

ASYNCHRONOUS FIFO VERIFICATION



Prepared by: Marwan Yasser Rifaat
Under Supervision: Eng Kareem Waseem

Asynchronous FIFO (Overview)

An **Asynchronous FIFO** is a type of First-In-First-Out memory buffer used to transfer data between two systems operating on **different clock domains**. Unlike a synchronous FIFO, where both read and write operations share the same clock, an asynchronous FIFO has **separate write and read clocks**, typically denoted as `wr_clk` and `rd_clk`.

This design is essential in digital systems where modules run at different frequencies or are not phase-aligned, such as communication interfaces or mixed-speed data paths. The asynchronous FIFO ensures **safe and reliable data transfer** by employing **clock domain crossing (CDC)** techniques, such as **Gray code pointers** and **synchronizer flip-flops**, to prevent metastability and timing violations.

Parameters

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

Ports

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

FIFO Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO1	Reset behavior: On reset, all pointers and count are zero.	Drive rst_n=0 for multiple cycles during simulation.	Cover final (wr_ptr==0 & rd_ptr==0 & count==0)	Assertion a_reset, scoreboard reference model resets memory and flags.
FIFO2	Write acknowledge: When wr_en=1 and FIFO not full, wr_ack must be asserted next cycle.	Randomized wr_en and full=0 conditions.	cover property(Write_Acknowledge)	Assertion Write_Acknowledge, scoreboard compares wr_ack vs ref model.
FIFO3	Overflow detection: If FIFO full and write occurs without read, overflow=1.	Generate wr_en=1, full=1, rd_en=0.	cover property(Overflow_Detection)	Assertion Overflow_Detection, scoreboard compares overflow_ref.
FIFO4	Underflow detection: If FIFO empty and read occurs without write, underflow=1.	Generate rd_en=1, empty=1, wr_en=0.	cover property(Underflow_Detection)	Assertion Underflow_Detection, scoreboard checks underflow_ref.
FIFO5	Empty flag correctness: When count==0, empty=1.	Simulate FIFO draining to zero elements.	cover property(Empty_Flag_Assertion)	Assertion Empty_Flag_Assertion, scoreboard empty_ref check.
FIFO6	Full flag correctness: When count==FIFO_DEPTH, full=1.	Fill FIFO completely via random writes.	cover property(Full_Flag_Assertion)	Assertion Full_Flag_Assertion, scoreboard full_ref check.
FIFO7	Almost full threshold: When count = DEPTH-1, almostfull=1.	Generate write operations until near full.	cover property(Almost_Full_Condition)	Assertion Almost_Full_Condition, scoreboard almostfull_ref check.
FIFO8	Almost empty threshold: When count = 1, almostempty=1.	Generate read operations until one element left.	cover property(Almost_Empty_Condition)	Assertion Almost_Empty_Condition, scoreboard almostempty_ref check.
FIFO9	Write pointer wraparound: When wr_ptr reaches end, it resets to 0.	Generate continuous writes exceeding FIFO depth.	cover property(Pointer_Wraparound_1)	Assertion Pointer_Wraparound_1, scoreboard implicit pointer size check.
FIFO10	Read pointer wraparound: When rd_ptr reaches end, it resets to 0.	Generate continuous reads exceeding FIFO depth.	cover property(Pointer_Wraparound_2)	Assertion Pointer_Wraparound_2.
FIFO11	Pointer threshold validity: wr_ptr, rd_ptr, count always ≤ FIFO_DEPTH.	Randomized write/read sequences.	cover property(Pointer_threshold)	Assertion Pointer_threshold ensures bounds.
FIFO12	Simultaneous wr/rd when empty: FIFO should not remain empty.	Generate wr_en=1, rd_en=1, empty=1.	cover property(wr_rd_simul_empty)	Assertion wr_rd_simul_empty, scoreboard verifies empty_ref change.
FIFO13	Simultaneous wr/rd when full: FIFO should not remain full.	Generate wr_en=1, rd_en=1, full=1.	cover property(wr_rd_simul_full)	Assertion wr_rd_simul_full, scoreboard verifies full_ref change.
FIFO14	FIFO data integrity: Data read = oldest written (FIFO order).	Randomized writes and reads.	Covered implicitly in transaction coverage crosses.	Scoreboard compares data_out vs expected FIFO model order.
FIFO15	Overflow prevention during concurrent wr/rd: No overflow if both active.	Randomized sequences with wr_en=rd_en=1.	Covered in cross wr_en_rd_en_OVERFLOW.	Scoreboard reference model checks overflow_ref.
FIFO16	Underflow prevention during concurrent wr/rd: No underflow if both active.	Randomized sequences with both enables high.	Cross wr_en_rd_en_UNDERFLOW.	Scoreboard underflow_ref check.
FIFO17	Functional cross coverage: Check all combinations of write/read with flags.	Randomized enables in 100k cycles.	Coverage crosses for WR_EN, RD_EN, and each flag.	Coverage bins ensure all operational modes tested.
FIFO18	Final test termination check: Report total correct/error counts.	End simulation when test_finished=1.	Coverage all goals reached.	\$display("error_count=%d, correct_count=%d") at end.

FIFO Design before fixing Bugs:

```
/ ///////////////////////////////
8 module FIFO(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
9 parameter FIFO_WIDTH = 16;
10 parameter FIFO_DEPTH = 8;
11 input [FIFO_WIDTH-1:0] data_in;
12 input clk, rst_n, wr_en, rd_en;
13 output reg [FIFO_WIDTH-1:0] data_out;
14 output reg wr_ack, overflow;
15 output full, empty, almostfull, almostempty, underflow;
16
17 localparam max_fifo_addr = $clog2(FIFO_DEPTH);
18
19 reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
20
21 reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
22 reg [max_fifo_addr:0] count;
23
24 always @(posedge clk or negedge rst_n) begin
25   if (!rst_n) begin
26     wr_ptr <= 0;
27   end
28   else if (wr_en && count < FIFO_DEPTH) begin
29     mem[wr_ptr] <= data_in;
30     wr_ack <= 1;
31     wr_ptr <= wr_ptr + 1;
32   end
33   else begin
34     wr_ack <= 0;
35     if (full & wr_en)
36       overflow <= 1;
37     else
38       overflow <= 0;
39   end
40 end
41
42 always @(posedge clk or negedge rst_n) begin
43   if (!rst_n) begin
44     rd_ptr <= 0;
45   end
46   else if (rd_en && count != 0) begin
47     data_out <= mem[rd_ptr];
48     rd_ptr <= rd_ptr + 1;
49   end
50 end
51
52 always @(posedge clk or negedge rst_n) begin
53   if (!rst_n) begin
54     count <= 0;
55   end
56   else begin
57     if ( ({wr_en, rd_en} == 2'b10) && !full)
58       count <= count + 1;
59     else if ( ({wr_en, rd_en} == 2'b01) && !empty)
60       count <= count - 1;
61   end
62 end
63
64 assign full = (count == FIFO_DEPTH)? 1 : 0;
65 assign empty = (count == 0)? 1 : 0;
66 assign underflow = (empty && rd_en)? 1 : 0;
67 assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
68 assign almostempty = (count == 1)? 1 : 0;
69
70 endmodule
```

FIFO Bug Report and Fix Summary

Bug ID	Category	Description (in Buggy Design)	Root Cause / Impact	Fix Implemented (in Fixed Design)	Verification / Check Used
Bug1	Reset Handling	wr_ack and overflow not reset to 0 during reset.	Left undefined after reset; may cause random outputs or false flags.	Added explicit reset assignments:fifoif.wr_ack <= 0;fifoif.overflow <= 0;	Assertion a_reset, scoreboard reset check.
Bug2	Overflow Condition	Overflow asserted when full & wr_en, but ignored concurrent read (rd_en).	During simultaneous wr_en and rd_en, overflow was incorrectly set.	Fixed condition:if (fifoif.full && fifoif.wr_en && !fifoif.rd_en)	Assertion Overflow_Detection, coverage cross WR_EN_RD_EN_OVERFLOW.
Bug3	Underflow Signal Type	underflow was combinational (assign underflow = ...)	Glitchy, unstable underflow output since it changes without clock edge.	Changed to sequential logic under clock in always block:fifoif.underflow <= 1/0;	Assertion Underflow_Detection, scoreboard underflow_ref check.
Bug4	Underflow Reset	underflow not reset when rst_n=0.	May retain '1' from previous cycles after reset.	Added reset logic:fifoif.underflow <= 0; in reset branch.	Assertion a_reset, scoreboard reset verification.
Bug5	Count Update (Concurrent wr/rd)	No handling for simultaneous write and read (wr_en && rd_en).	FIFO depth counter count incorrect; causes flag misbehavior near boundaries.	Added explicit handling: if(wr_en&&rd_en&&empty) count++; if(wr_en&&rd_en&&full) count--;	Assertions wr_rd_simul_empty and wr_rd_simul_full.
Bug6	Almost Full Threshold	almostfull asserted at count == FIFO_DEPTH-2.	Flag activated too early, misguiding flow control.	Fixed threshold to:count == FIFO_DEPTH-1.	Assertion Almost_Full_Condition, scoreboard check.
Bug7	Missing Sequential Underflow Logic	No sequential control for underflow when empty and rd_en.	underflow would not remain asserted for a full clock; unstable flag.	Added registered control:Underflow asserted/deasserted inside clocked block.	Assertion Underflow_Detection, coverage cross with RD_EN.
Bug8	Write Acknowledge Missing Reset	wr_ack uninitialized after reset.	May start high on first cycle.	Added reset initialization fifoif.wr_ack <= 0;	Assertion Write_Acknowledge, reset assertion.
Bug9	Missing Overflow Reset Behavior	overflow not reset during rst_n=0.	Overflow could stay high after reset.	Added fifoif.overflow <= 0; in reset block.	Assertion a_reset, scoreboard reset phase check.

