FIFO_Project

Submitted to: ENG. Kareem Waseem

Design code:

```
module FIFO(FIFO_interface.DUT FIFO_IC);

localparam max_fifo_addr = $clog2(FIFO_IC.FIFO_DEPTH);

reg [FIFO_IC.FIFO_WIDTH-1:0] mem [FIFO_IC.FIFO_DEPTH-1:0];
reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
reg [max_fifo_addr:0] count;

always @(posedge FIFO_IC.clk or negedge FIFO_IC.rst_n) begin
    if (!FIFO_IC.rst_n) begin
        wr_ptr <= 0;
        FIFO_IC.overflow<=0; // was added
    end
    else if (FIFO_IC.wr_en && count < FIFO_IC.FIFO_DEPTH) begin
        mem[wr_ptr] <= FIFO_IC.data_in;
        FIFO_IC.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        FIFO_IC.wr_ack <= 0;
        if (FIFO_IC.full & FIFO_IC.wr_en)
            FIFO_IC.overflow <= 1;
        else
            FIFO_IC.overflow <= 0;
    end
end

always @(posedge FIFO_IC.clk or negedge FIFO_IC.rst_n) begin
    if (!FIFO_IC.rst_n) begin
        rd_ptr <= 0;
        FIFO_IC.underflow<=0; //was added
    end
    else if (FIFO_IC.rd_en && count != 0) begin
        FIFO_IC.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin
        if (FIFO_IC.empty && FIFO_IC.rd_en)begin
            FIFO_IC.underflow <= 1;
        end
        else begin
            FIFO_IC.underflow <= 0;   // underflow was modified to be sequential instead of combinational
        end
    end
end
```

```
always @(posedge FIFO_IC.clk or negedge FIFO_IC.rst_n) begin
    if (!FIFO_IC.rst_n) begin
        count <= 0;
    end
    else begin
        if  (({FIFO_IC.wr_en, FIFO_IC.rd_en} == 2'b10) && !FIFO_IC.full)
            count <= count + 1;
        else if (({FIFO_IC.wr_en, FIFO_IC.rd_en} == 2'b01) && !FIFO_IC.empty)
            count <= count - 1;
        else if ({FIFO_IC.wr_en, FIFO_IC.rd_en} == 2'b11) begin // if condition was added to handle if the wr_en , rd_en are asserted at the same time
            if (FIFO_IC.full) begin
                count <= count - 1;
            end
            else if (FIFO_IC.empty) begin
                count <= count + 1;
            end
        end
    end
end

assign FIFO_IC.full = (count == FIFO_IC.FIFO_DEPTH)? 1 : 0;
assign FIFO_IC.empty = (count == 0)? 1 : 0;
assign FIFO_IC.almostfull = (count == FIFO_IC.FIFO_DEPTH-1)? 1 : 0;  // corrected to be -1  instead of -2
assign FIFO_IC.almostempty = (count == 1)? 1 : 0;
```

```
  property p_count_wr;
    @(posedge FIFO_IC.clk) disable iff (!FIFO_IC.rst_n)
    (FIFO_IC.wr_en && !FIFO_IC.full && !FIFO_IC.rd_en) |=> (count == $past(count) + 1'b1);
  endproperty

  property p_count_rd;
    @(posedge FIFO_IC.clk) disable iff (!FIFO_IC.rst_n)
    (FIFO_IC.rd_en && !FIFO_IC.empty && !FIFO_IC.wr_en) |=> (count == $past(count) - 1'b1);
  endproperty

  property p_w_ack;
    @(posedge FIFO_IC.clk) disable iff (!FIFO_IC.rst_n)
    (FIFO_IC.wr_en && !FIFO_IC.full) |=> (FIFO_IC.wr_ack == 1);
  endproperty

  property p_full;
    @(posedge FIFO_IC.clk)
    (count == FIFO_IC.FIFO_DEPTH) |-> (FIFO_IC.full == 1);
  endproperty

  property p_empty;
    @(posedge FIFO_IC.clk)
    (count == 0) |-> (FIFO_IC.empty == 1);
  endproperty

  property p_almostfull;
    @(posedge FIFO_IC.clk)
    (count == FIFO_IC.FIFO_DEPTH - 1) |-> (FIFO_IC.almostfull == 1);
  endproperty

  property p_almostempty;
    @(posedge FIFO_IC.clk)
    (count == 1) |-> (FIFO_IC.almostempty == 1);
  endproperty

  property p_underflow;
    @(posedge FIFO_IC.clk) disable iff (!FIFO_IC.rst_n)
    (FIFO_IC.empty && FIFO_IC.rd_en) |=> (FIFO_IC.underflow == 1);
  endproperty

  property p_overflow;
    @(posedge FIFO_IC.clk) disable iff (!FIFO_IC.rst_n)
    (FIFO_IC.full && FIFO_IC.wr_en) |=> (FIFO_IC.overflow == 1);
  endproperty

  property p_count_check;
    @(posedge FIFO_IC.clk)
    (count <= FIFO_IC.FIFO_DEPTH);
  endproperty

  property p_wr_ptr_check;
    @(posedge FIFO_IC.clk)
    (wr_ptr < FIFO_IC.FIFO_DEPTH);
  endproperty
```

```
139     property p_rd_ptr_check;
140       @(posedge FIFO_IC.clk)
141       (rd_ptr < FIFO_IC.FIFO_DEPTH);
142     endproperty
143
144
145     A_count_wr: assert property (p_count_wr);
146     C_count_wr: cover property (p_count_wr);
147
148     // Assertions and Cover for Read Count
149
150     A_count_rd: assert property (p_count_rd);
151     C_count_rd: cover property (p_count_rd);
152
153     // Assertions and Cover for Write Acknowledge
154
155     A_w_ack: assert property (p_w_ack);
156     C_w_ack: cover property (p_w_ack);
157
158     // Assertions and Cover for Full Condition
159
160     A_full: assert property (p_full);
161     C_full: cover property (p_full);
162
163     // Assertions and Cover for Empty Condition
164
165     A_empty: assert property (p_empty);
166     C_empty: cover property (p_empty);
167
168     // Assertions and Cover for Almost Full Condition
169
170     A_almostfull: assert property (p_almostfull);
171     C_almostfull: cover property (p_almostfull);
172
173     // Assertions and Cover for Almost Empty Condition
174
175     A_almostempty: assert property (p_almostempty);
176     C_almostempty: cover property (p_almostempty);
177
178     // Assertions and Cover for Underflow Condition
179
180     A_underflow: assert property (p_underflow);
181     C_underflow: cover property (p_underflow);
182
183     // Assertions and Cover for Overflow Condition
184
185     A_overflow: assert property (p_overflow);
186     C_overflow: cover property (p_overflow);
187
188     // Assertions and Cover for Counter Check
189
190     A_count_check: assert property (p_count_check);
191     C_count_check: cover property (p_count_check);
```

```
190     A_count_check: assert property (p_count_check);
191     C_count_check: cover property (p_count_check);
192
193     // Assertions and Cover for Write Pointer Check
194
195     A_wr_ptr_check: assert property (p_wr_ptr_check);
196     C_wr_ptr_check: cover property (p_wr_ptr_check);
197
198     // Assertions and Cover for Read Pointer Check
199
200     A_rd_ptr_check: assert property (p_rd_ptr_check);
201     C_rd_ptr_check: cover property (p_rd_ptr_check);
202
203     always_comb begin
204         if (!FIFO_IC.rst_n) begin
205
206             p_wr_ptr_reset: assert final (wr_ptr == 0);
207             c_wr_ptr_reset: cover (wr_ptr == 0);
208
209             p_rd_ptr_reset: assert final (rd_ptr == 0);
210             c_rd_ptr_reset: cover (rd_ptr == 0);
211
212             p_count_reset: assert final (count == 0);
213             c_count_reset: cover (count == 0);
214
215         end
216     end
217
218     endmodule
219
```

