Abdalrahman Taha Mahmed

SV Project FIFO

1 Bugs :

- wr_ack should be Low when reset is asserted .
- overflow should be Low when reset is asserted .
- underflow should be Low when reset is asserted .
- underflow output should be Sequential .
- Uncovered case when both wr_en and rd_en are high and FIFO if full , Reading process happens .
- Uncovered case when both wr_en and rd_en are high and FIFO if empty , Writing process happens .
- almostfull is high when there is two spots empty , while it should be when only one spot is empty .

2 Verification plan :

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	When the reset is asserted the FIFO outputs (data_out & other flags) should be low .	Directed at the start of the simulation ,then randomized with constraint that drive reset to be off most of the time .	-	Immediate assertion to check for the async reset functionality
FIFO_2	When the wr_en is asserted, the FIFO is not Full , Writing Operation takes place withl data_in input , wr_ack should be activatd .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and wr_ack .	Concurrent assertion to check the wr_ack Functionality .
FIFO_3	When the FIFO is full , the full flag should be activated .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and full .	Immediate assertion to check for the full Flag functionality
FIFO_4	When the FIFO is empty , the empty flag should be activated .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and empty .	Immediate assertion to check for the empty Flag functionality
FIFO_5	When the FIFO has only 1 space left , the almostfull flag should be activated .	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and almostfull .	Immediate assertion to check for the almostfull Flag functionality
FIFO_6	When the FIFO has only 1 space occupied , the almostempty flag should be activated .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and almostempty .	Immediate assertion to check for the almostempty Flag functionality
FIFO_7	When the FIFO is empty and the rd_en signal is high , underflow signal should be activated	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and underflow .	Immediate assertion to check for the underflow Flag functionality
FIFO_8	When the FIFO is full and the wr_en signal is high , overflow signal should be activated	Randomization under constrains on the wr_en signal to be on with value(WR_EN_ON_DIST) of the time .	Cross Coverage betweenf wr_en with rd_en and overflow .	Immediate assertion to check for the overflow Flag functionality
FIFO_9	When the FIFO is not empty & rd_en signal is high the data_out should take the value of the FIFO element with the rd_ptr address and the rd_ptr is incremented .	Randomization under constrains on the rd_en signal to be on with value(RD_EN_ON_DIST) of the time .	-	Concurrent assertion to check the rd_ptr_wraparound & rd_ptr_theshold , Also a Reference model is made to check data_out Functionality .

Code Design with assertion :