FIFO Interface

```

FIFO.IF.sv > ...
1 interface FIFO_IF(clk);
2 parameter FIFO_WIDTH = 16;
3 parameter FIFO_DEPTH = 8;
4 input bit clk;
5 logic [FIFO_WIDTH-1:0] data_in;
6 bit rst_n, wr_en, rd_en;
7 logic [FIFO_WIDTH-1:0] data_out;
8 logic wr_ack, overflow;
9 logic full, empty, almostfull, almostempty, underflow;
10
11 localparam max_fifo_addr = $clog2(FIFO_DEPTH);
12
13 modport TEST (output data_in, wr_en, rd_en, rst_n,
14 |           input clk, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
15
16 modport DUT (input data_in, wr_en, rd_en, clk, rst_n,
17 |           output full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
18
19 modport MONITOR(input data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
20
21 endinterface
22

```

FIFO Design after fixing bugs :

```
1 //////////////////////////////////////////////////////////////////
2 module FIFO(FIFO_IF.DUT fifoif);
3
4 reg [fifoif.FIFO_WIDTH-1:0] mem [fifoif.FIFO_DEPTH-1:0];
5
6 reg [fifoif.max_fifo_addr-1:0] wr_ptr, rd_ptr;
7 reg [fifoif.max_fifo_addr:0] count;
8
9
10 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
11     if (!fifoif.rst_n) begin
12         wr_ptr <= 0;
13         fifoif.wr_ack <= 0; // bug
14         fifoif.overflow <= 0; //bug
15     end
16     else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH) begin
17         mem[wr_ptr] <= fifoif.data_in;
18         fifoif.wr_ack <= 1;
19         wr_ptr <= wr_ptr + 1;
20     end
21     else begin
22         fifoif.wr_ack <= 0;
23         if (fifoif.full && fifoif.wr_en && !fifoif.rd_en) //bug foget condition of !rd_en
24             fifoif.overflow <= 1;
25         else
26             fifoif.overflow <= 0;
27     end
28 end
29
30 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
31     if (!fifoif.rst_n) begin
32         rd_ptr <= 0;
33         fifoif.underflow<=0; //bug
34     end
35     else if (fifoif.rd_en && count != 0) begin
36         fifoif.data_out <= mem[rd_ptr];
37         rd_ptr <= rd_ptr + 1;
38     end
39     // underflow is a sequential output here
40     else begin
41         if(fifoif.empty && fifoif.rd_en && !fifoif.wr_en)
42             fifoif.underflow<=1;
43         else
44             fifoif.underflow<=0;
45     end
46 end
47
48 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
49     if (!fifoif.rst_n) begin
50         count <= 0;
51     end
52     else begin
53         if ( ({(fifoif.wr_en, fifoif.rd_en} == 2'b10) && !fifoif.full)
54             count <= count + 1;
55         else if ( ({(fifoif.wr_en, fifoif.rd_en} == 2'b01) && !fifoif.empty)
56             count <= count - 1;
57         else if ( ({(fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.empty)
58             count <= count+1;
59         else if ( ({(fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.full)
60             count <= count-1;
61     end
62 end
63
64 assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
65 assign fifoif.empty = (count == 0)? 1 : 0;
66 //bug underflow was combinational
67 assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
68 assign fifoif.almostempty = (count == 1)? 1 : 0;
69
70
```

FIFO Assertions :

```
77 // assertions
78 // 1
79 always_comb begin
80     if(!fifoif.rst_n)
81         a_reset: assert final( (wr_ptr==0) && (rd_ptr==0)  && (count==0));
82 end
83 // 2
84 property Write_Acknowledge;
85     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (fifoif.wr_en && !fifoif.full) |=> fifoif.wr_ack;
86 endproperty
87 // 3
88 property Overflow_Detection;
89     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (fifoif.wr_en && fifoif.full && !fifoif.rd_en) |=> fifoif.overflow ;
90 endproperty
91 // 4
92 property Underflow_Detection;
93     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (fifoif.rd_en && !fifoif.wr_en && fifoif.empty) |=> fifoif.underflow ;
94 endproperty
95 // 5
96 property Empty_Flag_Assertion;
97     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (count==0) |-> fifoif.empty ;
98 endproperty
99 // 6
100 property Full_Flag_Assertion;
101    @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (count == fifoif.FIFO_DEPTH) |-> fifoif.full ;
102 endproperty
103 // 7
104 property Almost_Full_Condition;
105    @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (count == fifoif.FIFO_DEPTH-1) |-> fifoif.almostfull ;
106 endproperty
107 // 8
108 property Almost_Empty_Condition;
109    @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (count == 1) |-> fifoif.almostempty ;
110 endproperty
111 // 9
112 property Pointer_Wraparound_1;
113     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  ( (wr_ptr==fifoif.FIFO_DEPTH-1) && fifoif.wr_en && (count<=fifoif.FIFO_DEPTH-1)) |=> $fell(wr_ptr);
114 endproperty
115 property Pointer_Wraparound_2;
116     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  ( (rd_ptr==fifoif.FIFO_DEPTH-1) && fifoif.rd_en && (count>0) ) |=> $fell(rd_ptr);
117 endproperty
118 property Pointer_Wraparound_3;
119     @ (!fifoif.rst_n)  ( $past(count)==fifoif.FIFO_DEPTH) |-> (count==0);
120 endproperty
121 // 10
122 property Pointer_threshold;
123     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  ((wr_ptr < fifoif.FIFO_DEPTH) && (rd_ptr < fifoif.FIFO_DEPTH) && (count <= fifoif.FIFO_DEPTH));
124 endproperty
125 // 11
126 property wr_rd_simul_empty;
127     @ (posedge fifoif.clk) disable iff (!fifoif.rst_n)  (fifoif.wr_en && fifoif.rd_en && fifoif.empty) |=> (!fifoif.empty);
128 endproperty
129
```

```

130 // 12
131 property wr_rd_simul_full;
132   @posedge fifoif.clk disable iff(!fifoif.rst_n)  (fifoif.wr_en && fifoif.rd_en && fifoif.full) |=> (!fifoif.full);
133 endproperty
134
135 assert property(Write_Acknowledge);
136 assert property(Overflow_Detection);
137 assert property(Underflow_Detection);
138 assert property(Empty_Flag_Assertion);
139 assert property(Full_Flag_Assertion);
140 assert property(Almost_Full_Condition);
141 assert property(Almost_Empty_Condition);
142 assert property(Pointer_Wraparound_1);
143 assert property(Pointer_Wraparound_2);
144 assert property(Pointer_Wraparound_3);
145 assert property(Pointer_threshold);
146 assert property (wr_rd_simul_empty);
147 assert property (wr_rd_simul_full);
148
149 // cover
150 cover final( (wr_ptr==0) && (rd_ptr==0) && (count==0));
151 cover property(Write_Acknowledge);
152 cover property(Overflow_Detection);
153 cover property(Underflow_Detection);
154 cover property(Empty_Flag_Assertion);
155 cover property(Full_Flag_Assertion);
156 cover property(Almost_Full_Condition);
157 cover property(Almost_Empty_Condition);
158 cover property(Pointer_Wraparound_1);
159 cover property(Pointer_Wraparound_2);
160 cover property(Pointer_Wraparound_3);
161 cover property(Pointer_threshold);
162 cover property (wr_rd_simul_empty);
163 cover property (wr_rd_simul_full);
164
165
166 endmodule
167

```

FIFO packages:

1) shared package

```
shared_pkg.sv > ...
1 package shared_pkg;
2 bit test_finished;
3 int error_count;
4 int correct_count;
5 event etrigger;
6 endpackage
7
```

