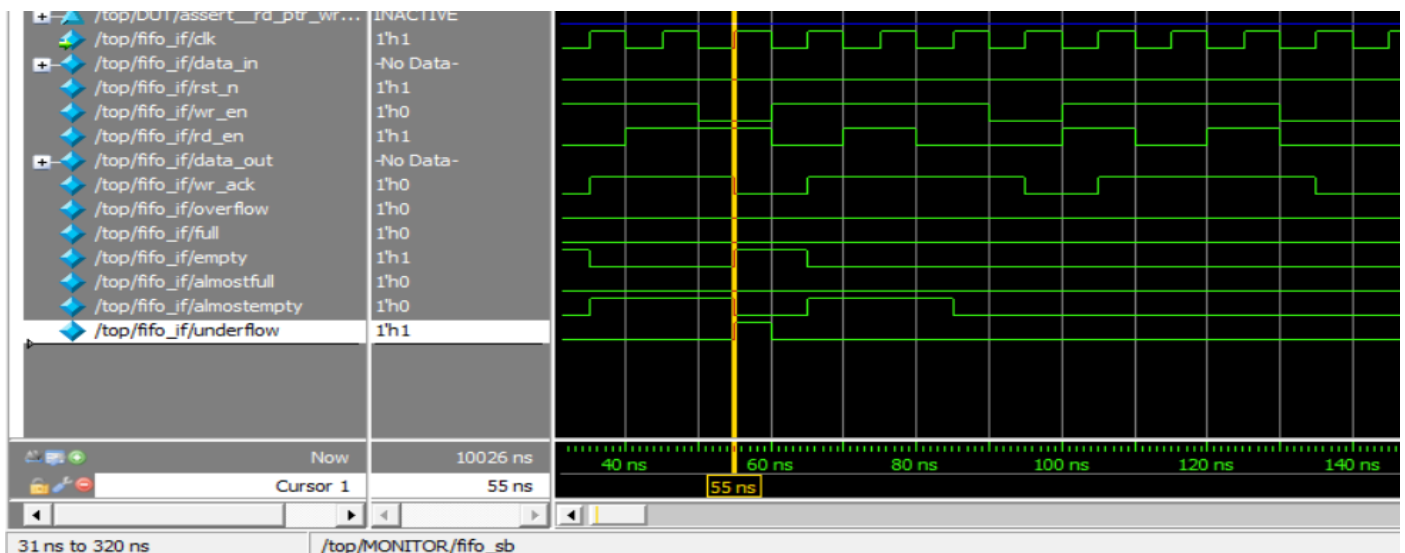
- 1- when resetting data_out register doesn't reset.



- 2- wr_ack is not affected by the reset.
- 2.1- and doesn't go low if it was high and the full flag is asserted before wr_en become low in the next rising edge of the clock.



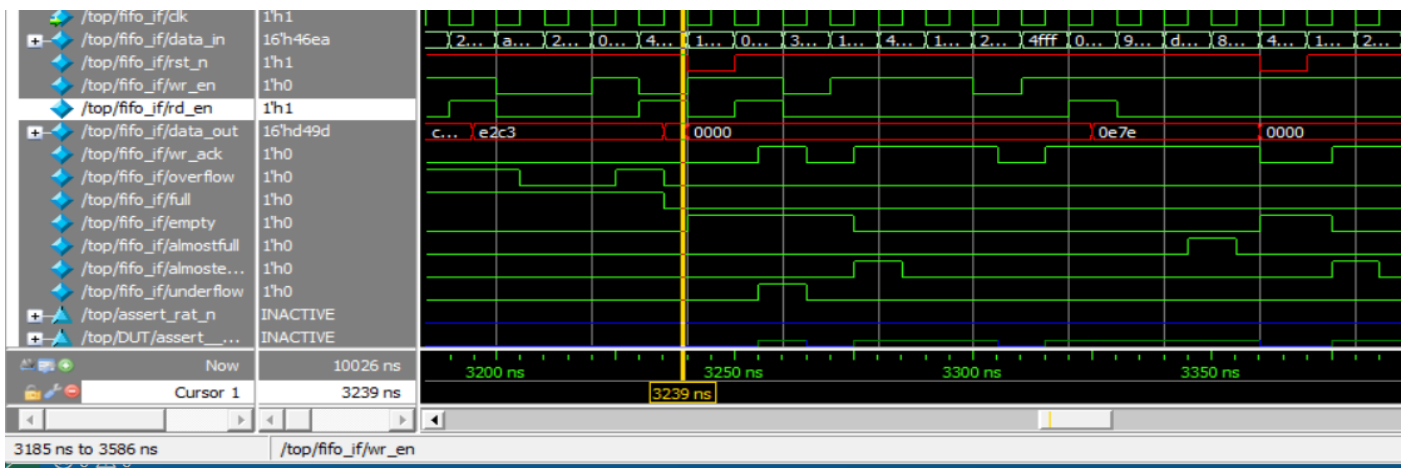
- 3- underflow is not sequential.