fifo.sv

```
1  ////////////////////////////////////////////////////////////////////////////////
2  // Author: Kareem Waseem
3  // Course: Digital Verification using SV & UVM
4  //
5  // Description: FIFO Design
6  //
7  ////////////////////////////////////////////////////////////////////////////////
8  module fifo(fifo_if.DUT fif);
9
10 localparam max_fifo_addr = $clog2(fif.FIFO_DEPTH);
11
12 reg [fif.FIFO_WIDTH-1:0] mem [fif.FIFO_DEPTH-1:0];
13
14 reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
15 reg [max_fifo_addr:0] count;
16
17 always @(posedge fif.clk or negedge fif.rst_n) begin
18    if (!fif.rst_n) begin
19       wr_ptr <= 0;
20       // BUG DETECTED : wr_ack should be Low when reset is asserted .
21       fif.wr_ack <= 0 ;
22       // BUG DETECTED : overflow should be Low when reset is asserted .
23       fif.overflow <= 0 ;
24
25    end
26    // Writing Block
27    else if (fif.wr_en && count < fif.FIFO_DEPTH ) begin
28       mem[wr_ptr] <= fif.data_in;
29       fif.wr_ack <= 1;
30       wr_ptr <= wr_ptr + 1;
31    end
32    else begin
33       fif.wr_ack <= 0;
34       if (fif.full & fif.wr_en)
35          fif.overflow <= 1;
36       else
37          fif.overflow <= 0;
38    end
39 end
40
```

```
41
42 // Reading Block
43 always @(posedge fif.clk or negedge fif.rst_n) begin
44    if (!fif.rst_n) begin
45       rd_ptr <= 0;
46       // BUG DETECTED : Underflow should be Low when reset is asserted .
47       fif.underflow <= 0 ;
48    end
49    else if (fif.rd_en && count != 0 ) begin
50       fif.data_out <= mem[rd_ptr];
51       rd_ptr <= rd_ptr + 1;
52    end
53    // BUG DETECTED : Underflow output should be Sequential .
54    else begin
55       if (fif.empty & fif.rd_en)
56          fif.underflow <= 1;
57       else
58          fif.underflow <= 0;
59 end
60 end
61
62 always @(posedge fif.clk or negedge fif.rst_n) begin
63    if (!fif.rst_n) begin
64       count <= 0;
65    end
66    else begin
67       if ( ({fif.wr_en, fif.rd_en} == 2'b10) && !fif.full)
68          count <= count + 1;
69       else if ( ({fif.wr_en, fif.rd_en} == 2'b01) && !fif.empty)
70          count <= count - 1;
71          // BUG DETECTED : Uncovered case when both wr_en and rd_en are high and FIFO if full , Reading process happens .
72          else if ( ({fif.wr_en, fif.rd_en} == 2'b11) && fif.full)
73          count <= count - 1;
74          //BUG DETECTED : Uncovered case when both wr_en and rd_en are high and FIFO if empty , Writing process happens .
75          else if ( ({fif.wr_en, fif.rd_en} == 2'b11) && fif.empty)
76          count <= count + 1;
77
78    end
79 end
80
81 assign fif.full = (count == fif.FIFO_DEPTH)? 1 : 0;
82 assign fif.empty = (count == 0)? 1 : 0;
83 //BUG DETECTED : almostfull is high when there is two spots empty , while it should be only one .
84 assign fif.almostfull = (count == fif.FIFO_DEPTH-1)? 1 : 0;
85 assign fif.almostempty = (count == 1)? 1 : 0;
86
```

```
89      property reset_check;
90          @(posedge fif.clk) (!fif.rst_n) |=> (wr_ptr == 0 && rd_ptr == 0 && count == 0);
91      endproperty
92      assert property (reset_check) else $error(" error in rst ");
93      cover property (reset_check);
94
95      property check1;
96         @(posedge fif.clk) disable iff (!fif.rst_n)
97              ( fif.wr_en && count < fif.FIFO_DEPTH ) |=>
98                  (fif.wr_ack == 1);
99      endproperty
100     assert property (check1) else $error("error in check1");
101     cover property (check1);
102
103     property check2;
104        @(posedge fif.clk) disable iff (!fif.rst_n)
105             ( fif.wr_en && fif.full ) |=>
106                 (fif.overflow == 1);
107     endproperty
108     assert property (check2) else $error("error in check2");
109     cover property (check2);
110
111     property check3;
112        @(posedge fif.clk) disable iff (!fif.rst_n)
113             ( fif.rd_en && fif.empty ) |=>
114                 (fif.underflow == 1);
115     endproperty
116     assert property (check3) else $error("error in check3");
117     cover property (check3);
118
119     property check4;
120        @(posedge fif.clk) disable iff (!fif.rst_n)
121             ( count == 0 ) |->
122                 (fif.empty);
123      endproperty
124     assert property (check4) else $error("error in check4");
125     cover property (check4);
126
127      property check5;
128        @(posedge fif.clk) disable iff (!fif.rst_n)
129           ( count == fif.FIFO_DEPTH ) |->
130               (fif.full);
131      endproperty
132     assert property (check5) else $error("error in check5");
133     cover property (check5);
134
```

```
135     property check6;
136        @(posedge fif.clk) disable iff (!fif.rst_n)
137           ( count == fif.FIFO_DEPTH - 1) |->
138                 (fif.almostfull);
139     endproperty
140     assert property (check6) else $error("error in check6");
141     cover property (check6);
142
143     property check7;
144        @(posedge fif.clk) disable iff (!fif.rst_n)
145             ( count == 1 ) |->
146                 (fif.almostempty);
147     endproperty
148     assert property (check7) else $error("error in check7");
149     cover property (check7);
150
151    property check8;
152        @(posedge fif.clk) disable iff (!fif.rst_n)
153              ( wr_ptr == 7 ) |-> (wr_ptr == 0) [=1];
154     endproperty
155     assert property (check8) else $error("error in check8");
156     cover property (check8);
157
158     property check9;
159        @(posedge fif.clk) disable iff (!fif.rst_n)
160             ( rd_ptr == 7 ) |-> (rd_ptr == 0) [=1];
161     endproperty
162     assert property (check9) else $error("error in check9");
163     cover property (check9);
164
165     property check10;
166        @(posedge fif.clk) disable iff (!fif.rst_n)
167              ( count == 8 ) |-> (count == 0 || count == 7) [=1] ;
168     endproperty
169     assert property (check10) else $error("error in check10");
170     cover property (check10);
171
172     property check11;
173         @(posedge fif.clk) disable iff (!fif.rst_n)
174              ( (fif.wr_en && count < fif.FIFO_DEPTH) )
175                |=> ( mem[$past(wr_ptr)] <= $past(fif.data_in) );
176     endproperty
177     assert property (check11) else $error("error in check11");
178     cover property (check11);
```

```
179
180     property check12 ;
181       @(posedge fif.clk) disable iff (!fif.rst_n)
182             (wr_ptr < fif.FIFO_DEPTH) ;
183      endproperty
184     assert property (check12 ) else $error("error in check12 ");
185     cover property (check12 );
186
187     property check13 ;
188      @(posedge fif.clk) disable iff (!fif.rst_n)
189          (rd_ptr < fif.FIFO_DEPTH) ;
190     endproperty
191     assert property (check13) else $error("error in check13");
192     cover property (check13);
193
194     property check14 ;
195        @(posedge fif.clk) disable iff (!fif.rst_n)
196          (count <= fif.FIFO_DEPTH) ;
197    endproperty
198     assert property (check14) else $error("error in check14");
199     cover property (check14);
200
201     property check15;
202       @(posedge fif.clk) disable iff (!fif.rst_n)
203            ( ({fif.wr_en, fif.rd_en} == 2'b10) && !fif.full) || ( ({fif.wr_en, fif.rd_en} == 2'b11) && fif.empty)
204                |=> (count == $past(count)+1);
205      endproperty
206     assert property (check15) else $error("error in check15");
207     cover property (check15);
208
209     property check16;
210       @(posedge fif.clk) disable iff (!fif.rst_n)
211            ( ({fif.wr_en, fif.rd_en} == 2'b01) && !fif.empty) || ( ({fif.wr_en, fif.rd_en} == 2'b11) && fif.full)
212                |=> (count == $past(count)-1);
213       endproperty
214     assert property (check16) else $error("error in check16");
215     cover property (check16);
216
217 endmodule
```

Code Interface :

fifo_if.sv

```
interface fifo_if (clk);
  parameter FIFO_WIDTH = 16;
  parameter FIFO_DEPTH = 8;

  input bit clk ;
  bit   rst_n, wr_en, rd_en;
  logic [FIFO_WIDTH-1:0] data_in;
  logic [FIFO_WIDTH-1:0] data_out;
  logic wr_ack, overflow;
  logic full, empty, almostfull, almostempty, underflow;

modport DUT (
input data_in,clk, rst_n, wr_en, rd_en,
output data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow);

modport TEST (
output data_in, rst_n, wr_en, rd_en,
input clk, data_out, wr_ack, overflow, full, empty, almostfull, almostempty, underflow);

modport MONITOR (
input data_in, rst_n, wr_en, rd_en, clk, data_out, wr_ack, overflow, full, empty, almostfull, almostempty, underflow);

endinterface
```

Code Shared_pkg :

shared_pkg.sv

```
package shared_pkg ;

bit test_finished ;

int error_count = 0 ;
int correct_count = 0 ;
endpackage
```

Code Transaction_pkg :

fifo_transaction_pkg.sv

```
1  package FIFO_transaction_pkg ;
2
3    class FIFO_transaction#(parameter FIFO_WIDTH = 16,
4                            parameter FIFO_DEPTH = 8) ;
5
6    rand bit rst_n, wr_en, rd_en;
7    rand logic [FIFO_WIDTH-1:0] data_in;
8
9    logic [FIFO_WIDTH-1:0] data_out;
10   logic wr_ack, overflow;
11   logic full, empty, almostfull, almostempty, underflow;
12
13   int RD_EN_ON_DIST ;
14   int WR_EN_ON_DIST ;
15
16   function new(int rd_val = 30, int wr_val = 70);
17     RD_EN_ON_DIST = rd_val;
18     WR_EN_ON_DIST = wr_val;
19   endfunction
20
21   constraint rst_n_c {
22     rst_n dist {0:=10 , 1:=90};
23     }
24
25   constraint wr_en_c {
26     wr_en dist {0:=(100-WR_EN_ON_DIST) , 1:=WR_EN_ON_DIST};
27     }
28
29   constraint rd_en_c {
30     rd_en dist {0:=(100-RD_EN_ON_DIST) , 1:=RD_EN_ON_DIST};
31     }
32
33   endclass //FIFO_transaction
34
35   endpackage
```