2) Transaction package

```
FIFO_transaction_pkg.sv > ...
1 package FIFO_transaction_pkg;
2 parameter FIFO_WIDTH = 16;
3 parameter FIFO_DEPTH = 8;
4 class FIFO_transaction;
5 bit clk;
6 rand bit [FIFO_WIDTH-1:0] data_in;
7 rand bit rst_n, wr_en, rd_en;
8 logic [FIFO_WIDTH-1:0] data_out;
9 logic wr_ack, overflow;
10 logic full, empty, almostfull, almostempty, underflow;
11
12 int RD_EN_ON_DIST , WR_EN_ON_DIST;
13
14 ~function new(int RD_EN_ON_DIST=30 , WR_EN_ON_DIST=70);
15 | this.RD_EN_ON_DIST=RD_EN_ON_DIST ;
16 | this.WR_EN_ON_DIST=WR_EN_ON_DIST ;
17 endfunction
18
19 constraint reset{ rst_n dist {0:/10 , 1:/90};}
20 constraint WR_EN{ wr_en dist {1:/WR_EN_ON_DIST , 0:/(100-WR_EN_ON_DIST)};}
21 constraint RD_EN{ rd_en dist {1:/RD_EN_ON_DIST , 0:/(100-RD_EN_ON_DIST)};}
22
23 endclass
24 endpackage
25
```

3) coverage package

```
0 FIFO_coverage_pkg.sv > {} FIFO_coverage_pkg > FIFO_coverage
1 package FIFO_coverage_pkg;
2 import FIFO_transaction_pkg::*;
3 class FIFO_coverage;
4 FIFO_transaction F_cvg_txn=new();
5 covergroup g1;
6 WR_EN      : coverpoint F_cvg_txn.wr_en ;
7 RD_EN      : coverpoint F_cvg_txn.rd_en ;
8 WR_ACK     : coverpoint F_cvg_txn.wr_ack ;
9 OVERFLOW    : coverpoint F_cvg_txn.overflow ;
10 FULL       : coverpoint F_cvg_txn.overflow ;
11 EMPTY      : coverpoint F_cvg_txn.empty ;
12 ALMOSTFULL : coverpoint F_cvg_txn.almostfull ;
13 ALMOSTEMPTY : coverpoint F_cvg_txn.almostempty ;
14 UNDERFLOW   : coverpoint F_cvg_txn.underflow ;
15
16 wr_en_rd_en_WR_ACK      :cross WR_EN , RD_EN , WR_ACK ;
17 wr_en_rd_en_OVERFLOW    :cross WR_EN , RD_EN , OVERFLOW ;
18 wr_en_rd_en_FULL        :cross WR_EN , RD_EN , FULL ;
19 wr_en_rd_en_EMPTY        :cross WR_EN , RD_EN , EMPTY ;
20 wr_en_rd_en_ALMOSTFULL :cross WR_EN , RD_EN , ALMOSTFULL ;
21 wr_en_rd_en_ALMOSTEMPTY :cross WR_EN , RD_EN , ALMOSTEMPTY ;
22 wr_en_rd_en_UNDERFLOW   :cross WR_EN , RD_EN , UNDERFLOW ;
23 endgroup
24
25 //constructor
26 function new();
27   g1 =new;
28 endfunction
29 //
30 function void sample_data(input FIFO_transaction F_txn);
31   F_cvg_txn=F_txn;
32   g1.sample();
33 endfunction
34
35 endclass
36 endpackage
```

4) scoreboard package

```
1  FIFO_scoreboard_pkg.sv > () FIFO_scoreboard_pkg > FIFO_scoreboard
2  package FIFO_scoreboard_pkg;
3  import FIFO_transaction_pkg::*;
4  import shared_pkg::*;
5  logic full_ref=0, empty_ref=1,almostfull_ref=0,almostempty_ref=0,overflow_ref=0,underflow_ref=0;
6  logic [FIFO_WIDTH-1:0] data_out_ref;
7  class FIFO_scoreboard;
8  // logic full_ref=0, empty_ref=1,almostfull_ref=0,almostempty_ref=0,overflow_ref=0,underflow_ref=0;
9  logic [FIFO_WIDTH-1:0]data_mem [$];
10
11 function void reference_model(input FIFO_transaction F_ref_txn);
12
13 if(F_ref_txn.rst_n)begin
14   data_mem.delete();
15   full_ref<=0;
16   empty_ref<=1;
17   overflow_ref<=0;
18   underflow_ref<=0;
19 end
20 else begin
21
22 if(F_ref_txn.wr_en && (data_mem.size() < FIFO_DEPTH) )
23   data_mem.push_back(F_ref_txn.data_in);
24 if(F_ref_txn.rd_en && !empty_ref)
25   data_out_ref=data_mem.pop_front();
26
27 if(F_ref_txn.wr_en && !F_ref_txn.rd_en && full_ref )
28   overflow_ref<=1;
29 else
30   overflow_ref<=0;
31
32 if(F_ref_txn.rd_en && !F_ref_txn.wr_en && empty_ref)
33   underflow_ref<=1;
34 else
35   underflow_ref<=0;
36
37 if (data_mem.size()==FIFO_DEPTH) full_ref<=1;
38 else full_ref<=0;
39
40 if(data_mem.size()==0) empty_ref<=1;
41 else empty_ref<=0;
42
43 end
44 if(data_mem.size() == FIFO_DEPTH-1) almostfull_ref <= 1 ;
45 else almostfull_ref<= 0;
46
47 if(data_mem.size() == 1) almostempty_ref <=1 ;
48 else almostempty_ref <= 0;
49
50 endfunction
51
52 task check_data(input FIFO_transaction F_chk_txn);
53 reference_model(F_chk_txn);
54 if(data_out_ref!=F_chk_txn.data_out || full_ref!=F_chk_txn.full || empty_ref!=F_chk_txn.empty
55 ||almostfull_ref!=F_chk_txn.almostfull ||almostempty_ref!=F_chk_txn.almostempty
56 ||overflow_ref!=F_chk_txn.overflow|| underflow_ref!=F_chk_txn.underflow )begin
57   $display("ERROR: incorrect data_out , data_out=%h ,EXPECTED=%h,F_chk_txn.data_out,data_out_ref");
58   $display("rst_n=%b , wr_en=%b , rd_en=%b , data_in=%h , full=%b ,full_ref=%b ,empty=%b ,empty_ref=%b ,
59   almostfull=%b,almostfull_ref=%b ,almostempty=%b,almostempty_ref=%b,overflow=%b,overflow_ref=%b,underflow=%b ,
60   underflow_ref=%b ",
61   F_chk_txn.rst_n,F_chk_txn.wr_en,F_chk_txn.rd_en,F_chk_txn.data_in,F_chk_txn.full,full_ref,F_chk_txn.empty,empty_ref,
62   F_chk_txn.almostfull,almostfull_ref,F_chk_txn.almostempty,almostempty_ref,F_chk_txn.overflow,overflow_ref,
63   F_chk_txn.underflow, underflow_ref);
64   $display("-----");
65   error_count=error_count+1;
66 end
67 else begin
68   correct_count=correct_count+1;
69 end
70
71 endtask
72 endclass
73 endpackage
74
```

FIFO Monitor :

```
❸ FIFO_monitor.sv > ...
 1  import FIFO_transaction_pkg::*;
 2  import FIFO_scoreboard_pkg::*;
 3  import FIFO_coverage_pkg::*;
 4  import shared_pkg::*;

 5
 6  module FIFO_monitor(FIFO_IF.MONITOR fifoif);
 7    FIFO_transaction fifo_txn;
 8    FIFO_scoreboard fifo_board;
 9    FIFO_coverage fifo_cvg;
10
11  initial begin
12    fifo_txn=new();
13    fifo_board=new();
14    fifo_cvg=new();
15
16    forever begin
17
18      //    wait(etrigger.triggered);
19      @etrigger;
20
21      fifo_txn.data_in=fifoif.data_in;
22      fifo_txn.wr_en=fifoif.wr_en;
23      fifo_txn.rd_en=fifoif.rd_en;
24      fifo_txn.rst_n=fifoif.rst_n;
25      fifo_txn.full=fifoif.full;
26      fifo_txn.empty=fifoif.empty;
27      fifo_txn.almostfull=fifoif.almostfull;
28      fifo_txn.almostempty=fifoif.almostempty;
29      fifo_txn.wr_ack=fifoif.wr_ack;
30      fifo_txn.overflow=fifoif.overflow;
31      fifo_txn.underflow=fifoif.underflow;
32      fifo_txn.data_out=fifoif.data_out;
33
34    fork
35      begin
36        fifo_cvg.sample_data(fifo_txn);
37      end
38
39      begin
40        fifo_board.check_data(fifo_txn) ;
41      end
42    join
43
44    if(test_finished==1)begin
45      $display("error_count =%d  ,correct_count =%d ",error_count,correct_count);
46      $stop;
47    end
48  end
49 end
50
51 endmodule
52
```

FIFO Testbench :

```
④ FIFO_tb.sv > FIFO_tb
  1 import FIFO_transaction_pkg::*;
  2 import FIFO_scoreboard_pkg::*;
  3 import FIFO_coverage_pkg::*;
  4 import shared_pkg::*;
  5 module FIFO_tb(FIFO_IF.TEST fifoif);
  6
  7 FIFO_transaction fifo_txn_tb=new();
  8
  9 initial begin
10     fifo_txn_tb.data_in=0;
11     fifo_txn_tb.wr_en=0;
12     fifo_txn_tb.rd_en=0;
13     fifo_txn_tb.rst_n=0;
14
15     fifoif.rst_n=0;
16     @(negedge fifoif.clk);
17     fifoif.rst_n=1;
18
19 repeat(100000)begin
20     @(negedge fifoif.clk);
21     assert(fifo_txn_tb.randomize());
22     fifoif.data_in=fifo_txn_tb.data_in;
23     fifoif.wr_en=fifo_txn_tb.wr_en;
24     fifoif.rd_en=fifo_txn_tb.rd_en;
25     fifoif.rst_n=fifo_txn_tb.rst_n;
26
27
28
29     -> etrigger;
30
31 end
32     test_finished=1;
33 end
34
35 endmodule
36
```