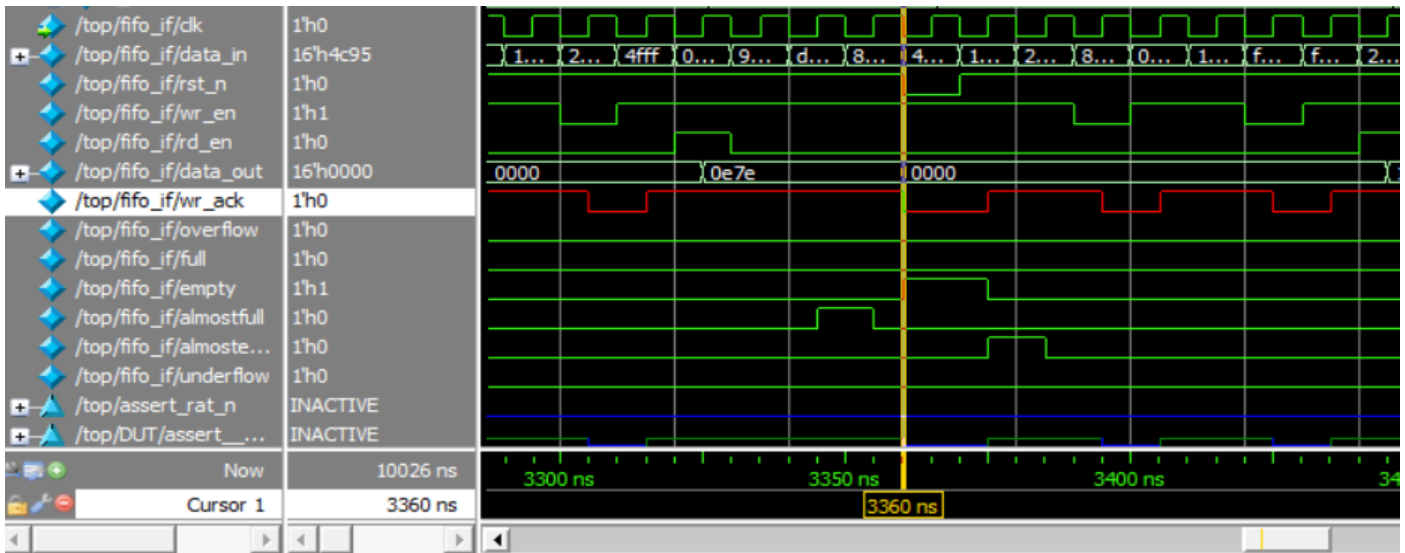
- 4- overflow shall be low if write operation is accepted.
- 5- if write or read operation is rejected the internal counter is not handled.
- 6- almost full when counter is FIFO_DEPTH-1 not -2