Code Scoreboard_pkg :

fifo_scoreboard_pkg.sv

```
1   package FIFO_scoreboard_pkg ;
2   import FIFO_transaction_pkg ::*;
3   import shared_pkg ::*;
4
5   parameter FIFO_WIDTH = 16;
6   parameter FIFO_DEPTH = 8;
7
8     localparam max_fifo_addr = $clog2(FIFO_DEPTH);
9     logic [FIFO_WIDTH-1:0] data_out_ref;
10    logic [FIFO_WIDTH-1:0] Queue [$] ;
11    logic full_ref , empty_ref ;
12    logic [max_fifo_addr:0] count;
13
14  class FIFO_scoreboard ;
15
16    task check_data (input FIFO_transaction F_chk_txn);
17      Reference_model (F_chk_txn);
18      if ((data_out_ref !== F_chk_txn.data_out)) begin
19            error_count ++ ;
20            $display("   [ERROR] Mismatch at time %0t:", $time);
21            $display("  data_out     => Expected: %0h, Actual: %0h", data_out_ref,    F_chk_txn.data_out);
22      end
23      else begin
24        correct_count ++;
25      end
26
27    endtask
```

```
29    function void Reference_model (input FIFO_transaction F_ref_txn);
30
31        if (!F_ref_txn.rst_n) begin
32          Queue <= {};
33          count <= 0;
34        end
35        else begin
36
37          if (F_ref_txn.wr_en && count <  FIFO_DEPTH) begin
38            Queue.push_back(F_ref_txn.data_in);
39            count <= Queue.size();
40          end
41
42          if (F_ref_txn.rd_en && count != 0) begin
43            data_out_ref <= Queue.pop_front();
44            count <= Queue.size();
45          end
46
47        end
48
49        full_ref = (count == FIFO_DEPTH);
50        empty_ref = (count == 0);
51      endfunction
52
53    endclass
54  endpackage
```

Code Coverage_pkg :

fifo_coverage_pkg.sv

```
1  package FIFO_coverage_pkg ;
2  import FIFO_transaction_pkg ::*;
3
4    class FIFO_coverage ;
5      FIFO_transaction  F_cvg_txn ;
6
7    covergroup fifo_cvr_gp ;
8
9                cover_wr_en : coverpoint F_cvg_txn.wr_en ;
10               cover_rd_en : coverpoint F_cvg_txn.rd_en ;
11               cover_wr_ack : coverpoint F_cvg_txn.wr_ack ;
12               cover_overflow : coverpoint F_cvg_txn.overflow ;
13               cover_underflow : coverpoint F_cvg_txn.underflow ;
14               cover_full : coverpoint F_cvg_txn.full ;
15               cover_empty : coverpoint F_cvg_txn.empty ;
16               cover_almostfull : coverpoint F_cvg_txn.almostfull ;
17               cover_almostempty : coverpoint F_cvg_txn.almostempty ;
18
19               label_cross1 : cross cover_wr_en , cover_rd_en , cover_wr_ack ;
20               label_cross2 : cross cover_wr_en , cover_rd_en , cover_overflow ;
21               label_cross3 : cross cover_wr_en , cover_rd_en , cover_underflow ;
22               label_cross4 : cross cover_wr_en , cover_rd_en , cover_full ;
23               label_cross5 : cross cover_wr_en , cover_rd_en , cover_empty ;
24               label_cross6 : cross cover_wr_en , cover_rd_en , cover_almostfull ;
25               label_cross7 : cross cover_wr_en , cover_rd_en , cover_almostempty ;
26
27   endgroup
28
29   function new();
30     fifo_cvr_gp = new();
31   endfunction
32
33   function void sample_data (input FIFO_transaction F_txn);
34     F_cvg_txn = F_txn ;
35     fifo_cvr_gp.sample();
36   endfunction
37
38   endclass
39 endpackage
```

Code Monitor :

fifo_monitor.sv

```
import FIFO_transaction_pkg ::*;
import FIFO_scoreboard_pkg ::*;
import FIFO_coverage_pkg ::*;
import shared_pkg ::*;

module fifo_monitor (fifo_if.MONITOR fif);

FIFO_transaction FIFO_mon_txn  =new();
FIFO_scoreboard FIFO_mon_sb =new();
FIFO_coverage FIFO_mon_cov = new();

  initial begin
    forever begin
      @(negedge fif.clk);
      assert(FIFO_mon_txn.randomize());
      FIFO_mon_txn.rst_n       = fif.rst_n;
      FIFO_mon_txn.wr_en       = fif.wr_en;
      FIFO_mon_txn.rd_en       = fif.rd_en;
      FIFO_mon_txn.data_in     = fif.data_in;
      FIFO_mon_txn.data_out    = fif.data_out;
      FIFO_mon_txn.wr_ack      = fif.wr_ack;
      FIFO_mon_txn.overflow    = fif.overflow;
      FIFO_mon_txn.full        = fif.full;
      FIFO_mon_txn.empty       = fif.empty;
      FIFO_mon_txn.almostfull  = fif.almostfull;
      FIFO_mon_txn.almostempty = fif.almostempty;
      FIFO_mon_txn.underflow   = fif.underflow;
      fork
        begin
          FIFO_mon_cov.sample_data(FIFO_mon_txn);
        end

        begin
          FIFO_mon_sb.check_data(FIFO_mon_txn);
        end
      join
      if(test_finished) begin
        $display("Simulation Stopped : Error count = %0d , Correct count = %0d",error_count,correct_count);
        $stop ;
      end

    end
  end


endmodule
```

Code Testbench :

fifo_tb.sv

```
1   import FIFO_transaction_pkg ::*;
2   import shared_pkg::*;
3
4   module fifo_tb (fifo_if.TEST fif);
5
6     FIFO_transaction test_txn = new();
7
8     initial begin
9       fif.rst_n = 0 ;
10      repeat(2) @(negedge fif.clk);
11      repeat(2000) begin
12      assert(test_txn.randomize());
13      fif.rst_n = test_txn.rst_n ;
14      fif.wr_en = test_txn.wr_en ;
15      fif.rd_en = test_txn.rd_en ;
16      fif.data_in = test_txn.data_in ;
17      repeat(2) @(negedge fif.clk);
18      end
19      test_finished = 1 ;
20    end
21  endmodule
```