FIFO TOP Module :

```
❶ FIFO_TOP.sv > ...
1  module top();
2  bit clk;
3
4  initial begin
5      clk=1;
6      forever begin
7          #1 clk=~clk;
8      end
9  end
10
11 FIFO_IF fifoif(clk);
12
13 FIFO    DUT(fifoif);
14
15 FIFO_tb TEST(fifoif);
16
17 FIFO_monitor MONITOR(fifoif);
18
19 always_comb begin
20     if(!fifoif.rst_n)
21         a_reset: assert final(!fifoif.full&& fifoif.empty&& !fifoif.almostfull && !fifoif.almostempty
22                                && !fifoif.wr_ack && !fifoif.overflow  && !fifoif.underflow);
23 end
24
25 endmodule
26
```

SRC_files & DO_file :

```
src_file.list
1 FIFO.IF.sv
2 FIFO.sv
3 shared_pkg.sv
4 FIFO_coverage_pkg.sv
5 FIFO_transaction_pkg.sv
6 FIFO_scoreboard_pkg.sv
7 FIFO_TOP.sv
8 FIFO_tb.sv
9 FIFO_monitor.sv
10
```

```
run_FIFO.do
1 vlib work
2 vlog -f src_file.list +cover -covercells
3 vsim -voptargs=+acc work.top -cover -sv_seed random
4 add wave /top/fifoif/*
5 add wave -position insertpoint sim:/FIFO_scoreboard_pkg/*
6 coverage save TOP_FIFO.ucdb -du FIFO -onexit
7 run -all
8
```

FIFO Results

```
#      Unable to replace existing ini file (D:/Digital_verification_diploma/project_1_FIFO/SV/project_1.mprj). .ini file can not be renamed.
VSM10> restart
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.top(fast)
# Loading work.FIFO_IF(fast_1)
# Loading work.FIFO(fast)
# Loading work.FIFO_transaction_pkg(fast)
# Loading work.FIFO_coverage_pkg(fast)
# Loading work.shared_pkg(fast)
# Loading work.FIFO_scoreboard_pkg(fast)
# Loading work.FIFO_tb_sv_unit(fast)
# Loading work.FIFO_tb(fast)
# Loading work.FIFO_monitor_sv_unit(fast)
# Loading work.FIFO_monitor(fast)
VSM11> run -all
# error_count =      0 ,correct_count =    100000
# ** Note: $stop : D:/Digital_verification_diploma/project_1_FIFO/SV/FIFO_monitor.sv(48)
#   Time: 200001 ns Iteration: 2 Instance: /top/MONITOR
# Break in Module FIFO_monitor at D:/Digital_verification_diploma/project_1_FIFO/SV/FIFO_monitor.sv line 48
```

CODE coverage report :

● Statement coverage

Code Coverage Analysis
Statements - by instance (top/DUT) Statement

FIFO.sv

```
15 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
17 wr_ptr <= 0;
18 fifoif.wr_ack <= 0; // bug
19 fifoif.overflow <= 0; //bug
23 mem[wr_ptr] <= fifoif.data_in;
24 fifoif.wr_ack <= 1;
25 wr_ptr <= wr_ptr + 1;
28 fifoif.wr_ack <= 0;
30 fifoif.overflow <= 1;
32 fifoif.overflow <= 0;
35 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
38 rd_ptr <= 0;
39 fifoif.underflow<=0; //bug
42 fifoif.data_out <= mem[rd_ptr];
43 rd_ptr <= rd_ptr + 1;
48 fifoif.underflow<=1;
50 fifoif.underflow<=0;
54 always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
56 count <= 0;
60 count <= count + 1;
62 count <= count - 1;
64 count <= count+1;
66 count <= count-1;
70 assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
71 assign fifoif.empty = (count == 0)? 1 : 0;
73 assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
74 assign fifoif.almostempty = (count == 1)? 1 : 0;
80 always_comb begin
```

● Branch coverage

Code Coverage Analysis
Branches - by instance (top/DUT) Branch

FIFO.sv

```
16 if (!fifoif.rst_n) begin
22 else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH) begin
27 else begin
29 if (fifoif.full && fifoif.wr_en && !fifoif.rd_en) //bug foget condition of !rd_en
31 else
37 if (!fifoif.rst_n) begin
41 else if (fifoif.rd_en && count != 0) begin
46 else begin
47 if(fifoif.empty && fifoif.rd_en && !fifoif.wr_en)
49 else
55 if (!fifoif.rst_n) begin
58 else begin
59 if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b10) && !fifoif.full)
61 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b01) && !fifoif.empty)
63 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b11) && !fifoif.empty)
65 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b11) && !fifoif.full)
70 assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
71 assign fifoif.empty = (count == 0)? 1 : 0;
73 assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
74 assign fifoif.almostempty = (count == 1)? 1 : 0;
81 if(!fifoif.rst_n)
```

● Condition coverage

Code Coverage Analysis
Conditions - by instance (top/DUT) Condition

FIFO.sv

```
22 else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH) begin
29 if (fifoif.full && fifoif.wr_en && !fifoif.rd_en) //bug foget condition of !rd_en
41 else if (fifoif.rd_en && count != 0) begin
47 if(fifoif.empty && fifoif.rd_en && !fifoif.wr_en)
59 if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b10) && !fifoif.full)
61 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b01) && !fifoif.empty)
63 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b11) && !fifoif.empty)
65 else if ( ( (fifoif.wr_en, fifoif.rd_en) == 2'b11) && !fifoif.full)
70 assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
71 assign fifoif.empty = (count == 0)? 1 : 0;
73 assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
74 assign fifoif.almostempty = (count == 1)? 1 : 0;
```

● Toggle coverage

Code Coverage Analysis

Toggles - by instance (/top/fifoif)

- sim:/top/fifoif
 - ✓ almostempty
 - ✓ almostfull
 - ✓ clk
 - +✓ data_in
 - +✓ data_out
 - ✓ empty
 - ✓ full
 - ✓ overflow
 - ✓ rd_en
 - ✓ rst_n
 - ✓ underflow
 - ✓ wr_ack
 - ✓ wr_en

Code Coverage Analysis

Toggles - by instance (/top/DUT)

- sim:/top/DUT
 - +✓ count
 - +✓ rd_ptr
 - +✓ wr_ptr

Branch Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
-----	-----	-----	-----	-----
Branches	27	27	0	100.00%

Branch Details

Branch Coverage for instance /\top#DUT

Line	Item	Count	Source
---	---	-----	-----
File FIFO.sv			

-----IF Branch-----

Line	Item	Count	Source
16	1	109052	Count coming in to IF
16	1	19120	if (!fifoif.rst_n) begin
22	1	56238	else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH) begin
27	1	33694	else begin

Branch totals: 3 hits of 3 branches = 100.00%

-----IF Branch-----

Line	Item	Count	Source
29	1	33694	Count coming in to IF
29	1	4529	if (fifoif.full && fifoif.wr_en && !fifoif.rd_en) //bug forgot condition of !rd_en
31	1	29165	else

Branch totals: 2 hits of 2 branches = 100.00%

-----IF Branch-----

Line	Item	Count	Source
37	1	109052	Count coming in to IF
37	1	19120	if (!fifoif.rst_n) begin
41	1	22689	else if (fifoif.rd_en && count != 0) begin
46	1	67243	else begin

Branch totals: 3 hits of 3 branches = 100.00%

-----IF Branch-----

Line	Item	Count	Source
47	1	67243	Count coming in to IF
47	1	1290	if(fifoif.empty && fifoif.rd_en && !fifoif.wr_en)
49	1	65953	else