⇒ Fixed bugs

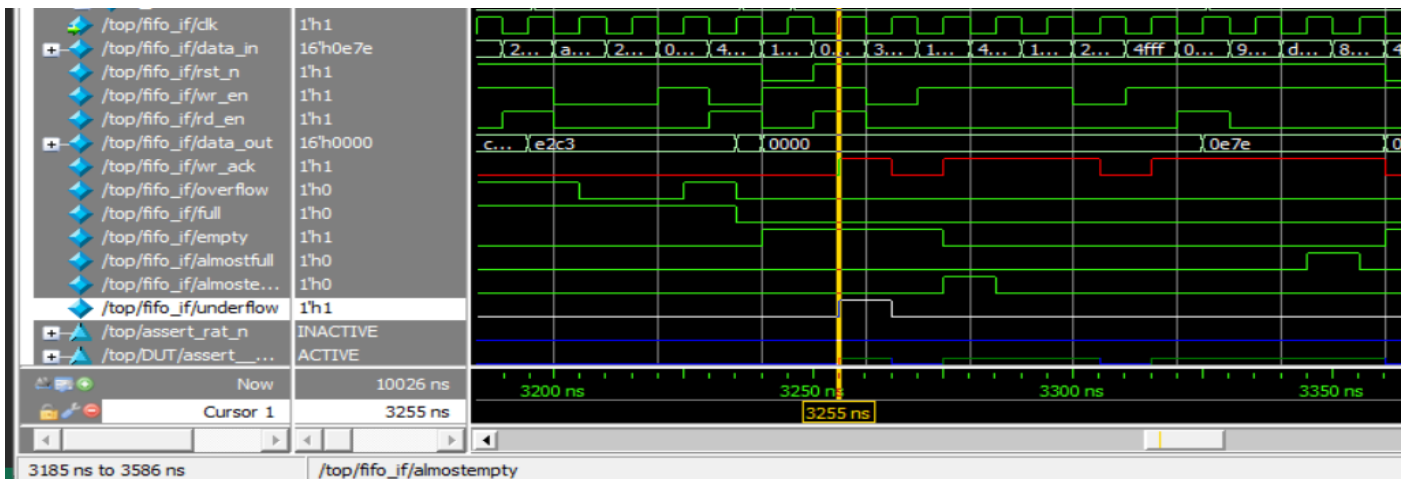
- Bug 1



○ Bug 2



○ Bug 3




○ Bug 4

```

18
19 always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
20     if (!fifo_if.rst_n) begin
21         wr_ptr <= 0;
22         fifo_if.wr_ack <= 0;
23         fifo_if.overflow <= 0;
24     end
25     else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
26         mem[wr_ptr] <= fifo_if.data_in;
27         fifo_if.wr_ack <= 1;
28         wr_ptr <= wr_ptr + 1;
29         fifo_if.overflow <= 0;
30     end
31     else if(fifo_if.full & fifo_if.wr_en) begin
32         fifo_if.wr_ack <= 0;
33         fifo_if.overflow <= 1;
34     end
35     else begin
36         fifo_if.overflow <= 0;
37     end
38 end
39


```

- Bug 5



```
60 always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
61     if (!fifo_if.rst_n) begin
62         count <= 0;
63     end
64     else begin
65         if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b10) && !fifo_if.full)
66             count <= count + 1;
67         else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b01) && !fifo_if.empty)
68             count <= count - 1;
69         else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.full)
70             count <= count - 1;
71         else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) && fifo_if.empty)
72             count <= count + 1;
73     end
74 end
```

- Bug 6



```
75
76 assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
77 assign fifo_if.empty = (count == 0)? 1 : 0;
78
79 assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
80 assign fifo_if.almostempty = (count == 1)? 1 : 0;
81
```

```
1  import shared_pkg::*;
2  import FIFO_transaction_pkg::*;
3
4  module fifo_tb (FIFO_if.TEST fifo_if);
5
6      FIFO_transaction fifo_txn = new;
7
8      initial begin
9          fifo_if.rst_n    = 0;
10         fifo_if.data_in = 0;
11         fifo_if.wr_en    = 0;
12         fifo_if.rd_en    = 0;
13
14         fifo_if.rst_n = 1;
15         @(negedge fifo_if.clk)
16         fifo_if.rst_n = 0;
17         @(negedge fifo_if.clk)
18         fifo_if.rst_n = 1;
19
20         repeat(10000)begin
21             assert(fifo_txn.randomize());
22             fifo_if.rst_n    = fifo_txn.rst_n    ;
23             fifo_if.data_in = fifo_txn.data_in ;
24             fifo_if.wr_en    = fifo_txn.wr_en    ;
25             fifo_if.rd_en    = fifo_txn.rd_en    ;
26             @(negedge fifo_if.clk);
27             -> trigger;
28             #1;
29         end
30         test_finished = 1;
31         -> trigger;
32     end
33
34 endmodule
```