Code Top :

fifo_top.sv

```
1   module fifo_top ;
2     bit clk ;
3
4     initial begin
5       clk = 0 ;
6       forever begin
7         #1 clk = ~clk ;
8       end
9     end
10
11    fifo_if fif (clk) ;
12    fifo DUT (fif);
13    fifo_tb TEST (fif);
14    fifo_monitor MON (fif);
15
16  endmodule
```

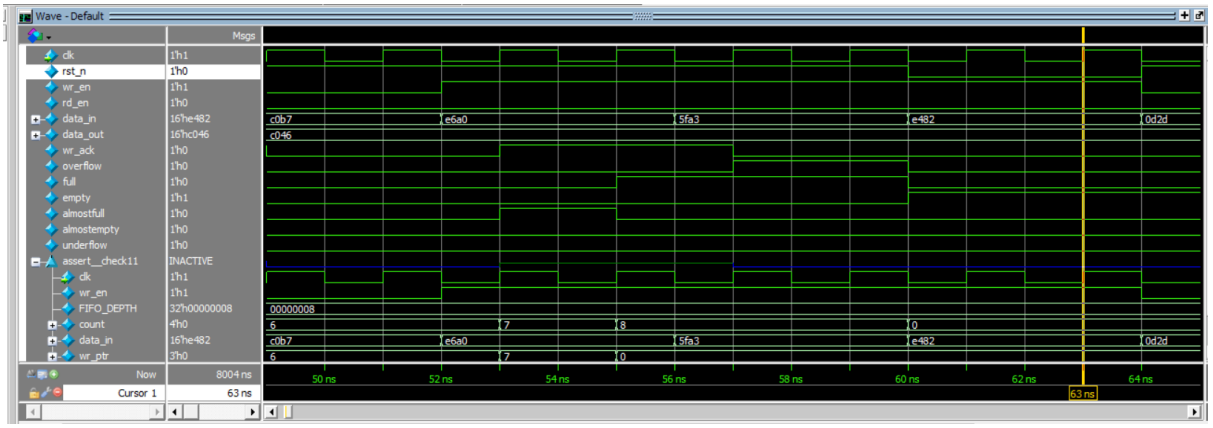
Do File :

fifo.do

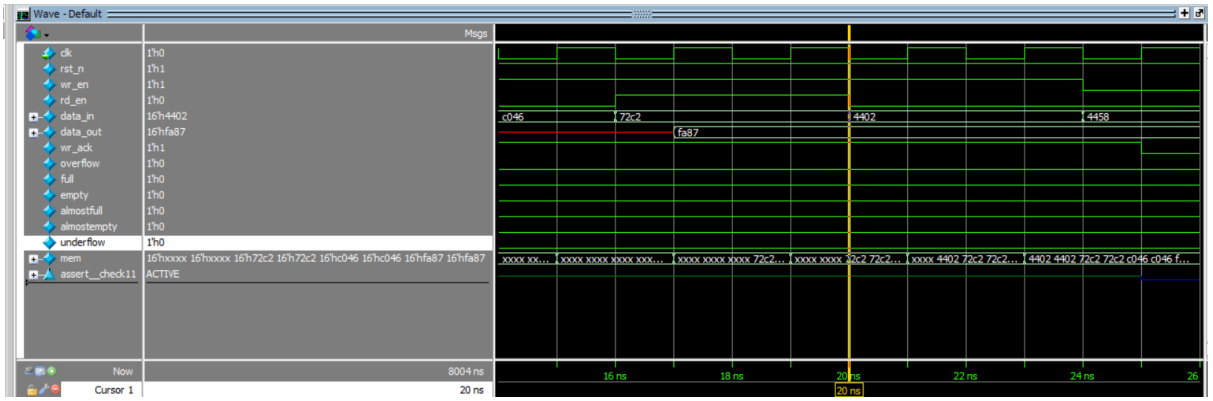
```
1   vlib work
2   vlog -f fifo_files.list +cover -covercells
3   vsim -voptargs=+acc work.fifo_top -cover
4   add wave /fifo_top/fif/*
5   coverage save fifo_tb.ucdb -onexit
6   run -all
```

Questasim Snippets :

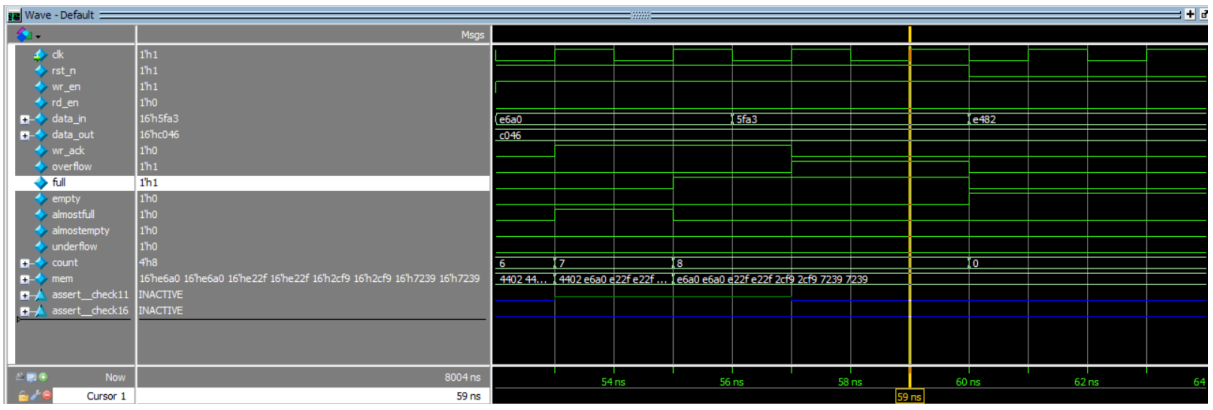
FIFO_1 : (When the reset is asserted the FIFO outputs should be low .



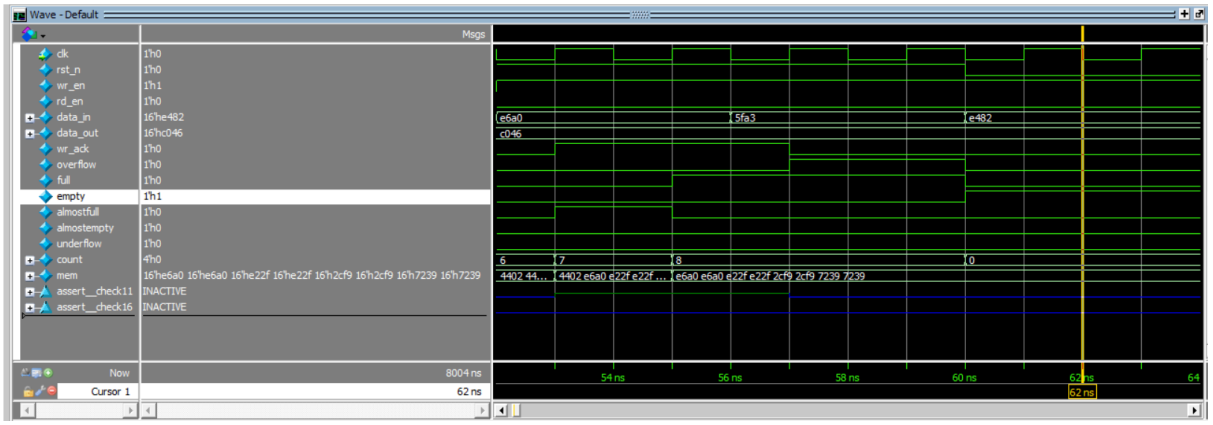
FIFO_2 : (When the wr_en is asserted, the FIFO is not Full , Writing Operation takes place with data_in input , wr_ack should be activated



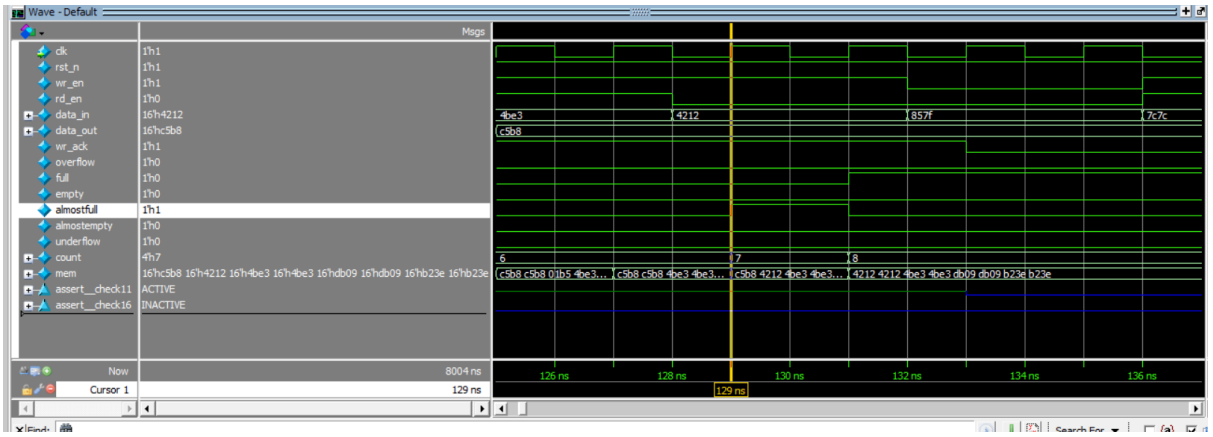
FIFO_3: (When the FIFO is full , the full flag should be activated



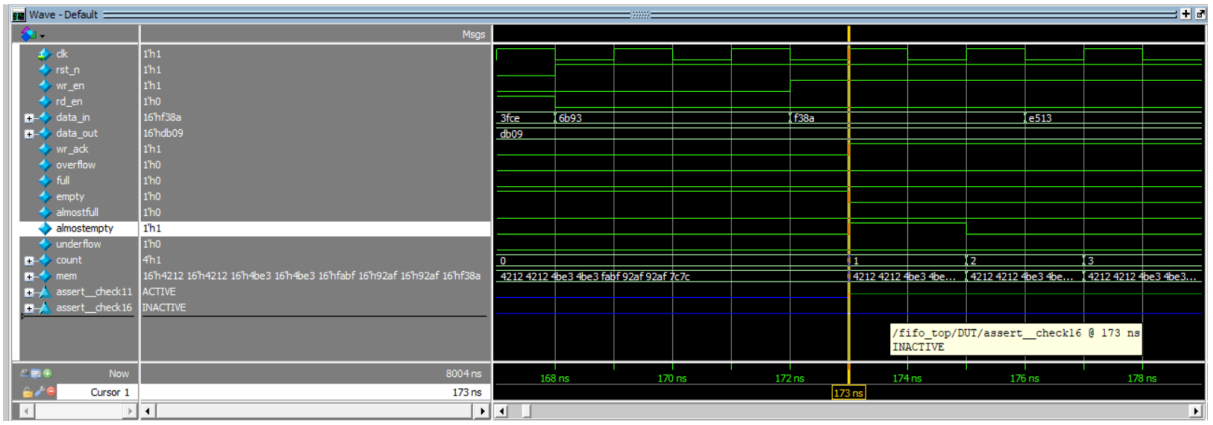
FIFO_4 : (When the FIFO is empty , the empty flag should be activated



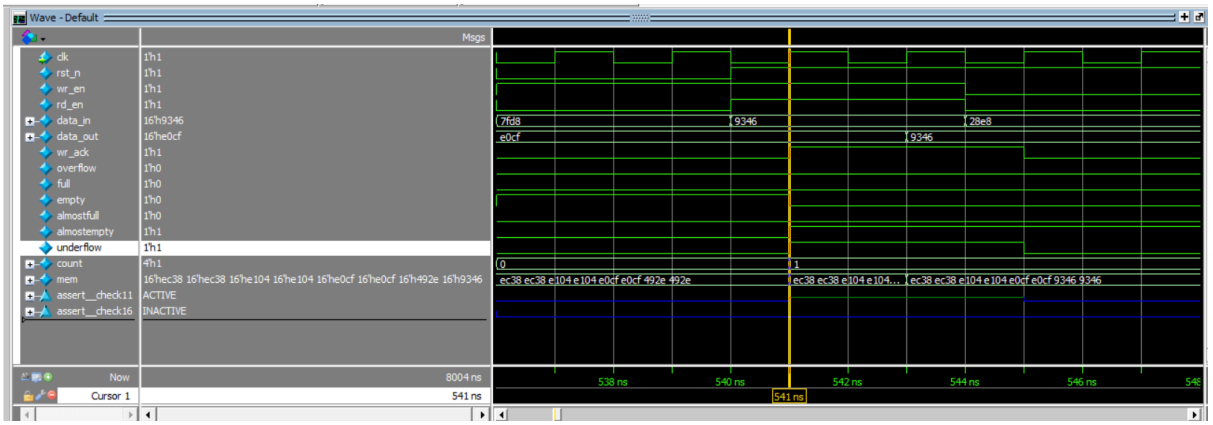
FIFO_5 : (When the FIFO has only 1 space left , the almostfull flag should be activated



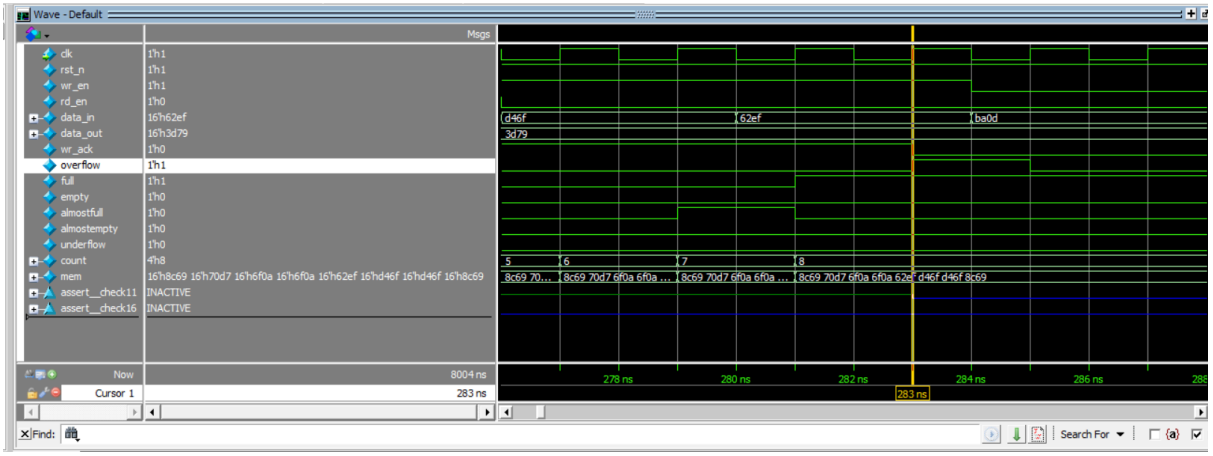
FIFO_6 : (When the FIFO has only 1 space occupied , the almostempty flag should be activated



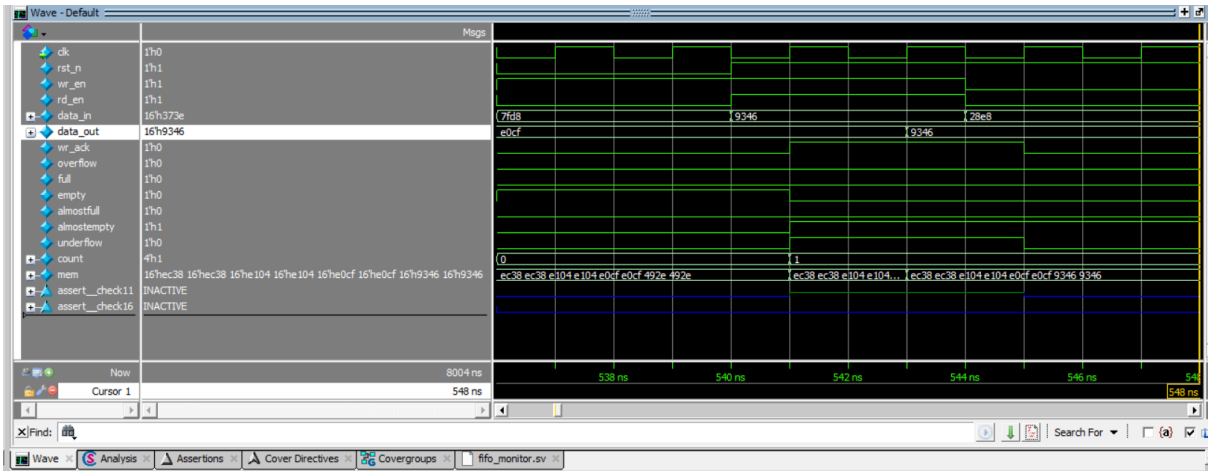
FIFO_7 : (When the FIFO is empty and the rd_en signal is high , underflow signal should be activated



FIFO_8 : (When the FIFO is full and the wr_en signal is high , overflow signal should be activated



FIFO_9 : (When the FIFO is not empty & rd_en signal is high the data_out should take the value of the FIFO element with the rd_ptr address and the rd_ptr is incremented



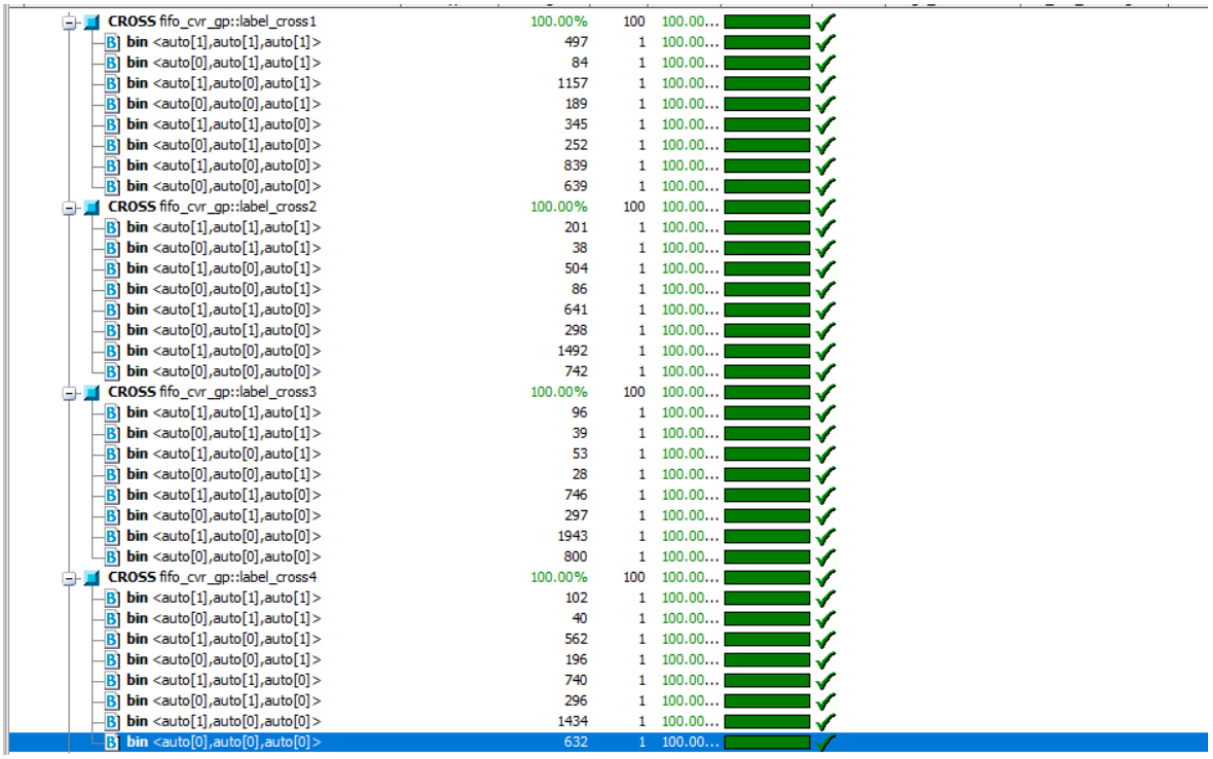
Assertion :

Assertions

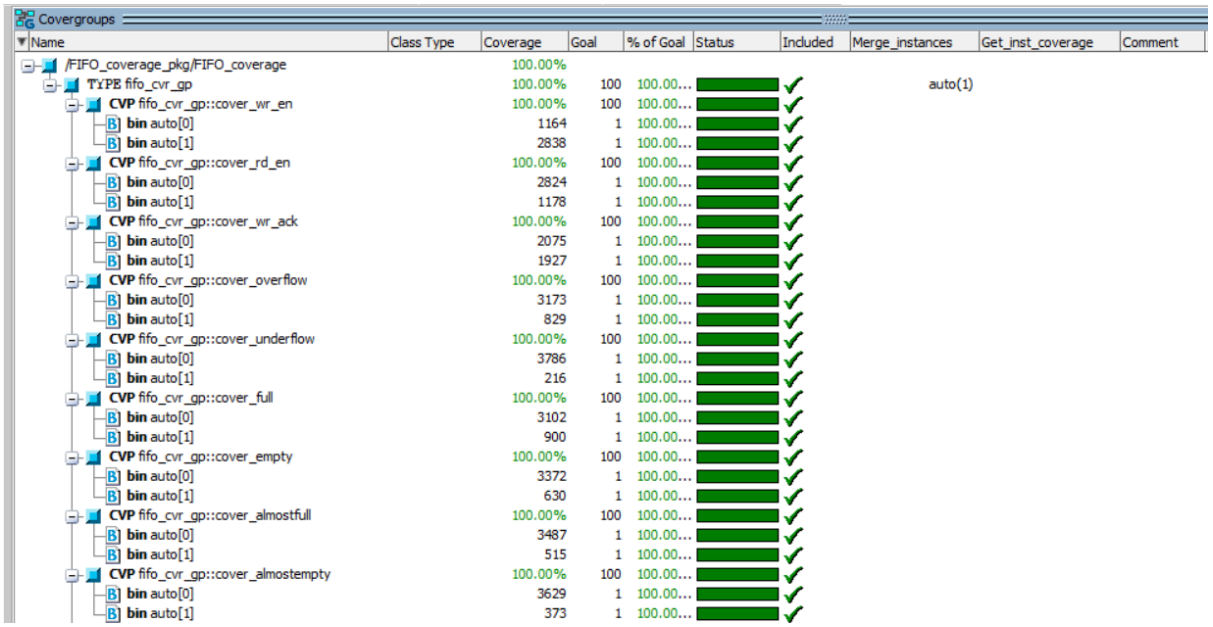
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Incluc
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) (~fif.rst...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disable i...	✓
/fifo_top/TEST/#u...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (test_txn.randomize())	✓
/fifo_top/MON/#ub...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (FIFO_mon_txn.randomize())	✓

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/fifo_top/DUT/cove...	SVA	✓	Off	320	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	1246	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	3584	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	1816	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	602	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	103	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	185	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	362	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	491	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	848	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	398	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	114	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	597	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	1816	1	Unli...	1	100%		✓	0	0	0 ns	0
/fifo_top/DUT/cove...	SVA	✓	Off	418	1	Unli...	1	100%		✓	0	0	0 ns	0

Covergroup :

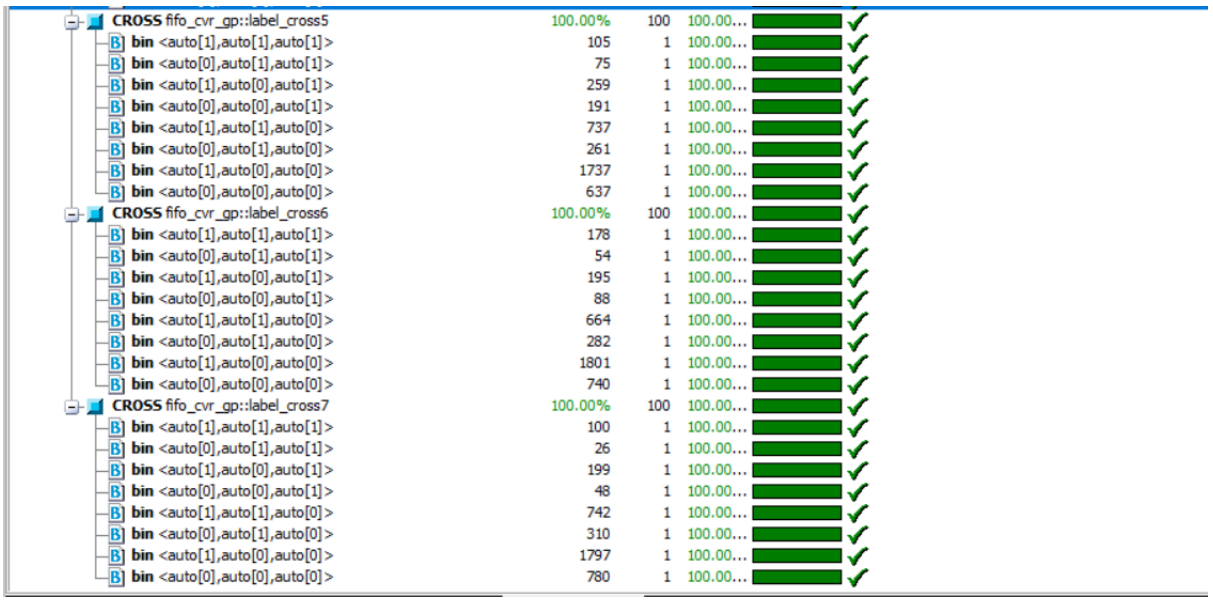


CROSS fifo_cvr_gp::label_cross1	100.00%	100	100.00...	
bin <auto[1],auto[1],auto[1]>	497	1	100.00...	
bin <auto[0],auto[1],auto[1]>	84	1	100.00...	
bin <auto[1],auto[0],auto[1]>	1157	1	100.00...	
bin <auto[0],auto[0],auto[1]>	189	1	100.00...	
bin <auto[1],auto[1],auto[0]>	345	1	100.00...	
bin <auto[0],auto[1],auto[0]>	252	1	100.00...	
bin <auto[1],auto[0],auto[0]>	839	1	100.00...	
bin <auto[0],auto[0],auto[0]>	639	1	100.00...	
CROSS fifo_cvr_gp::label_cross2	100.00%	100	100.00...	
bin <auto[1],auto[1],auto[1]>	201	1	100.00...	
bin <auto[0],auto[1],auto[1]>	38	1	100.00...	
bin <auto[1],auto[0],auto[1]>	504	1	100.00...	
bin <auto[0],auto[0],auto[1]>	86	1	100.00...	
bin <auto[1],auto[1],auto[0]>	641	1	100.00...	
bin <auto[0],auto[1],auto[0]>	298	1	100.00...	
bin <auto[1],auto[0],auto[0]>	1492	1	100.00...	
bin <auto[0],auto[0],auto[0]>	742	1	100.00...	
CROSS fifo_cvr_gp::label_cross3	100.00%	100	100.00...	
bin <auto[1],auto[1],auto[1]>	96	1	100.00...	
bin <auto[0],auto[1],auto[1]>	39	1	100.00...	
bin <auto[1],auto[0],auto[1]>	53	1	100.00...	
bin <auto[0],auto[0],auto[1]>	28	1	100.00...	
bin <auto[1],auto[1],auto[0]>	746	1	100.00...	
bin <auto[0],auto[1],auto[0]>	297	1	100.00...	
bin <auto[1],auto[0],auto[0]>	1943	1	100.00...	
bin <auto[0],auto[0],auto[0]>	800	1	100.00...	
CROSS fifo_cvr_gp::label_cross4	100.00%	100	100.00...	
bin <auto[1],auto[1],auto[1]>	102	1	100.00...	
bin <auto[0],auto[1],auto[1]>	40	1	100.00...	
bin <auto[1],auto[0],auto[1]>	562	1	100.00...	
bin <auto[0],auto[0],auto[1]>	196	1	100.00...	
bin <auto[1],auto[1],auto[0]>	740	1	100.00...	
bin <auto[0],auto[1],auto[0]>	296	1	100.00...	
bin <auto[1],auto[0],auto[0]>	1434	1	100.00...	
bin <auto[0],auto[0],auto[0]>	632	1	100.00...	



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_coverage_pkg/FIFO_coverage		100.00%							
TYPE fifo_cvr_gp		100.00%	100	100.00...			auto(1)		
CVP fifo_cvr_gp::cover_wr_en		100.00%	100	100.00...					
bin auto[0]		1164	1	100.00...					
bin auto[1]		2838	1	100.00...					
CVP fifo_cvr_gp::cover_rd_en		100.00%	100	100.00...					
bin auto[0]		2824	1	100.00...					
bin auto[1]		1178	1	100.00...					
CVP fifo_cvr_gp::cover_wr_ack		100.00%	100	100.00...					
bin auto[0]		2075	1	100.00...					
bin auto[1]		1927	1	100.00...					
CVP fifo_cvr_gp::cover_overflow		100.00%	100	100.00...					
bin auto[0]		3173	1	100.00...					
bin auto[1]		829	1	100.00...					
CVP fifo_cvr_gp::cover_underflow		100.00%	100	100.00...					
bin auto[0]		3786	1	100.00...					
bin auto[1]		216	1	100.00...					
CVP fifo_cvr_gp::cover_full		100.00%	100	100.00...					
bin auto[0]		3102	1	100.00...					
bin auto[1]		900	1	100.00...					
CVP fifo_cvr_gp::cover_empty		100.00%	100	100.00...					
bin auto[0]		3372	1	100.00...					
bin auto[1]		630	1	100.00...					
CVP fifo_cvr_gp::cover_almostfull		100.00%	100	100.00...					
bin auto[0]		3487	1	100.00...					
bin auto[1]		515	1	100.00...					
CVP fifo_cvr_gp::cover_almostempty		100.00%	100	100.00...					
bin auto[0]		3629	1	100.00...					
bin auto[1]		373	1	100.00...					



CROSS fifo_cvr_gp::label_cross5	100.00%	100	100.00...
bin <auto[1],auto[1],auto[1]>	105	1	100.00...
bin <auto[0],auto[1],auto[1]>	75	1	100.00...
bin <auto[1],auto[0],auto[1]>	259	1	100.00...
bin <auto[0],auto[0],auto[1]>	191	1	100.00...
bin <auto[1],auto[1],auto[0]>	737	1	100.00...
bin <auto[0],auto[1],auto[0]>	261	1	100.00...
bin <auto[1],auto[0],auto[0]>	1737	1	100.00...
bin <auto[0],auto[0],auto[0]>	637	1	100.00...
CROSS fifo_cvr_gp::label_cross6	100.00%	100	100.00...
bin <auto[1],auto[1],auto[1]>	178	1	100.00...
bin <auto[0],auto[1],auto[1]>	54	1	100.00...
bin <auto[1],auto[0],auto[1]>	195	1	100.00...
bin <auto[0],auto[0],auto[1]>	88	1	100.00...
bin <auto[1],auto[1],auto[0]>	664	1	100.00...
bin <auto[0],auto[1],auto[0]>	282	1	100.00...
bin <auto[1],auto[0],auto[0]>	1801	1	100.00...
bin <auto[0],auto[0],auto[0]>	740	1	100.00...
CROSS fifo_cvr_gp::label_cross7	100.00%	100	100.00...
bin <auto[1],auto[1],auto[1]>	100	1	100.00...
bin <auto[0],auto[1],auto[1]>	26	1	100.00...
bin <auto[1],auto[0],auto[1]>	199	1	100.00...
bin <auto[0],auto[0],auto[1]>	48	1	100.00...
bin <auto[1],auto[1],auto[0]>	742	1	100.00...
bin <auto[0],auto[1],auto[0]>	310	1	100.00...
bin <auto[1],auto[0],auto[0]>	1797	1	100.00...
bin <auto[0],auto[0],auto[0]>	780	1	100.00...

Code Coverage :

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        25        25         0   100.00%
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      27        27         0   100.00%
```

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