Branch totals: 2 hits of 2 branches = 100.00%

```

-----IF Branch-----
55      1          100294  Count coming in to IF
55      1          18762   if (!fifoif.rst_n) begin
58      1          81532   else begin
Branch totals: 2 hits of 2 branches = 100.00%
-----IF Branch-----
59      1          81532   Count coming in to IF
59      1          39273   if ( {{fifoif.wr_en, fifoif.rd_en} == 2'b10} && !fifoif.full)
61      1          6755    else if ( {{fifoif.wr_en, fifoif.rd_en} == 2'b01} && !fifoif.empty)
63      1          3027    else if ( {{fifoif.wr_en, fifoif.rd_en} == 2'b11} && fifoif.empty)
65      1          1996    else if ( {{fifoif.wr_en, fifoif.rd_en} == 2'b11} && fifoif.full)
30481   All False Count
Branch totals: 5 hits of 5 branches = 100.00%
-----IF Branch-----
70      1          59532   Count coming in to IF
70      1          3898    assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
70      2          55634   assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%
-----IF Branch-----
71      1          59532   Count coming in to IF
71      1          9911    assign fifoif.empty = (count == 0)? 1 : 0;
71      2          49621   assign fifoif.empty = (count == 0)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%
-----IF Branch-----
73      1          59532   Count coming in to IF
73      1          5484    assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
73      2          54048   assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
Branch totals: 2 hits of 2 branches = 100.00%
-----IF Branch-----
74      1          59532   Count coming in to IF
74      1          11130   assign fifoif.almostempty = (count == 1)? 1 : 0;
74      2          48402   assign fifoif.almostempty = (count == 1)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%
-----IF Branch-----
81      1          91768   Count coming in to IF
81      1          17727   if(!fifoif.rst_n)
74041   All False Count
Branch totals: 2 hits of 2 branches = 100.00%
Condition Coverage:
Enabled Coverage      Bins   Covered   Misses   Coverage
----- -----
Conditions           26       26        0     100.00%
=====
Condition Details=====
Condition Coverage for instance /\top#DUT --
File FIFO.sv
-----Focused Condition View-----
Line    22 Item   1 ((fifoif.wr_en && (count < fifoif.FIFO_DEPTH))
Condition totals: 2 of 2 input terms covered = 100.00%
      Input Term   Covered   Reason for no coverage   Hint
      ----- ----- -----
      fifoif.wr_en   Y
      (count < fifoif.FIFO_DEPTH)   Y
      Rows:      Hits   FEC Target
      ----- -----
      Row 1:      1   fifoif.wr_en_0
      Row 2:      1   fifoif.wr_en_1
      Row 3:      1   (count < fifoif.FIFO_DEPTH)_0
      Row 4:      1   (count < fifoif.FIFO_DEPTH)_1
      Non-masking condition(s)
-----Focused Condition View-----
Line    29 Item   1 ((fifoif.full && fifoif.wr_en) && ~fifoif.rd_en)
Condition totals: 3 of 3 input terms covered = 100.00%
      Input Term   Covered   Reason for no coverage   Hint
      ----- -----
      fifoif.full   Y
      fifoif.wr_en   Y
      fifoif.rd_en   Y

```

Rows:	Hits	FEC Target	Non-masking condition(s)
Row 1:	1	fifoif.full_0	-
Row 2:	1	fifoif.full_1	(~fifoif.rd_en && fifoif.wr_en)
Row 3:	1	fifoif.wr_en_0	fifoif.full
Row 4:	1	fifoif.wr_en_1	(~fifoif.rd_en && fifoif.full)
Row 5:	1	fifoif.rd_en_0	(fifoif.full && fifoif.wr_en)
Row 6:	1	fifoif.rd_en_1	(fifoif.full && fifoif.wr_en)

-----Focused Condition View-----			
Line	41	Item	1 (fifoif.rd_en && (count != 0))
Condition totals: 2 of 2 input terms covered = 100.00%			
Input Term	Covered	Reason for no coverage	Hint
fifoif.rd_en	Y		
(count != 0)	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
Row 1:	1	fifoif.rd_en_0	-
Row 2:	1	fifoif.rd_en_1	(count != 0)
Row 3:	1	(count != 0)_0	fifoif.rd_en
Row 4:	1	(count != 0)_1	fifoif.rd_en

-----Focused Condition View-----			
Line	47	Item	1 ((fifoif.empty && fifoif.rd_en) && ~fifoif.wr_en)
Condition totals: 3 of 3 input terms covered = 100.00%			

Input Term	Covered	Reason for no coverage	Hint
fifoif.empty	Y		
fifoif.rd_en	Y		
fifoif.wr_en	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
Row 1:	1	fifoif.empty_0	(~fifoif.wr_en && fifoif.rd_en)
Row 2:	1	fifoif.empty_1	fifoif.empty
Row 3:	1	fifoif.rd_en_0	(~fifoif.wr_en && fifoif.empty)
Row 4:	1	fifoif.rd_en_1	(~fifoif.wr_en && fifoif.empty)
Row 5:	1	fifoif.wr_en_0	(fifoif.empty && fifoif.rd_en)
Row 6:	1	fifoif.wr_en_1	(fifoif.empty && fifoif.rd_en)

-----Focused Condition View-----			
Line	59	Item	1 ((~fifoif.rd_en && fifoif.wr_en) && ~fifoif.full)
Condition totals: 3 of 3 input terms covered = 100.00%			

Input Term	Covered	Reason for no coverage	Hint
fifoif.rd_en	Y		
fifoif.wr_en	Y		
fifoif.full	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
Row 1:	1	fifoif.rd_en_0	(~fifoif.full && fifoif.wr_en)
Row 2:	1	fifoif.rd_en_1	fifoif.rd_en
Row 3:	1	fifoif.wr_en_0	(~fifoif.full && ~fifoif.rd_en)
Row 4:	1	fifoif.wr_en_1	(~fifoif.full && ~fifoif.rd_en)
Row 5:	1	fifoif.full_0	(fifoif.rd_en && fifoif.wr_en)
Row 6:	1	fifoif.full_1	(fifoif.rd_en && fifoif.wr_en)

-----Focused Condition View-----			
Line	61	Item	1 ((fifoif.rd_en && ~fifoif.wr_en) && ~fifoif.empty)
Condition totals: 3 of 3 input terms covered = 100.00%			

Input Term	Covered	Reason for no coverage	Hint
fifoif.rd_en	Y		
fifoif.wr_en	Y		
fifoif.empty	Y		