⇒ Monitor

```
2 import shared_pkg::*;
3 import FIFO_coverage_pkg::*;
4 import FIFO_transaction_pkg::*;
5 import FIFO_sb_pkg::*;
6
7 module monitor ( FIFO_if.MONITOR fifo_if );
8
9     FIFO_coverage fifo_cvg = new;
10    FIFO_sb fifo_sb = new;
11    FIFO_transaction fifo_txn = new;
12
13    initial begin
14        forever begin
15
16            @(trigger);
17
18            fifo_txn.rst_n      = fifo_if.rst_n      ;
19            fifo_txn.wr_en      = fifo_if.wr_en      ;
20            fifo_txn.rd_en      = fifo_if.rd_en      ;
21            fifo_txn.data_in     = fifo_if.data_in    ;
22            fifo_txn.data_out    = fifo_if.data_out   ;
23            fifo_txn.wr_ack     = fifo_if.wr_ack     ;
24            fifo_txn.overflow    = fifo_if.overflow   ;
25            fifo_txn.full        = fifo_if.full       ;
26            fifo_txn.empty       = fifo_if.empty      ;
27            fifo_txn.almostfull  = fifo_if.almostfull;
28            fifo_txn.almostempty = fifo_if.almostempty;
29            fifo_txn.underflow   = fifo_if.underflow  ;
30
31            fork
32                fifo_cvg.sample_data(fifo_txn);
33                fifo_sb.check_data(fifo_txn);
34            join
35
36            if(test_finished)begin
37                $display("test done... \n Number of correct tests:
38 \nNumber of failed tests:
39 ", correct_count, error_count);
40            end
41        end
42    end
43
```

```

39
40  function void reference_model(FIFO_transaction fifo_txn);
41      if(~fifo_txn.rst_n) begin
42          data_out_ref  = 0 ;
43          wr_ack_ref    = 0 ;
44          overflow_ref   = 0 ;
45          full_ref      = 0 ;
46          empty_ref     = 1 ;
47          almostfull_ref = 0 ;
48          almostempty_ref = 0 ;
49          underflow_ref  = 0 ;
50          counter        = 0 ;
51          w_ptr          = 0 ;
52          r_ptr          = 0 ;
53      end
54      else begin
55          if(fifo_txn.rd_en && !empty_ref)begin
56              data_out_ref = mem_ref[r_ptr];
57              r_ptr++;
58              underflow_ref = 0;
59              if(!fifo_txn.wr_en)
60                  counter--;
61              else if(fifo_txn.wr_en && full_ref)
62                  counter--;
63          end
64          else if(fifo_txn.rd_en && empty_ref)
65              underflow_ref = 1;
66          else
67              underflow_ref = 0;
68
69          if(fifo_txn.wr_en && !full_ref)begin
70              mem_ref[w_ptr] = fifo_txn.data_in;
71              w_ptr++;
72              wr_ack_ref = 1;
73              overflow_ref = 0;
74              if(!fifo_txn.rd_en)
75                  counter++;
76              else if(fifo_txn.rd_en && empty_ref)
77                  counter++;
78          end
79          else if(fifo_txn.wr_en && full_ref)begin
80              wr_ack_ref = 0;
81              overflow_ref = 1;
82          end
83          else begin
84              overflow_ref = 0;
85              wr_ack_ref = 0;
86          end
87
88
89
90          full_ref = (counter == FIFO_DEPTH)? 1 : 0 ;
91          empty_ref = (counter == 0)? 1 : 0 ;
92          almostfull_ref = (counter == FIFO_DEPTH-2)? 1 : 0 ;
93          almostempty_ref = (counter == 1)? 1 : 0 ;
94      end
95  endfunction //automatic