Verification Plan:

	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	FIFO_1	When the reset is asserted, the output data_out value should be low	Directed at the start of the sim, then randomized with constraint that drive the reset to be off most of the simulation time.	-	Immediate assertion to check the async reset functionality
3	FIFO_2	When wr_en is asserted and full is low wr_ack should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and wr_ack	Concurrent assertion to check for it
4	FIFO_3	When wr_en is asserted and full is high then overflow should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and overflow	Concurrent assertion to check for it
5	FIFO_4	When count is equal to FIFO depth, full value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and full	Concurrent assertion to check for it
6	FIFO_5	When count is equal to zero, empty value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and empty	Concurrent assertion to check for it
7	FIFO_6	When count is equal to FIFO depth -1, almostfull value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and almostfull	Concurrent assertion to check for it
8	FIFO_7	When count is equal to 1, almostempty value should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and almostempty	Concurrent assertion to check for it

9	FIFO_8	When rd_en is asserted and empty is high then underflow should be high	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time	Cover all combination values of rd_en , wr_en and underflow	Concurrent assertion to check for it
10	FIFO_9	According to the given inputs the data_out should be correct	Randomization under constraints on the wr_en signal to be on 70% of the time and on the rd_en signal to be off 70% of the time		Check data function with reference model function to check the data_out
11	FIFO_10				

Top file:

```
module top ();

  bit clk;

  initial begin
    clk = 0;
    forever
      #2 clk = ~clk;

  end

  FIFO_interface FIFO_IC (clk);
  FIFO_tb tb (FIFO_IC);
  FIFO DUT (FIFO_IC);
  monitor MONITOR (FIFO_IC);

endmodule
```

Interface code:

```
interface FIFO_interface (clk);

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

    input clk;


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


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

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

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

endinterface
```

Test bench code:

```
module FIFO_tb(FIFO_interface.TEST FIFO_IC);

    import shared_pkg::*;
    import trans_pkg::*;

    FIFO_transaction trans = new;

initial begin
    test_finished = 0;
    FIFO_IC.rst_n = 0;
    @(negedge FIFO_IC.clk);
    FIFO_IC.rst_n = 1;

    @(negedge FIFO_IC.clk);

    for (int i = 0; i < 10000; i++) begin

        assert(trans.randomize());

        FIFO_IC.rst_n = trans.rst_n;
        FIFO_IC.wr_en = trans.wr_en;
        FIFO_IC.rd_en = trans.rd_en;
        FIFO_IC.data_in = trans.data_in;


        @(negedge FIFO_IC.clk);
    end

    test_finished = 1;
end


endmodule
```

Monitor code:

```
module monitor (FIFO_interface.MONITOR FIFO_IC);

    import coverage_pkg::*;
    import scoreboard_pkg::*;
    import trans_pkg::*;
    import shared_pkg::*;

    FIFO_transaction transaction=new;
    FIFO_scoreboard score=new;
    FIFO_coverage coverage=new;

    initial begin

        forever begin

            @(negedge  FIFO_IC.clk);

            transaction.rd_en = FIFO_IC.rd_en;
            transaction.wr_en = FIFO_IC.wr_en;
            transaction.wr_ack = FIFO_IC.wr_ack;
            transaction.full = FIFO_IC.full;
            transaction.empty = FIFO_IC.empty;
            transaction.almostfull = FIFO_IC.almostfull;
            transaction.almostempty = FIFO_IC.almostempty;
            transaction.overflow = FIFO_IC.overflow;
            transaction.underflow = FIFO_IC.underflow;
            transaction.data_in = FIFO_IC.data_in;
            transaction.rst_n = FIFO_IC.rst_n;
            transaction.data_out = FIFO_IC.data_out;

            fork
                begin
                    coverage.sample_data(transaction);
                end

                begin
                    score.check_data(transaction);
                end
            join

            if (test_finished) begin
                $display("Error count  = %0d, correct count = %0d",Error_count, Correct_count);
                $stop;
            end
        end
    end
endmodule
```

Transaction package:

```
package trans_pkg;

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

    class  FIFO_transaction;

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

        int RD_EN_ON_DIST, WR_EN_ON_DIST;

        function new(int x = 30, int y = 70);

            RD_EN_ON_DIST = x;
            WR_EN_ON_DIST = y;

        endfunction

        constraint  A {rst_n dist {1:/90, 0:/10};}
        constraint  B {wr_en dist {1:/WR_EN_ON_DIST, 0:/100-WR_EN_ON_DIST};}
        constraint  C {rd_en dist {1:/RD_EN_ON_DIST, 0:/100-RD_EN_ON_DIST};}



    endclass

endpackage : trans_pkg
```

Coverage package:

```
1   package coverage_pkg;
2
3       import trans_pkg::*;
4
5       class  FIFO_coverage;
6
7           FIFO_transaction F_cvg_txn;
8
9           covergroup cg ;
10
11              cross_0: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.wr_ack;
12              cross_1: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.overflow;
13              cross_2: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.full;
14              cross_3: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.empty;
15              cross_4: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostfull;
16              cross_5: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostempty;
17              cross_6: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.underflow;
18
19          endgroup
20
21          function void sample_data(FIFO_transaction F_txn);
22              F_cvg_txn = F_txn;
23              cg.sample();
24          endfunction
25
26
27          function new();
28              cg=new();
29          endfunction
30
31      endclass
32
33  endpackage : coverage_pkg
```

Scoreboard package:

```
package scoreboard_pkg;
    import trans_pkg::*;
    import shared_pkg::*;

    logic [FIFO_WIDTH-1:0] fifo_queue[$];

    class FIFO_scoreboard;
        logic [FIFO_WIDTH-1:0] data_out_ref;
        logic wr_ack_ref;
        logic [2:0] wr_ptr, rd_ptr;
        logic [3:0] fifo_count;
        logic full_ref;
        logic empty_ref;


        function void check_data(input FIFO_transaction F_txn );

            reference_model(F_txn);
            if (data_out_ref != F_txn.data_out) begin
                $display("Error with data_out_ref = %0d and F_txn.data_out = %0d", data_out_ref,F_txn.data_out);
                Error_count++;
            end
            else begin
                Correct_count++;
                $display("correct with data_out_ref = %0d and F_txn.data_out = %0d", data_out_ref,F_txn.data_out);
            end
        endfunction

        function reference_model(input FIFO_transaction F_txn);

            if (!F_txn.rst_n) begin
                fifo_queue <= {};
                fifo_count <= 0;
            end
            else begin
                if (F_txn.wr_en && fifo_count < FIFO_DEPTH) begin
                    fifo_queue.push_back(F_txn.data_in);
                    fifo_count <= fifo_queue.size();
                end


                if (F_txn.rd_en && fifo_count != 0) begin
                    data_out_ref <= fifo_queue.pop_front();
                    fifo_count <= fifo_queue.size();
                end
            end

            full_ref = (fifo_count == FIFO_DEPTH);
            empty_ref = (fifo_count == 0);

        endfunction
    endclass
endpackage : scoreboard_pkg
```

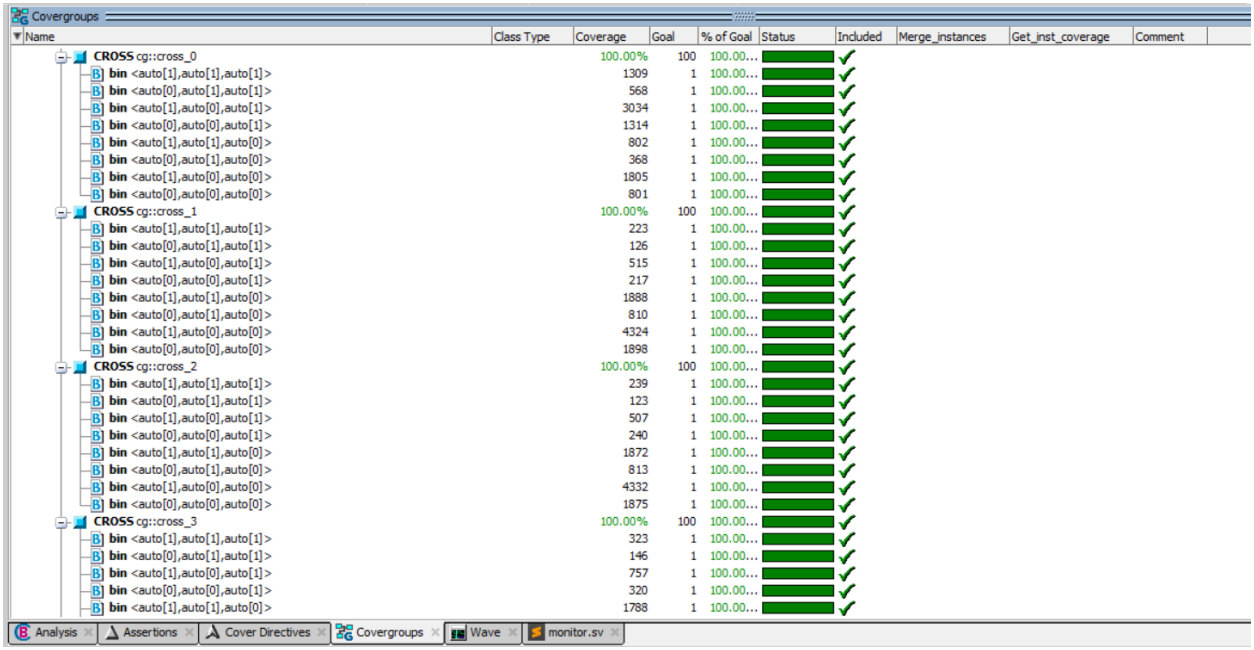
Shared package:

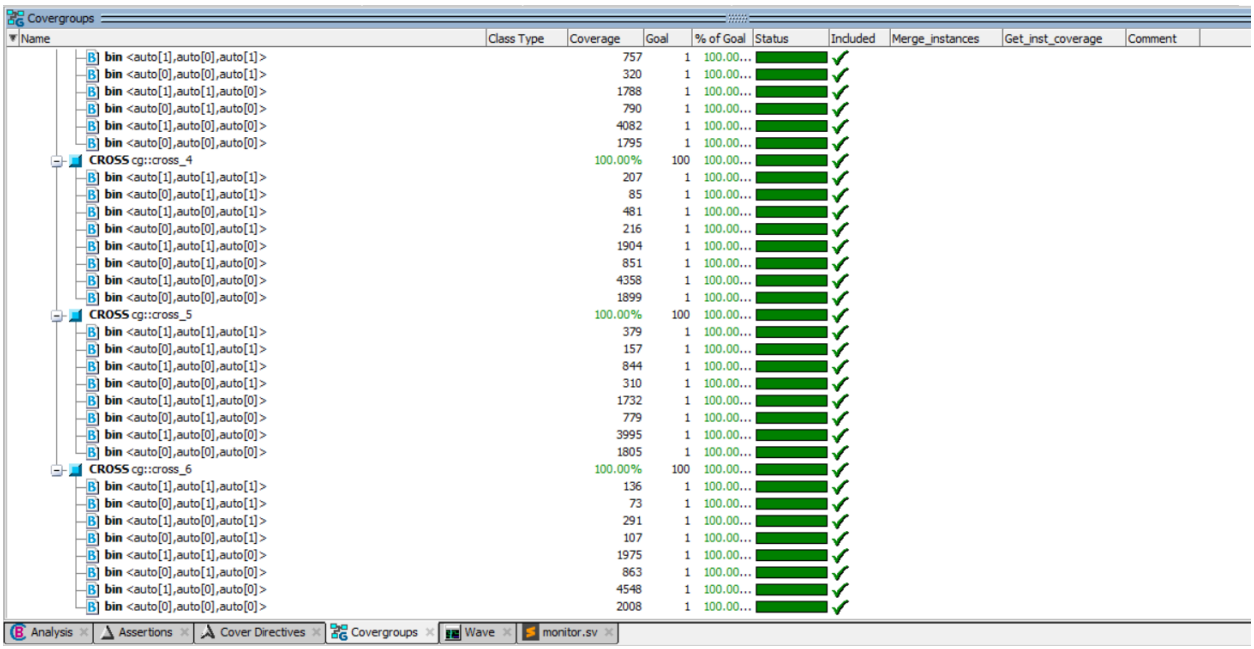
```
package shared_pkg;

    int Error_count, Correct_count;
    bit test_finished;

endpackage
```

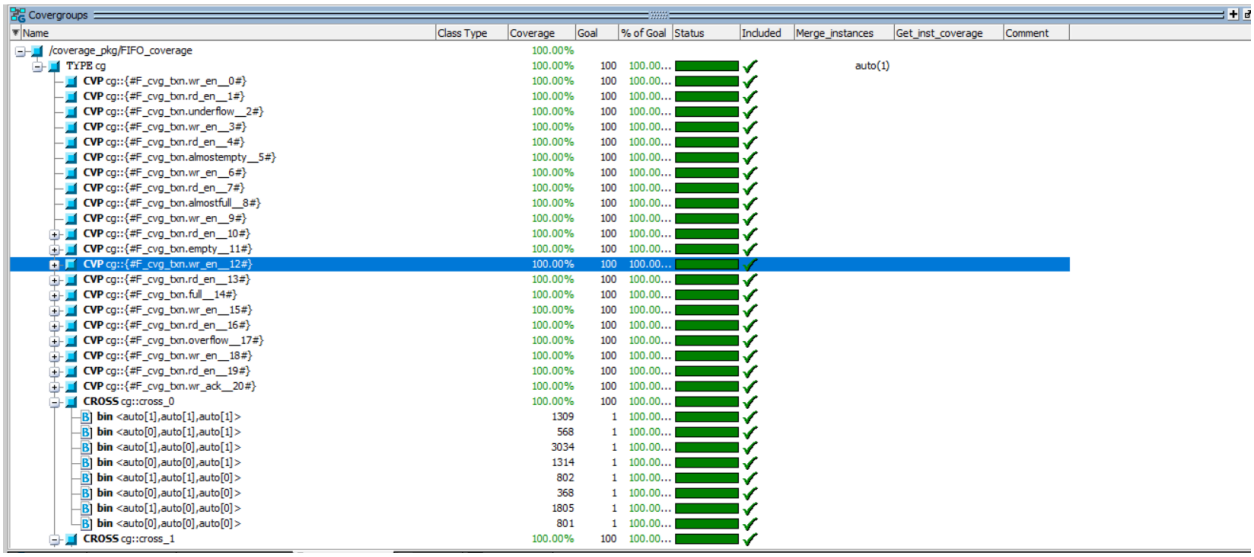
Do file:

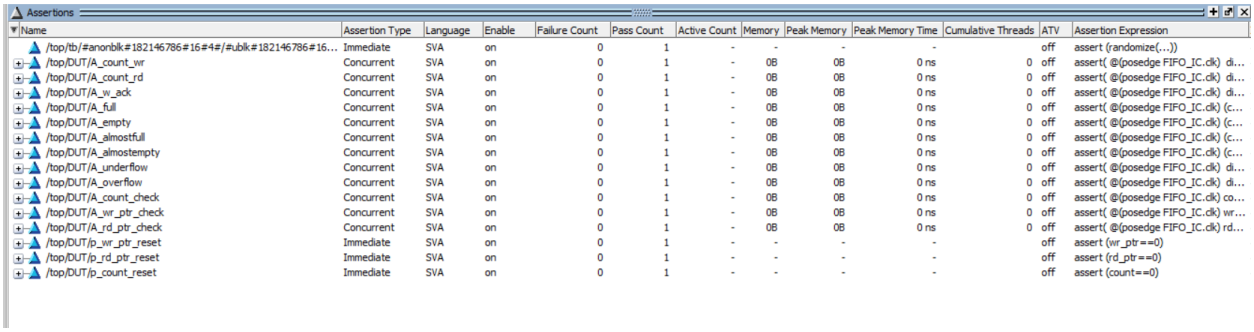
```
vlib work
vlog interface.sv pkg_trans.sv pkg_coverage.sv pkg_scoreboard.sv shared_pkg.sv FIFO.sv monitor.sv FIFO_tb.sv top.sv +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave *
coverage save top.ucdb -onexit
run -all
```

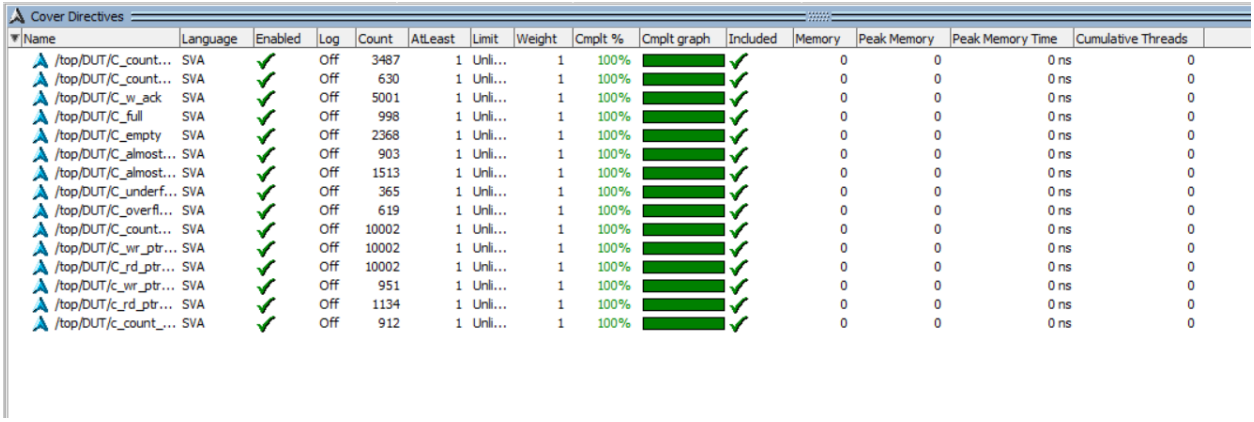
Coverage report:












```
=== Instance: /top/FIFO_IC
=== Design Unit: work.FIFO_interface
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%
```

```
================================================================================
=== Instance: /top/tb
=== Design Unit: work.FIFO_tb
================================================================================

Assertion Coverage:
    Assertions                       1         1         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/tb/#anonblk#182146786#16#4#/#ublk#182146786#16/immed__18
                     FIFO_tb.sv(18)                     0          1
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     15        15         0   100.00%
```

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

Assertion Coverage:
    Assertions                      15        15         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUT/A_count_wr  FIFO.sv(147)                       0          1
/top/DUT/A_count_rd  FIFO.sv(152)                       0          1
/top/DUT/A_w_ack     FIFO.sv(157)                       0          1
/top/DUT/A_full      FIFO.sv(162)                       0          1
/top/DUT/A_empty     FIFO.sv(167)                       0          1
/top/DUT/A_almostfull
                     FIFO.sv(172)                       0          1
/top/DUT/A_almostempty
                     FIFO.sv(177)                       0          1
/top/DUT/A_underflow FIFO.sv(182)                       0          1
/top/DUT/A_overflow  FIFO.sv(187)                       0          1
/top/DUT/A_count_check
                     FIFO.sv(192)                       0          1
/top/DUT/A_wr_ptr_check
                     FIFO.sv(197)                       0          1
/top/DUT/A_rd_ptr_check
                     FIFO.sv(202)                       0          1
/top/DUT/p_wr_ptr_reset
                     FIFO.sv(208)                       0          1
/top/DUT/p_rd_ptr_reset
                     FIFO.sv(211)                       0          1
/top/DUT/p_count_reset
                     FIFO.sv(214)                       0          1
Branch Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Branches                       29        29         0   100.00%
```

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

```
Directive Coverage:
    Directives                      15        15         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT/C_count_wr                      FIFO   Verilog  SVA  FIFO.sv(148)    3487 Covered
/top/DUT/C_count_rd                      FIFO   Verilog  SVA  FIFO.sv(153)     630 Covered
/top/DUT/C_w_ack                         FIFO   Verilog  SVA  FIFO.sv(158)    5001 Covered
/top/DUT/C_full                          FIFO   Verilog  SVA  FIFO.sv(163)     998 Covered
/top/DUT/C_empty                         FIFO   Verilog  SVA  FIFO.sv(168)    2368 Covered
/top/DUT/C_almostfull                    FIFO   Verilog  SVA  FIFO.sv(173)     903 Covered
/top/DUT/C_almostempty                   FIFO   Verilog  SVA  FIFO.sv(178)    1513 Covered
/top/DUT/C_underflow                     FIFO   Verilog  SVA  FIFO.sv(183)     365 Covered
/top/DUT/C_overflow                      FIFO   Verilog  SVA  FIFO.sv(188)     619 Covered
/top/DUT/C_count_check                   FIFO   Verilog  SVA  FIFO.sv(193)   10002 Covered
/top/DUT/C_wr_ptr_check                  FIFO   Verilog  SVA  FIFO.sv(198)   10002 Covered
/top/DUT/C_rd_ptr_check                  FIFO   Verilog  SVA  FIFO.sv(203)   10002 Covered
/top/DUT/c_wr_ptr_reset                  FIFO   Verilog  SVA  FIFO.sv(209)     951 Covered
/top/DUT/c_rd_ptr_reset                  FIFO   Verilog  SVA  FIFO.sv(212)    1134 Covered
/top/DUT/c_count_reset                   FIFO   Verilog  SVA  FIFO.sv(215)     912 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      25        25         0   100.00%

================================Statement Details================================
```

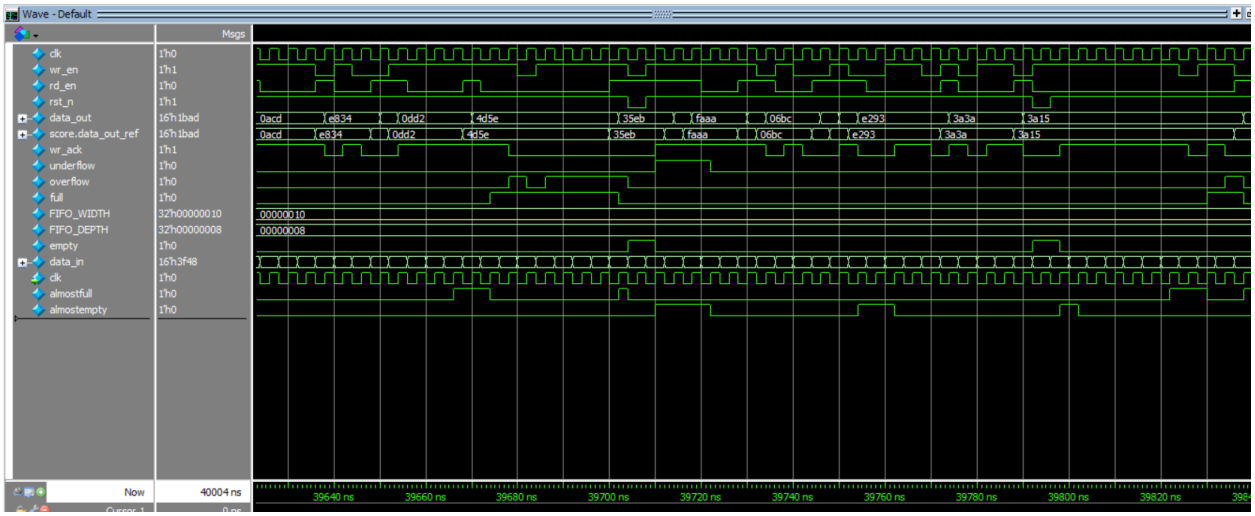
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         20        20         0   100.00%

================================Toggle Details================================
```

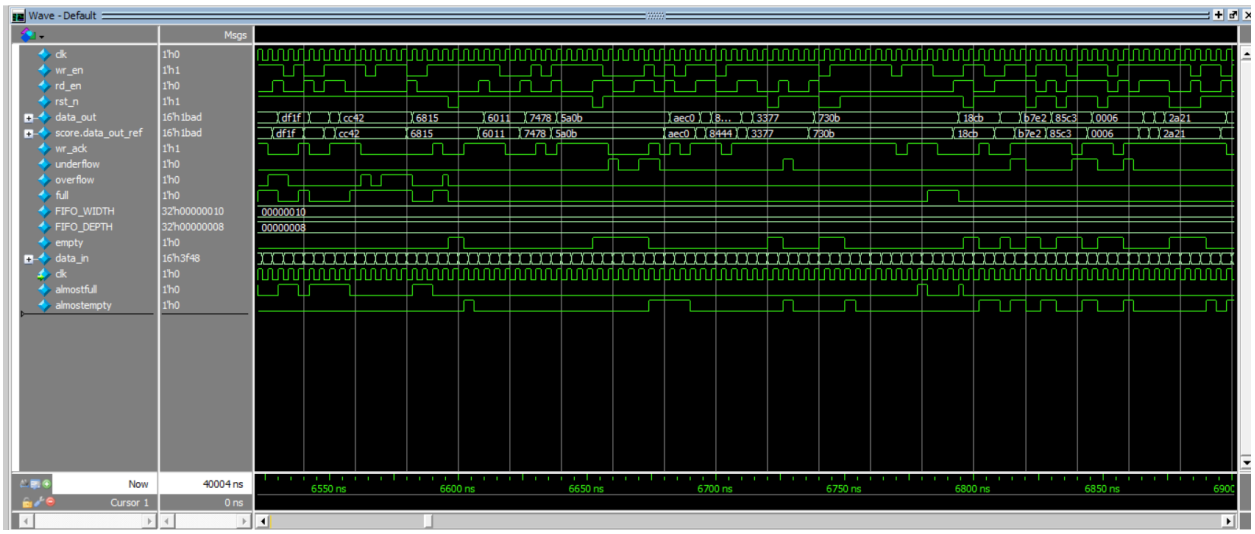
```
Covergroup Coverage:
    Covergroups                      1        na        na   100.00%
        Coverpoints/Crosses         28        na        na        na
            Covergroup Bins         98        98         0   100.00%
--------------------------------------------------------------------------------
```

```
            bin <auto[0],auto[0],auto[0]>              2008          1          -    Cov
 Statement Coverage:
     Enabled Coverage              Bins      Hits    Misses  Coverage
     ----------------              ----      ----    ------  --------
     Statements                       3         3         0   100.00%
```

```
TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT/C_count_wr                      FIFO   Verilog  SVA  FIFO.sv(148)    3487 Covered
/top/DUT/C_count_rd                      FIFO   Verilog  SVA  FIFO.sv(153)     630 Covered
/top/DUT/C_w_ack                         FIFO   Verilog  SVA  FIFO.sv(158)    5001 Covered
/top/DUT/C_full                          FIFO   Verilog  SVA  FIFO.sv(163)     998 Covered
/top/DUT/C_empty                         FIFO   Verilog  SVA  FIFO.sv(168)    2368 Covered
/top/DUT/C_almostfull                    FIFO   Verilog  SVA  FIFO.sv(173)     903 Covered
/top/DUT/C_almostempty                   FIFO   Verilog  SVA  FIFO.sv(178)    1513 Covered
/top/DUT/C_underflow                     FIFO   Verilog  SVA  FIFO.sv(183)     365 Covered
/top/DUT/C_overflow                      FIFO   Verilog  SVA  FIFO.sv(188)     619 Covered
/top/DUT/C_count_check                   FIFO   Verilog  SVA  FIFO.sv(193)   10002 Covered
/top/DUT/C_wr_ptr_check                  FIFO   Verilog  SVA  FIFO.sv(198)   10002 Covered
/top/DUT/C_rd_ptr_check                  FIFO   Verilog  SVA  FIFO.sv(203)   10002 Covered
/top/DUT/c_wr_ptr_reset                  FIFO   Verilog  SVA  FIFO.sv(209)     951 Covered
/top/DUT/c_rd_ptr_reset                  FIFO   Verilog  SVA  FIFO.sv(212)    1134 Covered
/top/DUT/c_count_reset                   FIFO   Verilog  SVA  FIFO.sv(215)     912 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 15

ASSERTION RESULTS:
-----------------------------------------------------------------
Name                 File(Line)                 Failure      Pass
                                                Count        Count
-----------------------------------------------------------------
/top/tb/#anonblk#182146786#16#4#/#ublk#182146786#16/immed__18
                     FIFO_tb.sv(18)                    0          1
/top/DUT/A_count_wr  FIFO.sv(147)                      0          1
/top/DUT/A_count_rd  FIFO.sv(152)                      0          1
/top/DUT/A_w_ack     FIFO.sv(157)                      0          1
/top/DUT/A_full      FIFO.sv(162)                      0          1
/top/DUT/A_empty     FIFO.sv(167)                      0          1
/top/DUT/A_almostfull
                     FIFO.sv(172)                      0          1
/top/DUT/A_almostempty
                     FIFO.sv(177)                      0          1
/top/DUT/A_underflow FIFO.sv(182)                      0          1
/top/DUT/A_overflow  FIFO.sv(187)                      0          1
/top/DUT/A_count_check
                     FIFO.sv(192)                      0          1
/top/DUT/A_wr_ptr_check
                     FIFO.sv(197)                      0          1
/top/DUT/A_rd_ptr_check
                     FIFO.sv(202)                      0          1
/top/DUT/p_wr_ptr_reset
                     FIFO.sv(208)                      0          1
/top/DUT/p_rd_ptr_reset
                     FIFO.sv(211)                      0          1
/top/DUT/p_count_reset
                     FIFO.sv(214)                      0          1
```

Simulations:





```
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 42592 and F_txn.data_out = 42592
# correct with data_out_ref = 29129 and F_txn.data_out = 29129
# correct with data_out_ref = 29129 and F_txn.data_out = 29129
# correct with data_out_ref = 4439 and F_txn.data_out = 4439
# correct with data_out_ref = 4439 and F_txn.data_out = 4439
# correct with data_out_ref = 4439 and F_txn.data_out = 4439
# correct with data_out_ref = 4439 and F_txn.data_out = 4439
# correct with data_out_ref = 4439 and F_txn.data_out = 4439
# correct with data_out_ref = 51761 and F_txn.data_out = 51761
# correct with data_out_ref = 51761 and F_txn.data_out = 51761
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 60277 and F_txn.data_out = 60277
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 5780 and F_txn.data_out = 5780
# correct with data_out_ref = 7085 and F_txn.data_out = 7085
# correct with data_out_ref = 7085 and F_txn.data_out = 7085
# correct with data_out_ref = 7085 and F_txn.data_out = 7085
# Error count  = 0, correct count = 10002
```