Rows:	Hits	FEC Target	Non-masking condition(s)
Row 1:	1	fifoif.rd_en_0	-
Row 2:	1	fifoif.rd_en_1	(~fifoif.empty && ~fifoif.wr_en)

```

        ,
Row  3:      1 fifoif.wr_en_0      fifoif.rd_en
Row  4:      1 fifoif.wr_en_1      (fifoif.empty && fifoif.rd_en)
Row  5:      1 fifoif.empty_0      (fifoif.rd_en && fifoif.wr_en)
Row  6:      1 fifoif.empty_1      (fifoif.rd_en && fifoif.wr_en)

-----Focused Condition View-----
Line    65 Item  1 ((fifoif.rd_en && fifoif.wr_en) && fifoif.full)
Condition totals: 3 of 3 input terms covered = 100.00%
Input Term  Covered  Reason for no coverage  Hint
-----
fifoif.rd_en      Y
fifoif.wr_en      Y
fifoif.full       Y

Rows:   Hits  FEC Target      Non-masking condition(s)
-----
Row  1:      1 fifoif.rd_en_0      -
Row  2:      1 fifoif.rd_en_1      (fifoif.full && fifoif.wr_en)
Row  3:      1 fifoif.wr_en_0      fifoif.rd_en
Row  4:      1 fifoif.wr_en_1      (fifoif.full && fifoif.rd_en)
Row  5:      1 fifoif.full_0      (fifoif.rd_en && fifoif.wr_en)
Row  6:      1 fifoif.full_1      (fifoif.rd_en && fifoif.wr_en)

-----Focused Condition View-----
Line    70 Item  1 (count == fifoif.FIFO_DEPTH)
Condition totals: 1 of 1 input term covered = 100.00%
Input Term  Covered  Reason for no coverage  Hint
-----
(count == fifoif.FIFO_DEPTH)      Y

Rows:   Hits  FEC Target      Non-masking condition(s)
-----
Row  1:      1 (count == fifoif.FIFO_DEPTH)_0      -
Row  2:      1 (count == fifoif.FIFO_DEPTH)_1      -

-----Focused Condition View-----
Line    71 Item  1 (count == 0)
Condition totals: 1 of 1 input term covered = 100.00%
Input Term  Covered  Reason for no coverage  Hint
-----
(count == 0)      Y

| Rows:   Hits  FEC Target      Non-masking condition(s)
| -----
| Row  1:      1 (count == 0)_0      -
|
| Rows:   Hits  FEC Target      Non-masking condition(s)
| -----
| Row  1:      1 (count == 0)_0      -
| Row  2:      1 (count == 0)_1      -

-----Focused Condition View-----
Line    73 Item  1 (count == (fifoif.FIFO_DEPTH - 1))
Condition totals: 1 of 1 input term covered = 100.00%
Input Term  Covered  Reason for no coverage  Hint
-----
(count == (fifoif.FIFO_DEPTH - 1))      Y

Rows:   Hits  FEC Target      Non-masking condition(s)
-----
Row  1:      1 (count == (fifoif.FIFO_DEPTH - 1))_0      -
Row  2:      1 (count == (fifoif.FIFO_DEPTH - 1))_1      -

-----Focused Condition View-----
Line    74 Item  1 (count == 1)
Condition totals: 1 of 1 input term covered = 100.00%
Input Term  Covered  Reason for no coverage  Hint
-----
(count == 1)      Y

Rows:   Hits  FEC Target      Non-masking condition(s)
-----
Row  1:      1 (count == 1)_0      -
Row  2:      1 (count == 1)_1      -


Statement Coverage:
  Enabled Coverage      Bins      Hits      Misses      Coverage
  -----      -----      -----      -----      -----
  Statements          28          28          0      100.00%


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

Statement Coverage for instance /top#DUT --
Line     Item      Count      Source
----      ---      ----      -----
File FIFO.sv
8           module FIFO(FIFO_IF.DUT fifoif);
9
10          reg [fifoif.FIFO_WIDTH-1:0] mem [fifoif.FIFO_DEPTH-1:0];
11
12          reg [fifoif.max_fifo_addr-1:0] wr_ptr, rd_ptr;
13          reg [fifoif.max_fifo_addr:0] count;
14
15          1      109052  always @ (posedge fifoif.clk or negedge fifoif.rst_n) begin
16                      if (!fifoif.rst_n) begin
17                          wr_ptr <= 0;
18                          fifoif.wr_ack <= 0; // bug
19                          fifoif.overflow <= 0; //bug
20
21                      end
22                      else if (fifoif.wr_en && count < fifoif.FIFO_DEPTH) begin
23                          mem[wr_ptr] <= fifoif.data_in;
24                          fifoif.wr_ack <= 1;
25                          wr_ptr <= wr_ptr + 1;

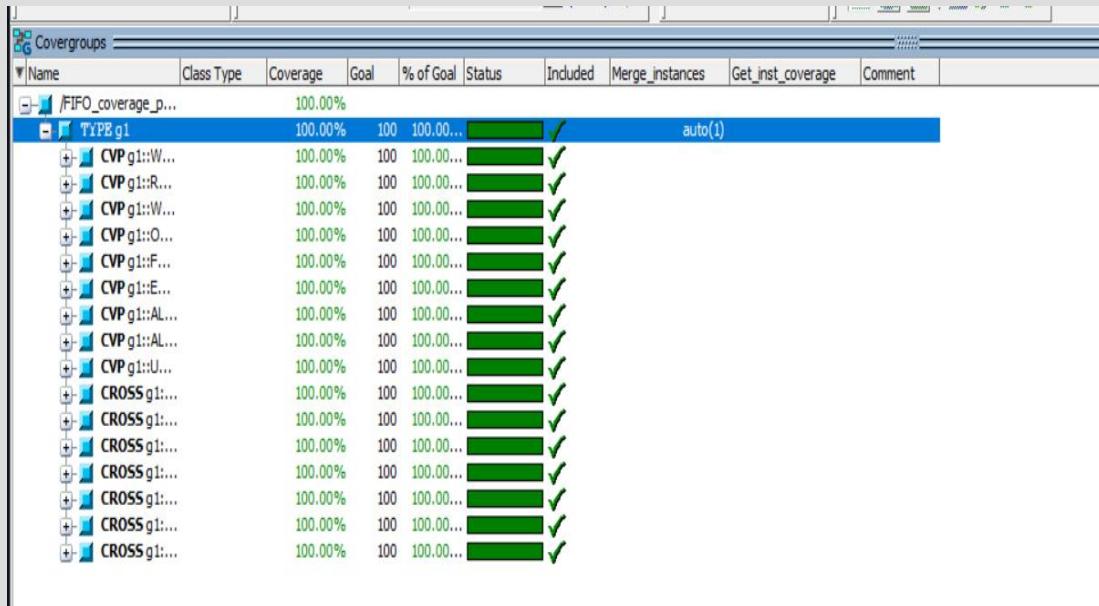
```

```

26           end
27           else begin
28             1           33694      fifoif.wr_ack <= 0;
29           if (fifoif.full && fifoif.wr_en && !fifoif.rd_en) //bug fotet condition of !rd_en
30             1           4529       fifoif.overflow <= 1;
31           else
32             1           29165      fifoif.overflow <= 0;
33           end
34           end
35
36             1           109052     always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
37               if (!fifoif.rst_n) begin
38                 1           19120      rd_ptr <= 0;
39                 1           19120      fifoif.underflow<=0; //bug
40               end
41               else if (fifoif.rd_en && count != 0) begin
42                 1           22689      fifoif.data_out <= mem[rd_ptr];
43                 1           22689      rd_ptr <= rd_ptr + 1;
44               end
45               // underflow is a sequential output here
46             else begin
47               if(fifoif.empty && fifoif.rd_en && !fifoif.wr_en)
48                 1           1290       fifoif.underflow<=1;
49             else
50               1           65953      fifoif.underflow<=0;
51             end
52           end
53
54             1           100294     always @(posedge fifoif.clk or negedge fifoif.rst_n) begin
55               if (!fifoif.rst_n) begin
56                 1           18762      count <= 0;
57               end
58             else begin
59               if      ( ({fifoif.wr_en, fifoif.rd_en} == 2'b10) && !fifoif.full)
60                 1           39273      count <= count + 1;
61               else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b01) && !fifoif.empty)
62                 1           6755       count <= count - 1;
63               else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.empty)
64                 1           3027       count <= count+1;
65               else if ( ({fifoif.wr_en, fifoif.rd_en} == 2'b11) && fifoif.full)
66                 1           1996       count <= count-1;
67             end
68           end
69
70             1           59533      assign fifoif.full = (count == fifoif.FIFO_DEPTH)? 1 : 0;
71             1           59533      assign fifoif.empty = (count == 0)? 1 : 0;
72             //bug underflow was combinational
73             1           59533      assign fifoif.almostfull = (count == fifoif.FIFO_DEPTH-1)? 1 : 0; //bug fifoif.FIFO_DEPTH-2
74             1           59533      assign fifoif.almostempty = (count == 1)? 1 : 0;
75
76
77
78           // assertions
79           // 1
80             1           91768      always_comb begin
81
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 "Coverage"
-----  -----
count[3:0]                   1        1    100.00
rd_ptr[2:0]                  1        1    100.00
wr_ptr[2:0]                  1        1    100.00
Total Node Count      =      10
Toggled Node Count     =      10
Untoggled Node Count   =      0
Toggle Coverage      = 100.00% (20 of 20 bins)

```

Functional coverage report :



```
Coverage Report by instance with details
=====
== Instance: /FIFO_coverage_pkg
== Design Unit: work.FIFO_coverage_pkg
=====

Coverage Coverage:
  Covergroups      1      na      na  100.00%
  Coverpoints/Crosses 16      na      na
  Covergroup Bins  74      74      0   100.00%
-----
Covergroup          Metric    Goal     Bins Status
-----
TYPE /FIFO_coverage_pkg/FIFO_coverage/g1      100.00%    100      -  Covered
  covered/total bins:           74      74      -
  missing/total bins:          0       74      -
  % Hit:                      100.00%    100      -
  Coverpoint WR_EN            100.00%    100      -  Covered
    covered/total bins:         2       2       -
    missing/total bins:         0       2       -
    % Hit:                      100.00%    100      -
    bin auto[0]                30349     1       -
    bin auto[1]                69651     1       -
  Coverpoint RD_EN            100.00%    100      -  Covered
    covered/total bins:         2       2       -
    missing/total bins:         0       2       -
    % Hit:                      100.00%    100      -
    bin auto[0]                69788     1       -
    bin auto[1]                30212     1       -
  Coverpoint WR_ACK           100.00%    100      -  Covered
    covered/total bins:         2       2       -
    missing/total bins:         0       2       -
    % Hit:                      100.00%    100      -
    bin auto[0]                43729     1       -
    bin auto[1]                56271     1       -
  Coverpoint OVERFLOW          100.00%    100      -  Covered
    covered/total bins:         2       2       -
    missing/total bins:         0       2       -
    % Hit:                      100.00%    100      -
    bin auto[0]                95492     1       -
    bin auto[1]                4508      1       -
  Coverpoint FULL              100.00%    100      -  Covered
    covered/total bins:         2       2       -
    missing/total bins:         0       2       -
    % Hit:                      100.00%    100      -
    bin auto[0]                95492     1       -
```