```


⇒ Coverage group

```
19
20   covergroup cvr_gp;
21       wr_en_c : coverpoint cov_seq_item.wr_en;
22       rd_en_c : coverpoint cov_seq_item.rd_en;
23       wr_ack_c : coverpoint cov_seq_item.wr_ack;
24       ovf_c    : coverpoint cov_seq_item.overflow;
25       unf_c    : coverpoint cov_seq_item.underflow;
26       full_c   : coverpoint cov_seq_item.full;
27       empty_c  : coverpoint cov_seq_item.empty;
28       cross_1: cross wr_en_c, rd_en_c, wr_ack_c iff(cov_seq_item.rst_n){
29           illegal_bins wr_rd_ack =   binsof(wr_en_c ) intersect {0} &&
30                                   binsof(rd_en_c ) intersect {1} &&
31                                   binsof(wr_ack_c) intersect {1}  ;
32           illegal_bins nwr_nrd_ack = binsof(wr_en_c ) intersect {0} &&
33                                   binsof(rd_en_c ) intersect {0} &&
34                                   binsof(wr_ack_c) intersect {1};
35       }
36       cross_2: cross wr_en_c, rd_en_c, ovf_c iff(cov_seq_item.rst_n){
37           illegal_bins wr_rd_ovf =   binsof(wr_en_c) intersect {0} &&
38                                   binsof(rd_en_c) intersect {1} &&
39                                   binsof(ovf_c)intersect {1}  ;
40           illegal_bins nwr_nrd_ovf = binsof(wr_en_c) intersect {0} &&
41                                   binsof(rd_en_c) intersect {0} &&
42                                   binsof(ovf_c) intersect {1};
43       }
44       cross_3: cross wr_en_c, rd_en_c, unf_c iff(cov_seq_item.rst_n){
45           illegal_bins wr_rd_unf =   binsof(wr_en_c) intersect {1} &&
46                                   binsof(rd_en_c) intersect {0} &&
47                                   binsof(unf_c)intersect {1}  ;
48           illegal_bins nwr_nrd_unf = binsof(wr_en_c) intersect {0} &&
49                                   binsof(rd_en_c) intersect {0} &&
50                                   binsof(unf_c) intersect {1};
51       }
52       cross_4: cross wr_en_c, rd_en_c, full_c iff(cov_seq_item.rst_n){
53           illegal_bins nwr_nrd_full = binsof(wr_en_c) intersect {0} &&
54                                   binsof(rd_en_c) intersect {1} &&
55                                   binsof(full_c) intersect {1};
56
57           illegal_bins wr_nrd_full = binsof(wr_en_c) intersect {1} &&
58                                   binsof(rd_en_c) intersect {1} &&
59                                   binsof(full_c) intersect {1};
60       }
61       cross_5: cross wr_en_c, rd_en_c, empty_c      iff(cov_seq_item.rst_n){
62           illegal_bins nwr_nrd_empty = binsof(wr_en_c) intersect {1} &&
63                                   binsof(rd_en_c) intersect {0} &&
64                                   binsof(empty_c) intersect {1};
65           illegal_bins wr_nrd_empty = binsof(wr_en_c) intersect {1} &&
66                                   binsof(rd_en_c) intersect {1} &&
67                                   binsof(empty_c) intersect {1};
68       }
69       cross_6: cross wr_en_c, rd_en_c, cov_seq_item.almostfull iff(cov_seq_item.rst_n);
70       cross_7: cross wr_en_c, rd_en_c, cov_seq_item.almostempty iff(cov_seq_item.rst_n) ;
71   endgroup
72
```


⇒ Functional coverage

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_cover
/FIFO_coverage_pkg/FIFO_coverage		100.00%						
TYPE cvr_gp		100.00%	100	100.00...		✓	auto(1)	
CVP cvr_gp::wr_en_c		100.00%	100	100.00...		✓		
CVP cvr_gp::rd_en_c		100.00%	100	100.00...		✓		
CVP cvr_gp::wr_ack_c		100.00%	100	100.00...		✓		
CVP cvr_gp::ovf_c		100.00%	100	100.00...		✓		
CVP cvr_gp::unf_c		100.00%	100	100.00...		✓		
CVP cvr_gp::full_c		100.00%	100	100.00...		✓		
CVP cvr_gp::empty_c		100.00%	100	100.00...		✓		
CVP cvr_gp::F_cvg_bxn.almostem...		100.00%	100	100.00...		✓		
CVP cvr_gp::F_cvg_bxn.almostfull...		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_1		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_2		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_3		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_4		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_5		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_6		100.00%	100	100.00...		✓		
CROSS cvr_gp::cross_7		100.00%	100	100.00...		✓		

⇒ Assertions

/uvm_pkg::uvm_reg_map::do_write/#ublk#21518115...	Immediate	SVA	on	0	0	-
/uvm_pkg::uvm_reg_map::do_read/#ublk#21518115...	Immediate	SVA	on	0	0	-
/FIFO_seq_pkg::FIFO_write_only_seq::body/#ublk#...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_write_after_ovf_seq::body/#u...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_read_after_empty_seq::body/...	Immediate	SVA	on	0	1	-
/FIFO_seq_pkg::FIFO_write_read_seq::body/#ublk#...	Immediate	SVA	on	0	1	-
/top/DUT/fifo_sva/rst_n_assert	Immediate	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_wr_ack_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_over_flow_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_under_flow_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_empty_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_full_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_almostfull_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_almostempty_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_wr_ptr_wrap_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_rd_ptr_wrap_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_rd_notfull_assert	Concurrent	SVA	on	0	1	-
/top/DUT/fifo_sva/assert_wr_notempty_assert	Concurrent	SVA	on	0	1	-