Verification of Synchronized FIFO using SV and Assertions

Ahmed Mohamed Abdellatif

Contents

1) Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	when the reset is asserted the ouput and internal counters should be low	Directed at the start of simlation then randomized during simulation with constrain to be low most of the time		immediate assertion and a checker in the scoreboard to check for reset functionality
FIFO_2	when reset is de asserted and write signal is asserted and the FIFO is not full then the value of data_in writed in the write pointer location and the write_ack flag will be high and the counter wil increase	Randomized during simulation with constrains on write signal to be high 70% of time	cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and increasing internal counters
FIFO_3	when reset is de asserted and read signal is asserted and the FIFO is not empty then data_out will take the value in the read pointer location and the counter wil decrease	Randomized during simulation with constrains on the read signal to be high 30% of time	cover group use cross coverage between write, read and output flags	a checker of the data_out value in the scoreboard and cocurrent assertion to check the output flags and decreasing internal counters
FIFO_4	when count is equal to the FIFO depth the full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and increasing internal
FIFO_5	when count is equal 0 depth the full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_6	when count is equal to the FIFO depth-1 the almost full flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_7	when count is equal to 1 the almost empty flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_8	when reset is deasserted and write signal is asserted and the FIFO is full then overflow flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters
FIFO_9	when reset is deasserted and read signal is asserted and the FIFO is empty then underflow flag will be asserted		cover group use cross coverage between write, read and output flags	cocurrent assertion to check the output flags and internal counters

2) Bugs detected

- **In the continuous assignment of the almost full signal the condition was (count == F.FIFO_DEPTH-2) and it should be (count == F.FIFO_DEPTH-1).**

```
assign F.almostfull = (count == F.FIFO_DEPTH-1)? 1 : 0;
```

- **The counter handling was missing the possibility that the read and write signal could be high together so we should handle this.**

```
always @(posedge F.clk or negedge F.rst_n) begin
    if (!F.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({F.wr_en, F.rd_en} == 2'b10) && !F.full)
            count <= count + 1;
        else if ( ({F.wr_en, F.rd_en} == 2'b01) && !F.empty)
            count <= count - 1;
        else if (({F.wr_en, F.rd_en} == 2'b11) && F.empty)
            count <= count + 1;
        else if (({F.wr_en, F.rd_en} == 2'b11) && F.full)
            count <= count - 1;
    end
end
```

3) Design after solving bugs

```
module FIFO_dut (FIFO_if.DUT F);

    localparam max_fifo_addr = $clog2(F.FIFO_DEPTH);

    reg [F.FIFO_WIDTH-1:0] mem [F.FIFO_DEPTH-1:0];

    reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    reg [max_fifo_addr:0] count;

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

```
  always @(posedge F.clk or negedge F.rst_n) begin
      if (!F.rst_n) begin
          rd_ptr <= 0;
          F.underflow <= 0;
      end
      else if (F.rd_en && count != 0) begin
          F.data_out <= mem[rd_ptr];
          rd_ptr <= rd_ptr + 1;
      end else begin
          if (F.empty && F.rd_en)
              F.underflow <= 1;
          else
              F.underflow <= 0;
      end
  end

  always @(posedge F.clk or negedge F.rst_n) begin
      if (!F.rst_n) begin
          count <= 0;
      end
      else begin
          if  ( ({F.wr_en, F.rd_en} == 2'b10) && !F.full)
              count <= count + 1;
          else if ( ({F.wr_en, F.rd_en} == 2'b01) && !F.empty)
              count <= count - 1;
          else if (({F.wr_en, F.rd_en} == 2'b11) && F.empty)
              count <= count + 1;
          else if (({F.wr_en, F.rd_en} == 2'b11) && F.full)
              count <= count - 1;
      end
  end
```

4) Top module

```
module FIFO_top ();
    bit clk;
    initial begin
        clk=0;
        forever begin
            #2 clk=~clk;
        end
    end

    FIFO_if F (clk);
    FIFO_test test (F);
    FIFO_dut dut (F);
    FIFO_monitor mon (F);

endmodule : FIFO_top
```

5) Interface

```
interface FIFO_if (input clk);
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    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 DUT (input clk,data_in, 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 clk,data_in, rst_n, wr_en, rd_en,
        data_out, wr_ack, overflow, full, empty, almostfull, almostempty, underflow);

endinterface : FIFO_if
```

6) Testbench

```
module FIFO_test (FIFO_if.TEST F);
    import shared_package::*;
    import FIFO_transaction_::*;

    FIFO_transaction t = new ;

    initial begin
        F.rst_n = 0;
        #10;
        F.rst_n = 1;

        repeat (1000) begin
            @(negedge F.clk);
            assert(t.randomize());
            F.data_in = t.data_in ;
            F.rst_n = t.rst_n ;
            F.wr_en = t.wr_en ;
            F.rd_en = t.rd_en;
            t.data_out = F.data_out ;
            t.wr_ack = F.wr_ack ;
            t.underflow = F.underflow ;
            t.overflow = F.overflow ;
            t.full = F.full ;
            t.empty = F.empty ;
            t.almostfull = F.almostfull ;
            t.almostempty = F.almostempty ;
        end
        test_finished = 1;
    end

endmodule : FIFO_test
```

7) Coverage

```
package FIFO_coverage_;

    import FIFO_transaction_::*;
    FIFO_transaction F_cvg_txn;

    class FIFO_coverage;
        covergroup cg;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.wr_ack;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.full;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.empty;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.almostfull;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.almostempty;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.overflow;
            cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.underflow;
        endgroup : cg

        function void sample_data(FIFO_transaction F_txn);
            F_cvg_txn = F_txn;
            cg.sample();
        endfunction

        function new();
            cg = new ;
        endfunction
    endclass : FIFO_coverage
endpackage : FIFO_coverage_
```

8) Scoreboard

```
package FIFO_scoreboard_;
    import FIFO_transaction_::*;
    import shared_package::*;

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    localparam max_fifo_addr = $clog2(FIFO_DEPTH);

    logic [FIFO_WIDTH-1:0] data_out_ref;
    logic wr_ack_ref, overflow_ref;
    logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;

    class FIFO_scoreboard;
            logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
            logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
            logic [max_fifo_addr:0] count;

        function check_data(FIFO_transaction F_sb_txn);
            if (F_sb_txn.data_out!=data_out_ref || F_sb_txn.wr_ack!=wr_ack_ref
            || F_sb_txn.overflow!=overflow_ref || F_sb_txn.full!=full_ref
            || F_sb_txn.empty!=empty_ref || F_sb_txn.almostfull!=almostfull_ref
            || F_sb_txn.almostempty!=almostempty_ref || F_sb_txn.underflow!=underflow_ref) begin
                $display("Error where data_out_ref=%d,wr_ack_ref=%d,overflow_ref=%d,full_ref=%d,
                    empty_ref=%d,almostfull_ref=%d,almostempty_ref=%d,underflow_ref=%d
                    and
                    data_out=%d,wr_ack=%d,overflow=%d,full=%d,empty=%d,almostfull=%d,almostempty=%d,underflow=%d",
                    data_out_ref,wr_ack_ref, overflow_ref,
                    full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref,
                    F_sb_txn.data_out,F_sb_txn.wr_ack,F_sb_txn.overflow,F_sb_txn.full,
                    F_sb_txn.empty,F_sb_txn.almostfull,F_sb_txn.almostempty,F_sb_txn.underflow);
                $stop;
                error_count++;
            end else begin
                correct_count++;
            end
            reference_model(F_sb_txn);
        endfunction
```

```
    function reference_model(FIFO_transaction F_sb_txn);

            if (!F_sb_txn.rst_n) begin
                wr_ptr = 0;
                rd_ptr = 0;
                count = 0;
                underflow_ref = 0;
                overflow_ref = 0;
            end
            else begin
                /* Read operatio before write operation to read the right value
                when read and write signals asserted together*/

                if (F_sb_txn.rd_en && count != 0) begin
                    data_out_ref = mem[rd_ptr];
                    rd_ptr = rd_ptr + 1;
                    count = count - 1;
                end else begin
                    if (empty_ref && F_sb_txn.rd_en)
                        underflow_ref = 1;
                    else
                        underflow_ref = 0;
                end

                if (F_sb_txn.wr_en && !full_ref) begin
                    mem[wr_ptr] = F_sb_txn.data_in;
                    wr_ack_ref = 1;
                    wr_ptr = wr_ptr + 1;
                    count = count + 1;
                end
                else begin
                    wr_ack_ref = 0;
                    if (full_ref & F_sb_txn.wr_en)
                        overflow_ref = 1;
                    else
                        overflow_ref = 0;
                end
            end
```

```
            if (count == FIFO_DEPTH)
                full_ref = 1;
            else
                full_ref = 0;

            if (count == 0)
                empty_ref = 1;
            else
                empty_ref = 0;

            if (count == FIFO_DEPTH-1)
                almostfull_ref = 1;
            else
                almostfull_ref = 0;

            if (count == 1)
                almostempty_ref = 1;
            else
                almostempty_ref = 0;
        endfunction
    endclass : FIFO_scoreboard
endpackage : FIFO_scoreboard_
```

9) Assertions

```
// Assertions
// full flag
assert property (@(posedge F.clk) (count == F.FIFO_DEPTH) |-> F.full );
cover property (@(posedge F.clk) (count == F.FIFO_DEPTH) |-> F.full );
// empty flag
assert property (@(posedge F.clk) (count == 0) |-> F.empty );
cover property (@(posedge F.clk) (count == 0) |-> F.empty );
// almost full flag
assert property (@(posedge F.clk) (count == F.FIFO_DEPTH-1) |-> F.almostfull );
cover property (@(posedge F.clk) (count == F.FIFO_DEPTH-1) |-> F.almostfull);
// almost empty flag
assert property (@(posedge F.clk) (count == 1) |-> F.almostempty );
cover property (@(posedge F.clk) (count == 1) |-> F.almostempty );
// Over flow flag
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.full && F.wr_en) |=> (F.overflow));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.full && F.wr_en) |=> (F.overflow));
// Under flow flag
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.empty && F.rd_en) |=> (F.underflow));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.empty && F.rd_en) |=> (F.underflow));
// Write acknoledge flag
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full) |=> (F.wr_ack));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full) |=> (F.wr_ack));
```

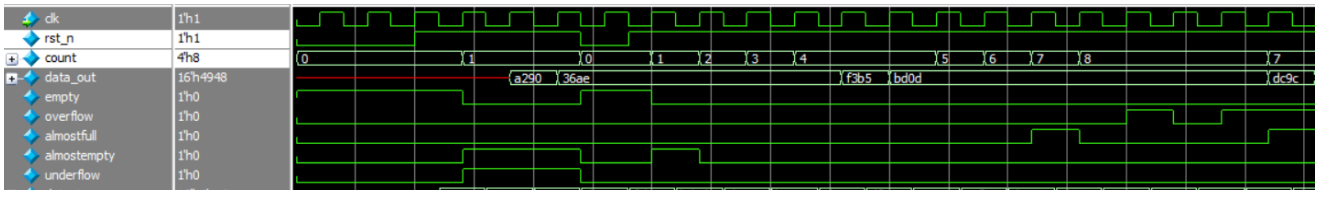
```
// internal counters
// Write pointer
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && wr_ptr!=7) |=> (wr_ptr==$past(wr_ptr)+1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && wr_ptr!=7) |=> (wr_ptr==$past(wr_ptr)+1));
// Write pointer if we wrote in the 8 places we will return to the beginning
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && wr_ptr==7) |=> (wr_ptr==0));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && wr_ptr==7) |=> (wr_ptr==0));
// Read pointer
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && rd_ptr!=7) |=> (rd_ptr==$past(rd_ptr)+1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && rd_ptr!=7) |=> (rd_ptr==$past(rd_ptr)+1));
// Read pointer if we read the 8 places we will return to the beginning
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && rd_ptr==7) |=> (rd_ptr==0));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && rd_ptr==7) |=> (rd_ptr==0));
// counter
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && !F.rd_en) |=> (count==$past(count)+1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && !F.full && !F.rd_en) |=> (count==$past(count)+1));
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && !F.wr_en) |=> (count==$past(count)-1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.rd_en && !F.empty && !F.wr_en) |=> (count==$past(count)-1));
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && F.empty && F.rd_en) |=> (count==$past(count)+1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && F.empty && F.rd_en) |=> (count==$past(count)+1));
assert property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && F.full && F.rd_en) |=> (count==$past(count)-1));
cover property (@(posedge F.clk) disable iff(!F.rst_n) (F.wr_en && F.full && F.rd_en) |=> (count==$past(count)-1));
```

10) Do file

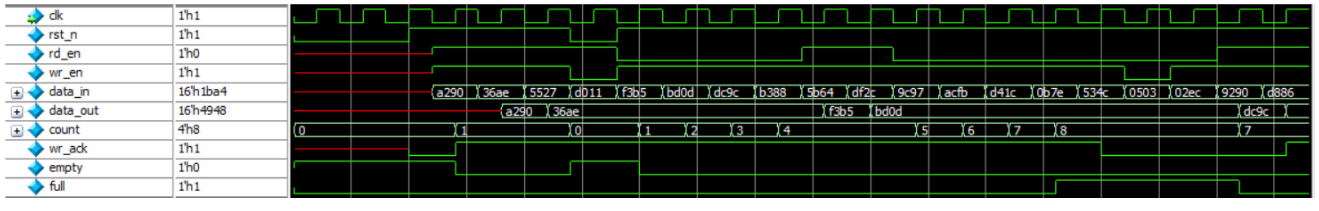
```
vlib work
vlog FIFO_if.sv shared_package.sv FIFO_transaction.sv FIFO_scoreboard.sv FIFO_coverage.sv FIFO_test.sv FIFO_dut.sv  FIFO_monitor.sv
FIFO_top.sv +cover -covercells
vsim -voptargs=+acc work.FIFO_top -cover -sv_seed random -l sim.log
add wave -position insertpoint sim:/FIFO_top/dut/F/*
coverage save FIFO_top.ucdb -onexit
run -all
```

11) Waveform

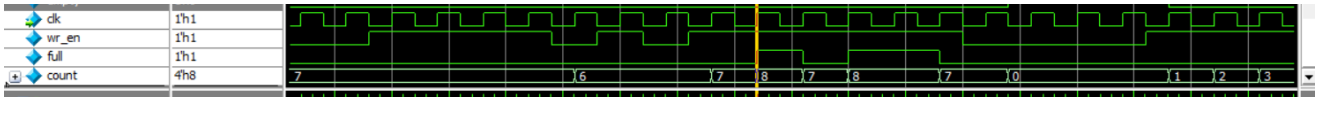
FIFO_1:



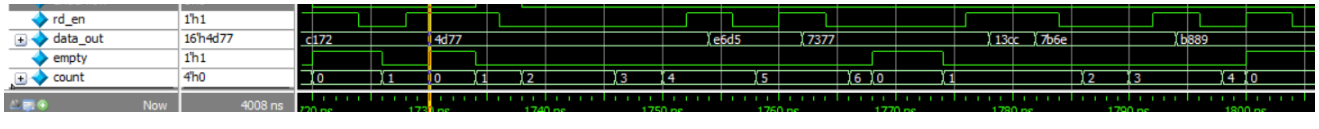
FIFO_2, FIFO_3:



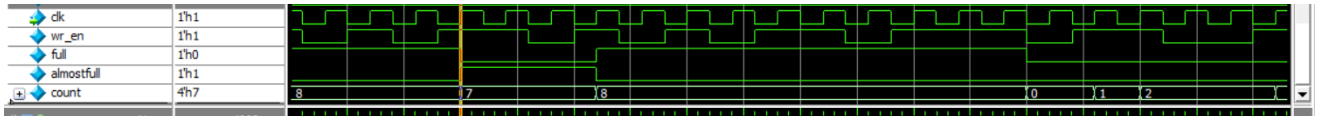
FIFO_4:



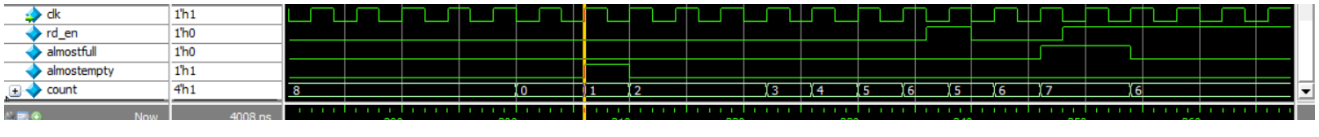
FIFO_5:



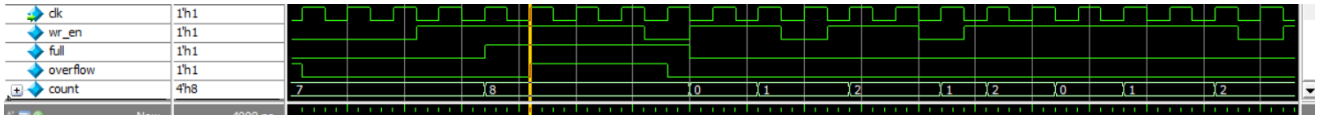
FIFO_6:



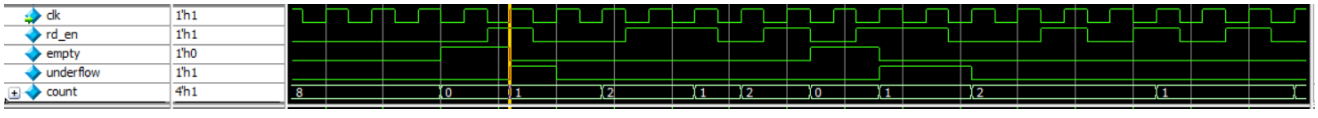
FIFO_7:



FIFO_8:


clk
wr_en
full
overflow
count
1'h1
1'h1
1'h1
1'h1
4'h8
7
8
0
1
2
1
2
0
1
2

FIFO_9:


clk
rd_en
empty
underflow
count
1'h1
1'h1
1'h0
1'h1
4'h1
8
0
1
2
1
2
0
1
2
1

12) Coverage report

Assertion coverage

```
Assertion Coverage:
    Assertions                      15        15         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/FIFO_top/dut/assert__14
                     FIFO_dut.sv(108)                0          1
/FIFO_top/dut/assert__13
                     FIFO_dut.sv(106)                0          1
/FIFO_top/dut/assert__12
                     FIFO_dut.sv(104)                0          1
/FIFO_top/dut/assert__11
                     FIFO_dut.sv(102)                0          1
/FIFO_top/dut/assert__10
                     FIFO_dut.sv(99)                 0          1
/FIFO_top/dut/assert__9
                     FIFO_dut.sv(96)                 0          1
/FIFO_top/dut/assert__8
                     FIFO_dut.sv(93)                 0          1
/FIFO_top/dut/assert__7
                     FIFO_dut.sv(90)                 0          1
/FIFO_top/dut/assert__6
                     FIFO_dut.sv(86)                 0          1
/FIFO_top/dut/assert__5
                     FIFO_dut.sv(83)                 0          1
/FIFO_top/dut/assert__4
                     FIFO_dut.sv(80)                 0          1
/FIFO_top/dut/assert__3
                     FIFO_dut.sv(77)                 0          1
/FIFO_top/dut/assert__2
                     FIFO_dut.sv(74)                 0          1
/FIFO_top/dut/assert__1
                     FIFO_dut.sv(71)                 0          1
/FIFO_top/dut/assert__0
                     FIFO_dut.sv(68)                 0          1
```

Directive coverage

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

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------
Name                                Design Design   Lang File(Line)      Hits Status
                                    Unit   UnitType
--------------------------------------------------------------------------------------
/FIFO_top/dut/cover__14             FIFO_dut Verilog  SVA  FIFO_dut.sv(109)  46 Covered
/FIFO_top/dut/cover__13             FIFO_dut Verilog  SVA  FIFO_dut.sv(107)  16 Covered
/FIFO_top/dut/cover__12             FIFO_dut Verilog  SVA  FIFO_dut.sv(105)  75 Covered
/FIFO_top/dut/cover__11             FIFO_dut Verilog  SVA  FIFO_dut.sv(103) 338 Covered
/FIFO_top/dut/cover__10             FIFO_dut Verilog  SVA  FIFO_dut.sv(100)  15 Covered
/FIFO_top/dut/cover__9              FIFO_dut Verilog  SVA  FIFO_dut.sv(97)  219 Covered
/FIFO_top/dut/cover__8              FIFO_dut Verilog  SVA  FIFO_dut.sv(94)   42 Covered
/FIFO_top/dut/cover__7              FIFO_dut Verilog  SVA  FIFO_dut.sv(91)  425 Covered
/FIFO_top/dut/cover__6              FIFO_dut Verilog  SVA  FIFO_dut.sv(87)  467 Covered
/FIFO_top/dut/cover__5              FIFO_dut Verilog  SVA  FIFO_dut.sv(84)   27 Covered
/FIFO_top/dut/cover__4              FIFO_dut Verilog  SVA  FIFO_dut.sv(81)  154 Covered
/FIFO_top/dut/cover__3              FIFO_dut Verilog  SVA  FIFO_dut.sv(78)   89 Covered
/FIFO_top/dut/cover__2              FIFO_dut Verilog  SVA  FIFO_dut.sv(75)  185 Covered
/FIFO_top/dut/cover__1              FIFO_dut Verilog  SVA  FIFO_dut.sv(72)  144 Covered
/FIFO_top/dut/cover__0              FIFO_dut Verilog  SVA  FIFO_dut.sv(69)  236 Covered
```

Branch coverage

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

================================Branch Details================================

Branch Coverage for instance /FIFO_top/dut

    Line         Item                    Count     Source
    ----         ----                    -----     ------
  File FIFO_dut.sv
------------------------------------IF Branch------------------------------------
    11                                    1040     Count coming in to IF
    11              1                       78                if (!F.rst_n) begin

    15              1                      475                else if (F.wr_en && count < F.FIFO_DEPTH) begin

    20              1                      487                else begin

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    22                                    487     Count coming in to IF
    22              1                      186                        if (F.full & F.wr_en)

    24              1                      301                        else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    30                                    1040     Count coming in to IF
    30              1                       78                if (!F.rst_n) begin

    34              1                      267                else if (F.rd_en && count != 0) begin

    37              1                      695                end else begin

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
```

```
------------------------------------IF Branch------------------------------------
    50                                    846     Count coming in to IF
    50              1                     334                         if       ( ({F.wr_en, F.rd_en} == 2'b10) && !F.full)

    52              1                      79                         else if ( ({F.wr_en, F.rd_en} == 2'b01) && !F.empty)

    54              1                      16                         else if (({F.wr_en, F.rd_en} == 2'b11) && F.empty)

    56              1                      63                         else if (({F.wr_en, F.rd_en} == 2'b11) && F.full)

                                          354     All False Count
Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------IF Branch------------------------------------
    61                                    530     Count coming in to IF
    61              1                      98        assign F.full = (count == F.FIFO_DEPTH)? 1 : 0;

    61              2                     432        assign F.full = (count == F.FIFO_DEPTH)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    62                                    530     Count coming in to IF
    62              1                      40        assign F.empty = (count == 0)? 1 : 0;

    62              2                     490        assign F.empty = (count == 0)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                    530     Count coming in to IF
    63              1                     127        assign F.almostfull = (count == F.FIFO_DEPTH-1)? 1 : 0;

    63              2                     403        assign F.almostfull = (count == F.FIFO_DEPTH-1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    64                                    530     Count coming in to IF
    64              1                      46        assign F.almostempty = (count == 1)? 1 : 0;

    64              2                     484        assign F.almostempty = (count == 1)? 1 : 0;

Branch totals: 2 hits of 2 branches = 100.00%
```

Statement coverage

```
Statement Coverage:
    Enabled Coverage                Bins       Hits     Misses  Coverage
    ----------------                ----       ----     ------  --------
    Statements                        26         26          0   100.00%

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

Statement Coverage for instance /FIFO_top/dut --

   Line         Item                     Count     Source
   ----         ----                     -----     ------
  File FIFO_dut.sv
   1                                             module FIFO_dut (FIFO_if.DUT F);
   2
   3                                                localparam max_fifo_addr = $clog2(F.FIFO_DEPTH);
   4
   5                                                reg [F.FIFO_WIDTH-1:0] mem [F.FIFO_DEPTH-1:0];
   6
   7                                                reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
   8                                                reg [max_fifo_addr:0] count;
   9
   10             1                      1040       always @(posedge F.clk or negedge F.rst_n) begin
   11                                                       if (!F.rst_n) begin
   12             1                        78                       wr_ptr <= 0;
   13             1                        78                       F.overflow <= 0;
   14                                                       end
   15                                                       else if (F.wr_en && count < F.FIFO_DEPTH) begin
```

```
16              1                       475                             mem[wr_ptr] <= F.data_in;
    17              1                       475                             wr_ptr <= wr_ptr + 1;
    18              1                       475                             F.wr_ack <= 1;
    19                                                                  end
    20                                                                  else begin
    21              1                       487                             F.wr_ack <= 0;
    22                                                                      if (F.full & F.wr_en)
    23              1                       186                                 F.overflow <= 1;
    24                                                                      else
    25              1                       301                                 F.overflow <= 0;
    26                                                                  end
    27                                                          end
    28
    29              1                      1040        always @(posedge F.clk or negedge F.rst_n) begin
    30                                                                  if (!F.rst_n) begin
    31              1                        78                             rd_ptr <= 0;
    32              1                        78                             F.underflow <= 0;
    33                                                                  end
    34                                                                  else if (F.rd_en && count != 0) begin
35              1                       267                             F.data_out <= mem[rd_ptr];
    36              1                       267                             rd_ptr <= rd_ptr + 1;
    37                                                                  end else begin
    38                                                                      if (F.empty && F.rd_en)
    39              1                        19                                 F.underflow <= 1;
    40                                                                      else
```

```
45              1                       923         always @(posedge F.clk or negedge F.rst_n) begin
46                                                          if (!F.rst_n) begin
47              1                        77                         count <= 0;
48                                                          end
49                                                          else begin
50                                                                  if       ( ({F.wr_en, F.rd_en} == 2'b10) && !F.full)
51              1                       334                         count <= count + 1;
52                                                                  else if ( ({F.wr_en, F.rd_en} == 2'b01) && !F.empty)
53              1                        79                         count <= count - 1;
54                                                                  else if (({F.wr_en, F.rd_en} == 2'b11) && F.empty)
55              1                        16                         count <= count + 1;
56                                                                  else if (({F.wr_en, F.rd_en} == 2'b11) && F.full)
57              1                        63                         count <= count - 1;
58                                                          end
59                                                  end
60
61              1                       531         assign F.full = (count == F.FIFO_DEPTH)? 1 : 0;
62              1                       531         assign F.empty = (count == 0)? 1 : 0;
63              1                       531         assign F.almostfull = (count == F.FIFO_DEPTH-1)? 1 : 0;
64              1                       531         assign F.almostempty = (count == 1)? 1 : 0;
```

Toggle coverage

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /FIFO_top/F --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                    almostempty           1           1      100.00
                                     almostfull           1           1      100.00
                                            clk           1           1      100.00
                                   data_in[15-0]          1           1      100.00
                                  data_out[15-0]          1           1      100.00
                                           empty          1           1      100.00
                                            full          1           1      100.00
                                        overflow          1           1      100.00
                                           rd_en          1           1      100.00
                                           rst_n          1           1      100.00
                                       underflow          1           1      100.00
                                          wr_ack          1           1      100.00
                                           wr_en          1           1      100.00

Total Node Count     =         43
Toggled Node Count   =         43
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (86 of 86 bins)
```

Covergroup coverage

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


----------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_/FIFO_coverage/cg              100.00%        100          -    Covered
    covered/total bins:                                  98         98          -
    missing/total bins:                                   0         98          -
    % Hit:                                           100.00%        100          -
Cross #cross__0#                                     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]>                103          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                 39          1          -    Covered
            bin <auto[1],auto[0],auto[1]>                236          1          -    Covered
            bin <auto[0],auto[0],auto[1]>                111          1          -    Covered
            bin <auto[1],auto[1],auto[0]>                106          1          -    Covered
            bin <auto[0],auto[1],auto[0]>                 48          1          -    Covered
            bin <auto[1],auto[0],auto[0]>                237          1          -    Covered
            bin <auto[0],auto[0],auto[0]>                120          1          -    Covered
    Cross #cross__1#                                 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]>                 65          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                 24          1          -    Covered
            bin <auto[1],auto[0],auto[1]>                130          1          -    Covered
            bin <auto[0],auto[0],auto[1]>                 66          1          -    Covered
            bin <auto[1],auto[1],auto[0]>                144          1          -    Covered
            bin <auto[0],auto[1],auto[0]>                 63          1          -    Covered
            bin <auto[1],auto[0],auto[0]>                343          1          -    Covered
            bin <auto[0],auto[0],auto[0]>                165          1          -    Covered
    Cross #cross__2#                                 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]>                 16          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                  3          1          -    Covered
```

```
Cross #cross__3#                                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]>                45          1          -    Covered
        bin <auto[0],auto[1],auto[1]>                22          1          -    Covered
        bin <auto[1],auto[0],auto[1]>               101          1          -    Covered
        bin <auto[0],auto[0],auto[1]>                48          1          -    Covered
        bin <auto[1],auto[1],auto[0]>               164          1          -    Covered
        bin <auto[0],auto[1],auto[0]>                65          1          -    Covered
        bin <auto[1],auto[0],auto[0]>               372          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               183          1          -    Covered
Cross #cross__4#                                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]>                11          1          -    Covered
        bin <auto[0],auto[1],auto[1]>                 3          1          -    Covered
        bin <auto[1],auto[0],auto[1]>                40          1          -    Covered
        bin <auto[0],auto[0],auto[1]>                20          1          -    Covered
        bin <auto[1],auto[1],auto[0]>               198          1          -    Covered
        bin <auto[0],auto[1],auto[0]>                84          1          -    Covered
        bin <auto[1],auto[0],auto[0]>               433          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               211          1          -    Covered
Cross #cross__5#                                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]>                58          1          -    Covered
        bin <auto[0],auto[1],auto[1]>                21          1          -    Covered
        bin <auto[1],auto[0],auto[1]>               100          1          -    Covered
        bin <auto[0],auto[0],auto[1]>                57          1          -    Covered
        bin <auto[1],auto[1],auto[0]>               151          1          -    Covered
        bin <auto[0],auto[1],auto[0]>                66          1          -    Covered
        bin <auto[1],auto[0],auto[0]>               373          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               174          1          -    Covered
Cross #cross__6#                                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]>                 5          1          -    Covered
        bin <auto[0],auto[1],auto[1]>                 1          1          -    Covered
        bin <auto[1],auto[0],auto[1]>                14          1          -    Covered
        bin <auto[0],auto[0],auto[1]>                 4          1          -    Covered
        bin <auto[1],auto[1],auto[0]>               204          1          -    Covered
        bin <auto[0],auto[1],auto[0]>                86          1          -    Covered
        bin <auto[1],auto[0],auto[0]>               459          1          -    Covered
        bin <auto[0],auto[0],auto[0]>               227          1          -    Covered
```