bin auto[1]	4508	1	-	Covered
Coverpoint EMPTY	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	84254	1	-	Covered
bin auto[1]	15746	1	-	Covered
Coverpoint ALMOSTFULL	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	91112	1	-	Covered
bin auto[1]	8888	1	-	Covered
Coverpoint ALMOSSTEMPTY	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	82139	1	-	Covered
bin auto[1]	17861	1	-	Covered
Coverpoint UNDERFLOW	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	98658	1	-	Covered
bin auto[1]	1342	1	-	Covered
Cross wr_en_rd_en_WR_ACK	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	11754	1	-	Covered
bin <auto[0],auto[1],auto[1]>	5180	1	-	Covered
bin <auto[1],auto[0],auto[1]>	27342	1	-	Covered
bin <auto[0],auto[0],auto[1]>	11995	1	-	Covered
bin <auto[1],auto[1],auto[0]>	9281	1	-	Covered
bin <auto[0],auto[1],auto[0]>	3997	1	-	Covered
bin <auto[1],auto[0],auto[0]>	21274	1	-	Covered
bin <auto[0],auto[0],auto[0]>	9177	1	-	Covered
Cross wr_en_rd_en_OVERFLOW	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	945	1	-	Covered
bin <auto[0],auto[1],auto[1]>	426	1	-	Covered
bin <auto[1],auto[0],auto[1]>	2251	1	-	Covered
bin <auto[0],auto[0],auto[1]>	886	1	-	Covered
bin <auto[1],auto[1],auto[0]>	20090	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8751	1	-	Covered
bin <auto[1],auto[0],auto[0]>	46365	1	-	Covered
bin <auto[0],auto[0],auto[0]>	20286	1	-	Covered
Cross wr_en_rd_en_FULL	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	945	1	-	Covered
bin <auto[0],auto[1],auto[1]>	426	1	-	Covered
bin <auto[1],auto[0],auto[1]>	2251	1	-	Covered
bin <auto[0],auto[0],auto[1]>	886	1	-	Covered
bin <auto[1],auto[1],auto[0]>	20090	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8751	1	-	Covered
bin <auto[1],auto[0],auto[0]>	46365	1	-	Covered
bin <auto[0],auto[0],auto[0]>	20286	1	-	Covered
Cross wr_en_rd_en_EMPTY	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3358	1	-	Covered
bin <auto[0],auto[1],auto[1]>	1470	1	-	Covered
bin <auto[1],auto[0],auto[1]>	7592	1	-	Covered
bin <auto[0],auto[0],auto[1]>	3326	1	-	Covered
bin <auto[1],auto[1],auto[0]>	17677	1	-	Covered
bin <auto[0],auto[1],auto[0]>	7707	1	-	Covered
bin <auto[1],auto[0],auto[0]>	41024	1	-	Covered
bin <auto[0],auto[0],auto[0]>	17846	1	-	Covered
Cross wr_en_rd_en_ALMOSTFULL	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	1860	1	-	Covered
bin <auto[0],auto[1],auto[1]>	761	1	-	Covered
bin <auto[1],auto[0],auto[1]>	4325	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1942	1	-	Covered
bin <auto[1],auto[1],auto[0]>	19175	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8416	1	-	Covered
bin <auto[1],auto[0],auto[0]>	44291	1	-	Covered
bin <auto[0],auto[0],auto[0]>	19230	1	-	Covered
Cross wr_en_rd_en_ALMOSSTEMPTY	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3727	1	-	Covered

File: C:\Users\Yasser\Documents\NetBeansProjects\Project1\FIFO_coverage_pkg\FIFO_coverage\g1					
Covergroup Coverage:					
Covergroup	Metric	Goal	Bins	Status	
TYPE /FIFO_coverage_pkg/FIFO_coverage/g1	100.00%	100	-	Covered	
covered/total bins:	74	74	-		
missing/total bins:	0	74	-		
% Hit:	100.00%	100	-		
Coverpoint WR_EN	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	30349	1	-	Covered	
bin auto[1]	69651	1	-	Covered	
Coverpoint RD_EN	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	69788	1	-	Covered	
bin auto[1]	30212	1	-	Covered	
Coverpoint WR_ACK	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	43729	1	-	Covered	
bin auto[1]	56271	1	-	Covered	
Coverpoint OVERFLOW	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	95492	1	-	Covered	
bin auto[1]	4508	1	-	Covered	
Coverpoint FULL	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	95492	1	-	Covered	
bin auto[1]	4508	1	-	Covered	
Coverpoint EMPTY	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	84254	1	-	Covered	
bin auto[1]	15746	1	-	Covered	
Coverpoint ALMOSTFULL	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	91112	1	-	Covered	
bin auto[1]	8888	1	-	Covered	
Coverpoint ALMOSSTEMPTY	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	82139	1	-	Covered	
bin auto[1]	17861	1	-	Covered	
Coverpoint UNDERFLOW	100.00%	100	-	Covered	
covered/total bins:	2	2	-		
missing/total bins:	0	2	-		
% Hit:	100.00%	100	-		
bin auto[0]	98658	1	-	Covered	
bin auto[1]	1342	1	-	Covered	
Cross wr_en_rd_en_WR_ACK	100.00%	100	-	Covered	
covered/total bins:	8	8	-		
missing/total bins:	0	8	-		
% Hit:	100.00%	100	-		
Auto, Default and User Defined Bins:					
bin <auto[1],auto[1],auto[1]>	11754	1	-	Covered	
bin <auto[0],auto[1],auto[1]>	5180	1	-	Covered	
bin <auto[1],auto[0],auto[1]>	27342	1	-	Covered	
bin <auto[0],auto[0],auto[1]>	11995	1	-	Covered	
bin <auto[1],auto[1],auto[0]>	9281	1	-	Covered	
bin <auto[0],auto[1],auto[0]>	3997	1	-	Covered	
bin <auto[1],auto[0],auto[0]>	21274	1	-	Covered	

bin <auto[0],auto[0],auto[0]>	9177	1	-	Covered
Cross wr_en_rd_en_OVERFLOW	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	945	1	-	Covered
bin <auto[0],auto[1],auto[1]>	426	1	-	Covered
bin <auto[1],auto[0],auto[1]>	2251	1	-	Covered
bin <auto[0],auto[0],auto[1]>	886	1	-	Covered
bin <auto[1],auto[1],auto[0]>	20090	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8751	1	-	Covered
bin <auto[1],auto[0],auto[0]>	46365	1	-	Covered
bin <auto[0],auto[0],auto[0]>	20286	1	-	Covered
Cross wr_en_rd_en_FULL	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	945	1	-	Covered
bin <auto[0],auto[1],auto[1]>	426	1	-	Covered
bin <auto[1],auto[0],auto[1]>	2251	1	-	Covered
bin <auto[0],auto[0],auto[1]>	886	1	-	Covered
bin <auto[1],auto[1],auto[0]>	20090	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8751	1	-	Covered
bin <auto[1],auto[0],auto[0]>	46365	1	-	Covered
bin <auto[0],auto[0],auto[0]>	20286	1	-	Covered
Cross wr_en_rd_en_EMPTY	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3358	1	-	Covered
bin <auto[0],auto[1],auto[1]>	1470	1	-	Covered
bin <auto[1],auto[0],auto[1]>	7592	1	-	Covered
bin <auto[0],auto[0],auto[1]>	3326	1	-	Covered
bin <auto[1],auto[1],auto[0]>	17677	1	-	Covered
bin <auto[0],auto[1],auto[0]>	7707	1	-	Covered
bin <auto[1],auto[0],auto[0]>	41024	1	-	Covered
bin <auto[0],auto[0],auto[0]>	17846	1	-	Covered
Cross wr_en_rd_en_ALMOSTFULL	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	1860	1	-	Covered
bin <auto[0],auto[1],auto[1]>	761	1	-	Covered
bin <auto[1],auto[0],auto[1]>	4325	1	-	Covered
bin <auto[0],auto[1],auto[1]>	/61	1	-	Covered
bin <auto[1],auto[0],auto[1]>	4325	1	-	Covered
bin <auto[0],auto[0],auto[1]>	1942	1	-	Covered
bin <auto[1],auto[1],auto[0]>	19175	1	-	Covered
bin <auto[0],auto[1],auto[0]>	8416	1	-	Covered
bin <auto[1],auto[0],auto[0]>	44291	1	-	Covered
bin <auto[0],auto[0],auto[0]>	19230	1	-	Covered
Cross wr_en_rd_en_ALMOSTEMPTY	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	3727	1	-	Covered
bin <auto[0],auto[1],auto[1]>	1607	1	-	Covered
bin <auto[1],auto[0],auto[1]>	8711	1	-	Covered
bin <auto[0],auto[0],auto[1]>	3816	1	-	Covered
bin <auto[1],auto[1],auto[0]>	17308	1	-	Covered
bin <auto[0],auto[1],auto[0]>	7570	1	-	Covered
bin <auto[1],auto[0],auto[0]>	39905	1	-	Covered
bin <auto[0],auto[0],auto[0]>	17356	1	-	Covered
Cross wr_en_rd_en_UNDERFLOW	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	285	1	-	Covered
bin <auto[0],auto[1],auto[1]>	115	1	-	Covered
bin <auto[1],auto[0],auto[1]>	657	1	-	Covered
bin <auto[0],auto[0],auto[1]>	285	1	-	Covered
bin <auto[1],auto[1],auto[0]>	20750	1	-	Covered
bin <auto[0],auto[1],auto[0]>	9062	1	-	Covered
bin <auto[1],auto[0],auto[0]>	47959	1	-	Covered
bin <auto[0],auto[0],auto[0]>	20887	1	-	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

Assertion coverage report :

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression
/top/a_reset	Immediate	SVA	on	0	1	-	-	-	-	0 off	assert (~fifoif.full&fifoif.empty&~fifoif...)	
/top/DUT/a_reset	Immediate	SVA	on	0	1	-	-	-	-	0 off	assert (wr_ptr==0&d_ptr==0&count==0)	
/top/DUT/assert_Almost_Empty_Condition	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Almost_Full_Condition	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.empty))	
/top/DUT/assert_Empty_Flag_Assertion	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Full_Flag_Assertion	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.empty))	
/top/DUT/assert_Overflow_Detection	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Pointer_threshold	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Pointer_Wraparound_1	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Pointer_Wraparound_2	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Pointer_Wraparound_3	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(#implicit-wire#0) (\$past(count)==0))	
/top/DUT/assert_Underflow_Detection	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_wr_rd_simul_empty	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_wr_rd_simul_full	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/DUT/assert_Write_Acknowledge	Concurrent	SVA	on	0	1	-	08	08	0 ns	0 off	assert(@(posedge fifoif.dk) disable iff (~fifoif.full))	
/top/TEST/#blk#182146786#19/immed_21	Immediate	SVA	on	0	1	-	-	-	-	0 off	assert (randomize(...))	

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmpl %	Cmpl graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/cover_wr_rd_simul_full	SVA	✓	Off	1761	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_wr_rd_simul_empty	SVA	✓	Off	2726	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Pointer_threshold	SVA	✓	Off	90030	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Pointer_Wraparound_3	SVA	✓	Off	1064	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Pointer_Wraparound_2	SVA	✓	Off	661	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Pointer_Wraparound_1	SVA	✓	Off	3292	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Almost_Empty_Condition	SVA	✓	Off	16079	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Almost_Full_Condition	SVA	✓	Off	8018	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Full_Flag_Assertion	SVA	✓	Off	9237	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Empty_Flag_Assertion	SVA	✓	Off	14192	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Underflow_Detection	SVA	✓	Off	1200	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Overflow_Detection	SVA	✓	Off	4030	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/cover_Write_Acknowledge	SVA	✓	Off	50760	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	
/top/DUT/i_cover_150	SVA	✓	Off	8613	1 Uni...	1	100%	██████	✓	0	0	0 ns	0	

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

Assertion Coverage:
 Assertions 14 14 0 100.00%
-----
Name File(Line) Failure Count Pass Count
-----
/top/DUT/assert_wr_rd_simul_full FIFO.sv(147) 0 1
/top/DUT/assert_wr_rd_simul_empty FIFO.sv(146) 0 1
/top/DUT/assert_Pointer_threshold FIFO.sv(145) 0 1
/top/DUT/assert_Pointer_Wraparound_3 FIFO.sv(144) 0 1
/top/DUT/assert_Pointer_Wraparound_2 FIFO.sv(143) 0 1
/top/DUT/assert_Pointer_Wraparound_1 FIFO.sv(142) 0 1
/top/DUT/assert_Almost_Empty_Condition FIFO.sv(141) 0 1
/top/DUT/assert_Almost_Full_Condition FIFO.sv(140) 0 1
/top/DUT/assert_Full_Flag_Assertion FIFO.sv(139) 0 1
/top/DUT/assert_Empty_Flag_Assertion FIFO.sv(138) 0 1
/top/DUT/assert_Underflow_Detection FIFO.sv(137) 0 1
/top/DUT/assert_Overflow_Detection FIFO.sv(136) 0 1
/top/DUT/assert_Write_Acknowledge FIFO.sv(135) 0 1
/top/DUT/a_reset FIFO.sv(82) 0 1
```

```
Directive Coverage:
 Directives 14 14 0 100.00%
-----
DIRECTIVE COVERAGE:
-----
Name Design Design Lang File(Line) Hits Status
Unit UnitType
-----
/top/DUT/cover_wr_rd_simul_full FIFO Verilog SVA FIFO.sv(163) 1761 Covered
/top/DUT/cover_wr_rd_simul_empty FIFO Verilog SVA FIFO.sv(162) 2726 Covered
/top/DUT/cover_Pointer_threshold FIFO Verilog SVA FIFO.sv(161) 90030 Covered
/top/DUT/cover_Pointer_Wraparound_3 FIFO Verilog SVA FIFO.sv(160) 1064 Covered
/top/DUT/cover_Pointer_Wraparound_2 FIFO Verilog SVA FIFO.sv(159) 661 Covered
/top/DUT/cover_Pointer_Wraparound_1 FIFO Verilog SVA FIFO.sv(158) 3292 Covered
/top/DUT/cover_Almost_Empty_Condition FIFO Verilog SVA FIFO.sv(157) 16079 Covered
/top/DUT/cover_Almost_Full_Condition FIFO Verilog SVA FIFO.sv(156) 8018 Covered
/top/DUT/cover_Full_Flag_Assertion FIFO Verilog SVA FIFO.sv(155) 9237 Covered
/top/DUT/cover_Empty_Flag_Assertion FIFO Verilog SVA FIFO.sv(154) 14192 Covered
/top/DUT/cover_Underflow_Detection FIFO Verilog SVA FIFO.sv(153) 1200 Covered
/top/DUT/cover_Overflow_Detection FIFO Verilog SVA FIFO.sv(152) 4030 Covered
/top/DUT/cover_Write_Acknowledge FIFO Verilog SVA FIFO.sv(151) 50760 Covered
/top/DUT/i_cover_150 FIFO Verilog SVA FIFO.sv(150) 8613 Covered
=====
```

```

=====
== Instance: /top/TEST
== Design Unit: work.FIFO_tb
=====

Assertion Coverage:
 Assertions 1 1 0 100.00%
-----
Name File(Line) Failure Pass
Count Count
-----
/top/TEST/#ublk#182146786#19/immed_21
    FIFO_tb.sv(21) 0 1
=====

=====
== Instance: /top
== Design Unit: work.top
=====

Assertion Coverage:
 Assertions 1 1 0 100.00%
-----
Name File(Line) Failure Pass
Count Count
-----
/top/a_reset FIFO_TOP.sv(21) 0 1
=====
```

DIRECTIVE COVERAGE:

Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status
/top/DUT/cover__wr_rd_simul_full	FIFO	Verilog	SVA	FIFO.sv(163)	1761	Covered
/top/DUT/cover__wr_rd_simul_empty	FIFO	Verilog	SVA	FIFO.sv(162)	2726	Covered
/top/DUT/cover__Pointer_threshold	FIFO	Verilog	SVA	FIFO.sv(161)	90030	Covered
/top/DUT/cover__Pointer_Wraparound_3	FIFO	Verilog	SVA	FIFO.sv(160)	1064	Covered
/top/DUT/cover__Pointer_Wraparound_2	FIFO	Verilog	SVA	FIFO.sv(159)	661	Covered
/top/DUT/cover__Pointer_Wraparound_1	FIFO	Verilog	SVA	FIFO.sv(158)	3292	Covered
/top/DUT/cover__Almost_Empty_Condition	FIFO	Verilog	SVA	FIFO.sv(157)	16079	Covered
/top/DUT/cover__Almost_Full_Condition	FIFO	Verilog	SVA	FIFO.sv(156)	8018	Covered
/top/DUT/cover__Full_Flag_Assertion	FIFO	Verilog	SVA	FIFO.sv(155)	9237	Covered
/top/DUT/cover__Empty_Flag_Assertion	FIFO	Verilog	SVA	FIFO.sv(154)	14192	Covered
/top/DUT/cover__Underflow_Detection	FIFO	Verilog	SVA	FIFO.sv(153)	1200	Covered
/top/DUT/cover__Overflow_Detection	FIFO	Verilog	SVA	FIFO.sv(152)	4930	Covered
/top/DUT/cover__Write_Acknowledge	FIFO	Verilog	SVA	FIFO.sv(151)	50760	Covered
/top/DUT/i_cover__150	FIFO	Verilog	SVA	FIFO.sv(150)	8613	Covered

TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 14

ASSERTION RESULTS:

Name	File(Line)	Failure Count	Pass Count
/top/a_reset	FIFO_TOP.sv(21)	0	1
/top/DUT/assert__wr_rd_simul_full	FIFO.sv(147)	0	1
/top/DUT/assert__wr_rd_simul_empty	FIFO.sv(146)	0	1
/top/DUT/assert__Pointer_threshold	FIFO.sv(145)	0	1
/top/DUT/assert__Pointer_Wraparound_3	FIFO.sv(144)	0	1
/top/DUT/assert__Pointer_Wraparound_2	FIFO.sv(143)	0	1
/top/DUT/assert__Pointer_Wraparound_1	FIFO.sv(142)	0	1
/top/DUT/assert__Almost_Empty_Condition	FIFO.sv(141)	0	1
/top/DUT/assert__Almost_Full_Condition	FIFO.sv(140)	0	1
/top/DUT/assert__Full_Flag_Assertion	FIFO.sv(139)	0	1
/top/DUT/assert__Empty_Flag_Assertion	FIFO.sv(138)	0	1
/top/DUT/assert__Underflow_Detection	FIFO.sv(137)	0	1
/top/DUT/assert__Overflow_Detection	FIFO.sv(136)	0	1
/top/DUT/assert__Write_Acknowledge	FIFO.sv(135)	0	1
/top/DUT/a_reset	FIFO.sv(82)	0	1
/top/TEST/#ublk#182146786#19/immed_21	FIFO_tb.sv(21)	0	1

Total Coverage By Instance (filtered view): 100.00%

Waveform snippets:

