Synchronous FIFO

Submitted to:-

Eng/Kareem Waseem

Eng/Alaa

Submitted By:-

Mahmoud Lotfy AbdelHamid Amr

mahmoudlotfy383@gmail.com

Table of Contents

I will attach a zip file that will contain all codes if you want to see more.

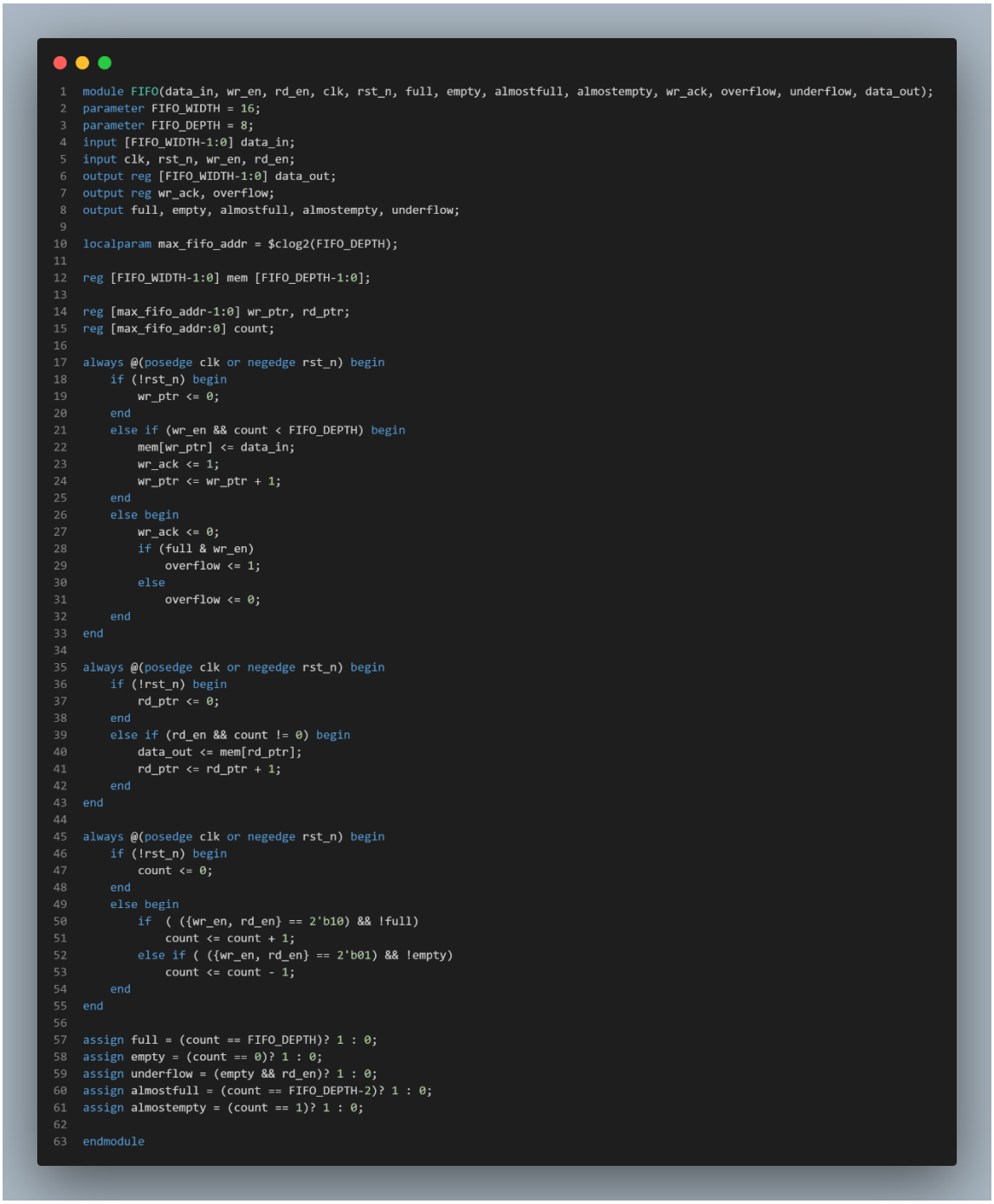
1- Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET	When the reset is asserted, the output flags value should be low and empty flag must be high and make resrt be deasserted most of time	Directed at the start of tb then make it randomized	cover reset	immedtiate assertions to chech async reset and also check the effect of rst in score_board
FIFO_1	the effect when wr_en high and rd_en low should write	randomized during the tb	cover all casses of wr_en and rd_en by cross coverage	concurrent assertions to check all effects of wr_en and rd_en and also by score_board
FIFO_2	the effect when wr_en low and rd_en high should read	randomized during the tb	cover all casses of wr_en and rd_en by cross coverage	concurrent assertions to check all effects of wr_en and rd_en and also by score_board
FIFO_3	the effect when wr_en high and rd_en high and full should rea	randomized during the tb	cover all casses of wr_en and rd_en by cross coverage	concurrent assertions to check all effects of wr_en and rd_en and also by score_board
FIFO_4	the effect when wr_en high and rd_en high and empty write	randomized during the tb	cover all casses of wr_en and rd_en by cross coverage	concurrent assertions to check all effects of wr_en and rd_en and also by score_board
FIFO_5	wr_en high and full	randomized during the tb	cover all casses of wr_en and rd_en and full by cross coverage	concurrent assertions to check all effects of wr_en and also by score_board
FIFO_6	rd_en high and empty	randomized during the tb	cover all casses of wr_en and rd_en and empty by cross coverage	concurrent assertions to check all effects of rd_en and also by score_board
FIFO_7	check almostfull	randomized during the tb	cover all casses of wr_en and rd_en and almostfull by cross coverage	concurrent assertions to check almostfull and also by score_board
FIFO_8	check almostempty	randomized during the tb	cover all casses of wr_en and rd_en and almostempty by cross coverage	concurrent assertions to check almostempty and also by score_board
FIFO_9	check overflow	randomized during the tb	cover all casses of wr_en and rd_en and overflow by cross coverage	concurrent assertions to check overflow and also by score_board
FIFO_10	check underflow	randomized during the tb	cover all casses of wr_en and rd_en and underflow by cross coverage	concurrent assertions to check underflow and also by score_board
FIFO_11	check relation between full and almostfull	randomized during the tb		concurrent assertions to check relation between full and almostfull
FIFO_12	check relation between empty and almostempty	randomized during the tb		concurrent assertions to check relation between empty and almostempty
FIFO_13	check wr_ptr	randomized during the tb		concurrent assertions to check all possible casses for wr_ptr
FIFO_14	check rd_ptr	randomized during the tb		concurrent assertions to check all possible casses for rd_ptr
FIFO_15	check count	randomized during the tb		concurrent assertions to check all possible casses for count
FIFO_16	check max_fifo_addr	randomized during the tb		immediate assertions to check max_fifo_addr

2- Modules

1- FIFO.sv

The design has more than bug there is the original design



```
module FIFO(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
input [FIFO_WIDTH-1:0] data_in;
input clk, rst_n, wr_en, rd_en;
output reg [FIFO_WIDTH-1:0] data_out;
output reg wr_ack, overflow;
output full, empty, almostfull, almostempty, underflow;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

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

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

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

assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign underflow = (empty && rd_en)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
assign almostempty = (count == 1)? 1 : 0;

endmodule
```

BUGS:-

1- Wr_ack must be rested

2- Overflow must be rested

3- almostfull must be (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0 ;
not (count == FIFO_IF.FIFO_DEPTH-2)? 1 : 0;

4- underflow must be sequential output as required not combinational output

5- underflow must be zero if (FIFO_IF.rd_en && count != 0) and that ignored in the original design and that important to do that because not make underflow remain has old value and update each clk cycle
6- overflow must be zero if (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH) and that ignored in the original design and that important to do that because not make overflow remain has old value and update each clk cycle
7- these are two conditions are ignored in design and I add them as required and the two conditions are

> Note: If a read and write enables were high and the FIFO was empty, only writing will take place and vice verse if the FIFO was full.

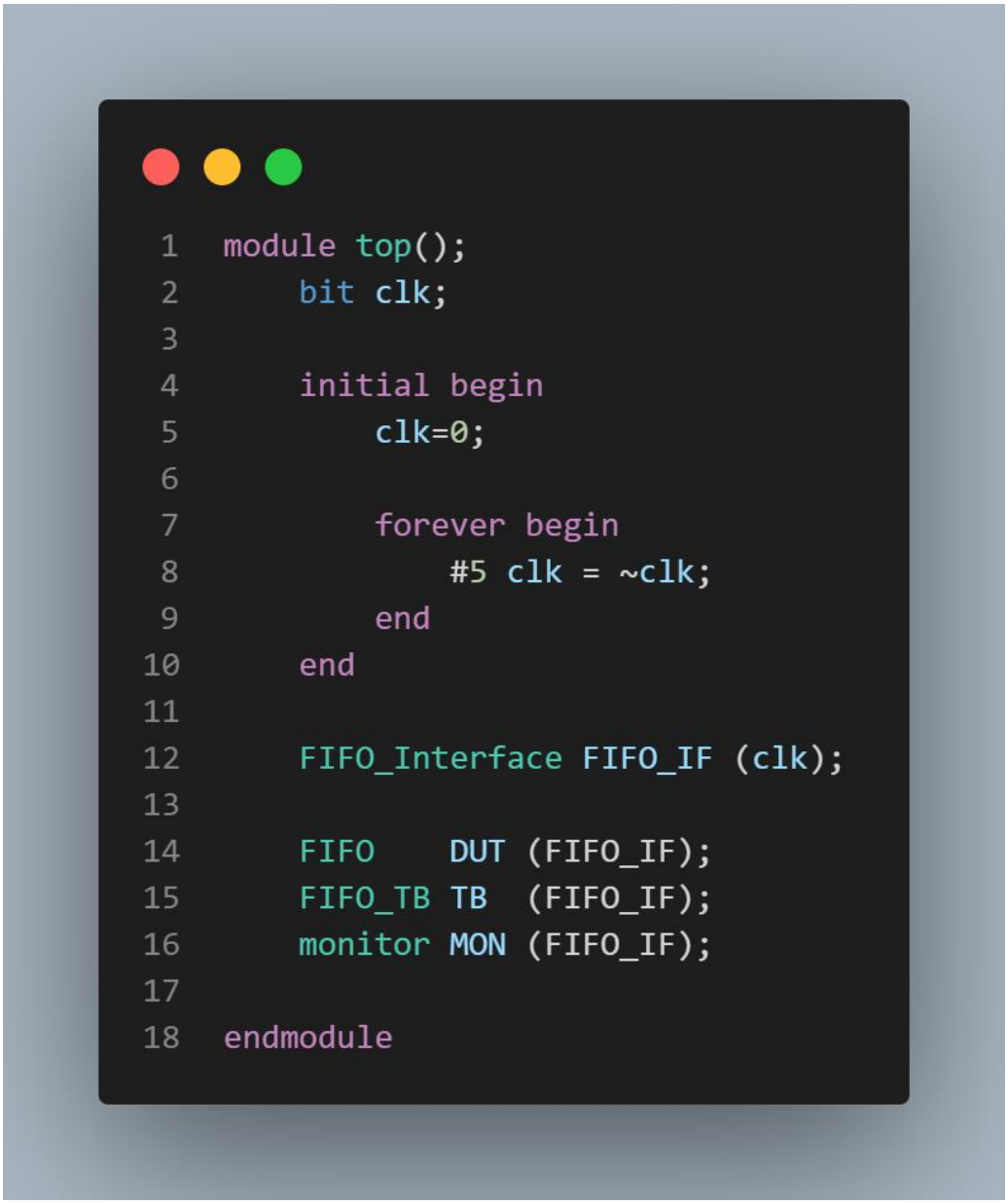
Must make count can deal with these two conditions

And this is the modified version of design after fix all bugs mentioned above.

```
module FIFO(FIFO_Interface.DUT FIFO_IF);

localparam max_fifo_addr = $clog2(FIFO_IF.FIFO_DEPTH);

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

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

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

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

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

assign FIFO_IF.full = (count == FIFO_IF.FIFO_DEPTH)? 1 : 0;
assign FIFO_IF.empty = (count == 0)? 1 : 0;
assign FIFO_IF.almostfull = (count == FIFO_IF.FIFO_DEPTH-1)? 1 : 0; /// modified (-2 => -1)
assign FIFO_IF.almostempty = (count == 1)? 1 : 0;
```

2- top.sv

Here I generate the clk and pass it to the interface then pass the interface to modules (FIFO,FIFO_TB,monitor)



```
1  module top();
2      bit clk;
3
4      initial begin
5          clk=0;
6
7          forever begin
8              #5 clk = ~clk;
9          end
10     end
11
12     FIFO_Interface FIFO_IF (clk);
13
14     FIFO    DUT (FIFO_IF);
15     FIFO_TB TB  (FIFO_IF);
16     monitor MON (FIFO_IF);
17
18 endmodule
```

3- FIFO_TB.sv

Here the tb take the interface as input and import FIFO_Transaction_pkg and SHARED_PKG and make rst task and take_inputs task and repeat 1000 iterations to make code coverage and functional coverage and Assertions coverage be 100% and use test_finished as required.

```
import FIFO_Transaction_pkg::*;
import SHARED_PKG::*;
module FIFO_TB(FIFO_Interface.TEST FIFO_IF);

FIFO_Transaction  FIFO_Transaction_obj;
SHARED_class      SHARED_class_obj;

initial begin
    FIFO_Transaction_obj = new();
    SHARED_class_obj     = new();
    SHARED_class_obj.test_finished=0;

    RST();

    repeat(1000) begin
       assert (FIFO_Transaction_obj.randomize());
       take_inputs(FIFO_Transaction_obj.data_in,FIFO_Transaction_obj.rst_n,FIFO_Transaction_obj.wr_en,FIFO_Transaction_obj.rd_en);
    end

   repeat(2) @(negedge FIFO_IF.clk);
    SHARED_class_obj.test_finished=1;

end
    task take_inputs(
       input [FIFO_IF.FIFO_WIDTH-1:0]data_in,
       input rst_n, wr_en, rd_en
    );

        FIFO_IF.data_in = data_in;
        FIFO_IF.rst_n   = rst_n;
        FIFO_IF.wr_en   = wr_en;
        FIFO_IF.rd_en   = rd_en;
        @(negedge FIFO_IF.clk);

    endtask

    task RST();
    FIFO_IF.rst_n=0;
    assert (FIFO_Transaction_obj.randomize() with {FIFO_Transaction_obj.rst_n==0;});
    take_inputs(FIFO_Transaction_obj.data_in,FIFO_Transaction_obj.rst_n,FIFO_Transaction_obj.wr_en,FIFO_Transaction_obj.rd_en);
    @(negedge FIFO_IF.clk);
    FIFO_IF.rst_n=1;
    @(negedge FIFO_IF.clk);
    endtask


endmodule
```

4- monitor.sv

Here the monitor take interface as input and import all packages and the variable SHARED_class_obj.score_bord_check I will explain it in details in SHARED_PKG section (I did it EXTRA „ EXTRA = = bonus) then assign variables then sample them and check them in parallel as required.

```
import FIFO_scoreboard_pkg::*;
import FIFO_Transaction_pkg::*;
import FIFO_coverage_pkg::*;
import SHARED_PKG::*;
module monitor (FIFO_Interface.MONITOR FIFO_IF);

FIFO_Transaction  FIFO_Transaction_obj;
FIFO_scoreboard   FIFO_scoreboard_obj;
FIFO_coverage     FIFO_coverage_obj;
SHARED_class      SHARED_class_obj;
initial begin
    FIFO_Transaction_obj = new();
    FIFO_scoreboard_obj  = new();
    FIFO_coverage_obj    = new();
    SHARED_class_obj     = new();

   //EXTRA
    /*0 mean in score board the data_out only will be checked as required and all other flags will checked only
     by assertions
     1 mean data_out and all falgs will check in scoreboard and in assertions also (extra) */
    SHARED_class_obj.score_bord_check=1;

    @(negedge FIFO_IF.clk);
    forever begin
      @(negedge FIFO_IF.clk);
      FIFO_Transaction_obj.data_in     = FIFO_IF.data_in;
      FIFO_Transaction_obj.rst_n       = FIFO_IF.rst_n;
      FIFO_Transaction_obj.rd_en       = FIFO_IF.rd_en;
      FIFO_Transaction_obj.wr_en       = FIFO_IF.wr_en;
      FIFO_Transaction_obj.data_out    = FIFO_IF.data_out;
      FIFO_Transaction_obj.wr_ack      = FIFO_IF.wr_ack;
      FIFO_Transaction_obj.overflow    = FIFO_IF.overflow;
      FIFO_Transaction_obj.underflow   = FIFO_IF.underflow;
      FIFO_Transaction_obj.full        = FIFO_IF.full;
      FIFO_Transaction_obj.empty       = FIFO_IF.empty;
      FIFO_Transaction_obj.almostfull  = FIFO_IF.almostfull;
      FIFO_Transaction_obj.almostempty = FIFO_IF.almostempty;

      fork
        //sample_data
        begin
           FIFO_coverage_obj.sample_data(FIFO_Transaction_obj);
        end
       //check_data
        begin
           FIFO_scoreboard_obj.check_data(FIFO_Transaction_obj);
        end
      join

        if(SHARED_class_obj.test_finished) begin
            $display("Correct_count = %d ,,, Error_count = %d",SHARED_class_obj.correct_count,SHARED_class_obj.error_count);
            $stop;
        end
    end
end

endmodule
```

3- Interface

```
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,underflow;
logic full, empty, almostfull, almostempty ;

modport DUT (
input data_in,
input clk, rst_n, wr_en, rd_en,

output data_out,
output wr_ack, overflow,underflow,
output full, empty, almostfull, almostempty
);

modport TEST (
output data_in,
output rst_n, wr_en, rd_en,

input clk,
input data_out,
input wr_ack, overflow,underflow,
input full, empty, almostfull, almostempty
);

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

4- Packages and Classes

1- SHARED_PKG (SHARED_class)

Here I did error_count , correct_count and test_finished as required but I add score_board_check and I explained in details why I added it in screen.

(I did it EXTRA ,, EXTRA = = bonus)

```
1  package SHARED_PKG;
2
3  class SHARED_class;
4      static int error_count , correct_count;
5      static bit test_finished ;
6
7  /* //////////////////          EXTRA            ////////////////////////////////
8     Here to control if all outputs will be checked in score_board or no
9     because the require is to check data out only in score_board and all other flags by assertions
10    so i will make score_bord_check = 0 the score board will check data_out only and other flags will be
11    checked by assertions
12    if score_bord_check = 1 all outputs will be checked in score board and all flags will be checked also
13    in assertions
14 */
15     static bit score_board_check ;
16
17 endclass
18
19 endpackage
```

2- FIFO_Transaction_pkg (FIFO_Transaction)

Here I did all constraints as required

```
package FIFO_Transaction_pkg;

class FIFO_Transaction;

parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

rand bit [FIFO_WIDTH-1:0] data_in;
rand bit rst_n, wr_en, rd_en;

bit [FIFO_WIDTH-1:0] data_out;
bit wr_ack, overflow,underflow;
bit full, empty, almostfull, almostempty ;

int RD_EN_ON_DIST,WR_EN_ON_DIST;

function new( input int RD_EN_ON_DIST=30,WR_EN_ON_DIST=70 );

   this.RD_EN_ON_DIST=RD_EN_ON_DIST ;
   this.WR_EN_ON_DIST=WR_EN_ON_DIST;

endfunction

constraint RST {
    rst_n dist {0:=2, 1:=98};
}

constraint write_enable_dist {
    wr_en dist {1:=WR_EN_ON_DIST, 0:=(100-WR_EN_ON_DIST)};
}

constraint read_enable_dist {
    rd_en dist {1:=RD_EN_ON_DIST, 0:=(100-RD_EN_ON_DIST)};
}

endclass
endpackage
```

3- FIFO_Coverage_pkg (FIFO_Coverage)

Here I covered all required coverpoints and I added one more coverpoint for rst (I did it EXTRA „ EXTRA = = bonus)

```
package FIFO_coverage_pkg;
import FIFO_Transaction_pkg::*;

class FIFO_coverage;
    FIFO_Transaction F_cvg_txn;

    covergroup covgrp;
        // EXTRA will cover rst also
        RST                    : coverpoint  F_cvg_txn.rst_n;

        wr_en_rd_en_wr_ack      : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.wr_ack       iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_overflow    : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.overflow     iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_underflow   : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.underflow    iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_full        : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.full         iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_empty       : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.empty        iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_almostfull  : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.almostfull   iff (F_cvg_txn.rst_n) ; // only if rst deasseted
        wr_en_rd_en_almostempty : cross F_cvg_txn.wr_en , F_cvg_txn.rd_en , F_cvg_txn.almostempty  iff (F_cvg_txn.rst_n) ; // only if rst deasseted

    endgroup

    function void sample_data(input FIFO_Transaction F_txn);
        F_cvg_txn=F_txn;
        covgrp.start();
        covgrp.sample();
        covgrp.stop();
    endfunction

    function new();
    covgrp=new();
    endfunction
endclass

endpackage
```

4- FIFO_Scoreboard_pkg (FIFO_Scoreboard)

Here I check data_out as required and used queue to be as a FIFO (modeling(reference_model))

Also I did 7 checks EXRTA, I checked (full , empty , almostfull , almostempty ,overflow ,underflow ,wr_ack)

(I did it EXTRA ,, EXTRA = = bonus)

```
1  package FIFO_scoreboard_pkg;
2
3  import FIFO_Transaction_pkg::*;
4  import SHARED_PKG::*;
5
6  class FIFO_scoreboard ;
7
8  parameter FIFO_WIDTH = 16;
9  parameter FIFO_DEPTH = 8;
10
11 bit [FIFO_WIDTH-1:0] data_out_ref;
12 bit wr_ack_ref, overflow_ref,underflow_ref;
13 bit full_ref, empty_ref, almostfull_ref, almostempty_ref ;
14
15 SHARED_class shared_obj;
16
17 int queue_check[$];
```

```
function void check_data(input FIFO_Transaction  FIFO_Transaction_chech_data);

    Reference_model(FIFO_Transaction_chech_data);

if (FIFO_Transaction_chech_data.rd_en && !FIFO_Transaction_chech_data.empty) begin

    if(FIFO_Transaction_chech_data.data_out == data_out_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, data_out the expected is %h and the actual is %h",data_out_ref,FIFO_Transaction_chech_data.data_out);
        $display("************************************************");

    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in data_out the expected is %h and the actual is %h",data_out_ref,FIFO_Transaction_chech_data.data_out);
        $display("************************************************");
    end

end

if(shared_obj.score_board_check) begin

    if(FIFO_Transaction_chech_data.wr_ack == wr_ack_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, wr_ack the expected is %b and the actual is %b",wr_ack_ref,FIFO_Transaction_chech_data.wr_ack);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in wr_ack the expected is %b and the actual is %b",wr_ack_ref,FIFO_Transaction_chech_data.wr_ack);
        $display("************************************************");
    end

    if(FIFO_Transaction_chech_data.overflow == overflow_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, overflow the expected is %b and the actual is %b",overflow_ref,FIFO_Transaction_chech_data.overflow);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in overflow the expected is %b and the actual is %b",overflow_ref,FIFO_Transaction_chech_data.overflow);
        $display("************************************************");
    end

    if(FIFO_Transaction_chech_data.underflow == underflow_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, underflow the expected is %b and the actual is %b",underflow_ref,FIFO_Transaction_chech_data.underflow);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in underflow the expected is %b and the actual is %b",underflow_ref,FIFO_Transaction_chech_data.underflow);
        $display("************************************************");
    end

   if(FIFO_Transaction_chech_data.almostempty == almostempty_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, almostempty the expected is %b and the actual is %b",almostempty_ref,FIFO_Transaction_chech_data.almostempty);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in almostempty the expected is %b and the actual is %b",almostempty_ref,FIFO_Transaction_chech_data.almostempty);
        $display("************************************************");
    end

    if(FIFO_Transaction_chech_data.empty == empty_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, empty the expected is %b and the actual is %b",empty_ref,FIFO_Transaction_chech_data.empty);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in empty the expected is %b and the actual is %b",empty_ref,FIFO_Transaction_chech_data.empty);
        $display("************************************************");
    end

    if(FIFO_Transaction_chech_data.almostfull == almostfull_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, almostfull the expected is %b and the actual is %b",almostfull_ref,FIFO_Transaction_chech_data.almostfull);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in almostfull the expected is %b and the actual is %b",almostfull_ref,FIFO_Transaction_chech_data.almostfull);
        $display("************************************************");
    end

     if(FIFO_Transaction_chech_data.full == full_ref) begin
        shared_obj.correct_count = shared_obj.correct_count+1;
        $display("************************************************");
        $display("Succeeded,, full the expected is %b and the actual is %b",full_ref,FIFO_Transaction_chech_data.full);
        $display("************************************************");
    end
    else begin
        shared_obj.error_count = shared_obj.error_count+1;
        $display("************************************************");
        $display("there is error in full the expected is %b and the actual is %b",full_ref,FIFO_Transaction_chech_data.full);
        $display("************************************************");
    end
end

endfunction
```

```
function void Reference_model(input FIFO_Transaction  FIFO_Transaction_Reference_model);

    if (!FIFO_Transaction_Reference_model.rst_n) begin
        wr_ack_ref <= 0;
        overflow_ref <= 0;
    end
    else if (FIFO_Transaction_Reference_model.wr_en && !full_ref) begin
        queue_check.push_front(FIFO_Transaction_Reference_model.data_in);
        wr_ack_ref <= 1;
        overflow_ref <= 0;
    end
    else begin
        wr_ack_ref <= 0;
        if (full_ref & FIFO_Transaction_Reference_model.wr_en)
            overflow_ref <= 1;
        else
            overflow_ref <= 0;
    end

    if (!FIFO_Transaction_Reference_model.rst_n) begin
        underflow_ref<=0;
        queue_check.delete();
    end
    else if (FIFO_Transaction_Reference_model.rd_en && !empty_ref) begin
        data_out_ref <= queue_check.pop_back();
        underflow_ref<=0;
    end
    else begin
        if (empty_ref && FIFO_Transaction_Reference_model.rd_en) begin
            underflow_ref <=1;
        end
        else begin
            underflow_ref <=0;
        end
    end


    full_ref        <= (queue_check.size() == FIFO_Transaction_Reference_model.FIFO_DEPTH)? 1 : 0;
    empty_ref       <= (queue_check.size() == 0)? 1 : 0;
    almostfull_ref  <= (queue_check.size() == FIFO_Transaction_Reference_model.FIFO_DEPTH-1)? 1 : 0;
    almostempty_ref <= (queue_check.size() == 1)? 1 : 0;


endfunction

function new();
    full_ref        = 0;
    empty_ref       = 1;
    almostfull_ref  = 0;
    almostempty_ref = 0;
    wr_ack_ref      = 0;
    underflow_ref   = 0;
    overflow_ref    = 0;
endfunction
endclass

endpackage
```

5- Assertions

I added Assertions in design module (FIFO.sv) and : Guard the assertions using conditional compilation with the `ifdef

b. Extra part to be done: Guard the assertions using conditional compilation with the `ifdef directive with macro named SIM, and then include the macro in the vlog command +define+SIM option in the vlog command of your do file. Refer to this link to learn more about conditional compilation.

I did the extra part (I did it EXTRA ,, EXTRA = = bonus) 🤷‍♀️

The required Assertions are outputs flags and the counter only

Guess what?!! I did extra Assertions

Assertions: -

1- 1 Assertion for max_fifo_addr (EXTRA)
2- 2 Assertions for wr_ptr (EXTRA)
3- 2 Assertions for rd_ptr (EXTRA)
4- 5 Assertions for Count
5- 3 Assertions for wr_ack
6- 3 Assertions for overflow
7- 3 Assertions for underflow
8- 2 Assertions for full
9- 2 Assertions for almostfull
10- 2 Assertions for Relation between full and almostfull (EXTRA)
11- 2 Assertions for empty
12- 2 Assertions for almostempty
13- 2 Assertions for Relation between empty and almostempty (EXTRA)

```
`ifdef SIM
//EXTA
// max_fifo_addr Assertions
always_comb begin
    max_fifo_addr_p: assert final (max_fifo_addr == $clog2(FIFO_IF.FIFO_DEPTH));
end
//EXTA
// wr_ptr Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    wr_ptr1_p: assert final (wr_ptr == 0);
end

property wr_ptr2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n)  (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH) |-> ##1 (wr_ptr == ($past(wr_ptr)+1'b1));
endproperty
wr_ptr2_p: assert property (wr_ptr2);
wr_ptr2_c: cover  property (wr_ptr2);
//EXTA
// rd_ptr Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    rd_ptr1_p: assert final (rd_ptr == 0);
end

property rd_ptr2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n)  (FIFO_IF.rd_en && count != 0) |-> ##1 (rd_ptr == ($past(rd_ptr)+1'b1));
endproperty
rd_ptr2_p: assert property (rd_ptr2);
rd_ptr2_c: cover  property (rd_ptr2);
```

```
// Counter Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    counter1_p: assert final (count == 0);
end

property counter2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b10) && !FIFO_IF.full) |-> ##1 (count == ($past(count)+1'b1))
endproperty
counter2_p: assert property (counter2);
counter2_c: cover  property (counter2);


property counter3;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b01) && !FIFO_IF.empty) |-> ##1 (count == ($past(count)-1'b1))
endproperty
counter3_p: assert property (counter3);
counter3_c: cover  property (counter3);

property counter4;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.full) |-> ##1 (count == ($past(count)-1'b1))
endproperty
counter4_p: assert property (counter4);
counter4_c: cover  property (counter4);

property counter5;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (({FIFO_IF.wr_en, FIFO_IF.rd_en} == 2'b11) && FIFO_IF.empty) |-> ##1 (count == ($past(count)+1'b1))
endproperty
counter5_p: assert property (counter5);
counter5_c: cover  property (counter5);
```

```
// wr_ack Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    wr_ack1_p: assert final (FIFO_IF.wr_ack == 0);
end

property wr_ack2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.wr_en && count < FIFO_IF.FIFO_DEPTH)  |-> ##1 (FIFO_IF.wr_ack == 1);
endproperty
wr_ack2_p: assert property (wr_ack2);
wr_ack2_c: cover  property (wr_ack2);

property wr_ack3;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (!FIFO_IF.wr_en || !(count < FIFO_IF.FIFO_DEPTH))  |-> ##1 (FIFO_IF.wr_ack == 0);
endproperty
wr_ack3_p: assert property (wr_ack3);
wr_ack3_c: cover  property (wr_ack3);
```

```
/overflow Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    overflow1_p: assert final (FIFO_IF.overflow == 0);
end

property overflow2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.full && FIFO_IF.wr_en) |-> ##1 (FIFO_IF.overflow == 1);
endproperty
overflow2_p: assert property (overflow2);
overflow2_c: cover  property (overflow2);

property overflow3;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) ((!FIFO_IF.full) && FIFO_IF.wr_en) |-> ##1 (FIFO_IF.overflow == 0);
endproperty
overflow3_p: assert property (overflow3);
overflow3_c: cover  property (overflow3);
```

```
//underflow Assertions
//immediate because the rst is async
always_comb begin
    if(!FIFO_IF.rst_n)
    underflow1_p: assert final (FIFO_IF.underflow == 0);
end

property underflow2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.empty && FIFO_IF.rd_en) |-> ##1 (FIFO_IF.underflow == 1);
endproperty
underflow2_p: assert property (underflow2);
underflow2_c: cover  property (underflow2);

property underflow3;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) ((!FIFO_IF.empty) && FIFO_IF.rd_en) |-> ##1 (FIFO_IF.underflow == 0);
endproperty
underflow3_p: assert property (underflow3);
underflow3_c: cover  property (underflow3);

```

```
// full Assertions
property full1;
 @(posedge FIFO_IF.clk)  (count == FIFO_IF.FIFO_DEPTH)  |->  (FIFO_IF.full == 1);
endproperty
full1_p: assert property (full1);
full1_c: cover  property (full1);

property full2;
 @(posedge FIFO_IF.clk)  (count != FIFO_IF.FIFO_DEPTH)  |->  (FIFO_IF.full == 0);
endproperty
full2_p: assert property (full2);
full2_c: cover  property (full2);

// almostfull Assertions
property almostfull1;
 @(posedge FIFO_IF.clk)  (count == (FIFO_IF.FIFO_DEPTH-1'b1))  |->  (FIFO_IF.almostfull == 1);
endproperty
almostfull1_p: assert property (almostfull1);
almostfull1_c: cover  property (almostfull1);

property almostfull2;
 @(posedge FIFO_IF.clk)  (count != (FIFO_IF.FIFO_DEPTH-1'b1))  |->  (FIFO_IF.almostfull == 0);
endproperty
almostfull2_p: assert property (almostfull2);
almostfull2_c: cover  property (almostfull2);
//EXTRA
// compination assertions between full and almostfull
property full_almostfull1;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.almostfull && (!(FIFO_IF.rd_en)&&(FIFO_IF.wr_en)) )  |-> ##1  (FIFO_IF.full);
endproperty
full_almostfull1_p: assert property (full_almostfull1);
full_almostfull1_c: cover  property (full_almostfull1);


property full_almostfull2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.full && ((FIFO_IF.rd_en)&&(!FIFO_IF.wr_en)) )  |-> ##1  (FIFO_IF.almostfull);
endproperty
full_almostfull2_p: assert property (full_almostfull2);
full_almostfull2_c: cover  property (full_almostfull2);
```

```
// empty Assertions
property empty1;
 @(posedge FIFO_IF.clk)  (count == 0)  |->  (FIFO_IF.empty == 1);
endproperty
empty1_p: assert property (empty1);
empty1_c: cover  property (empty1);

property empty2;
 @(posedge FIFO_IF.clk)  (count != 0)  |->  (FIFO_IF.empty == 0);
endproperty
empty2_p: assert property (empty2);
empty2_c: cover  property (empty2);


// almostempty Assertions
property almostempty1;
 @(posedge FIFO_IF.clk)  (count == 1)  |->  (FIFO_IF.almostempty == 1);
endproperty
almostempty1_p: assert property (almostempty1);
almostempty1_c: cover  property (almostempty1);

property almostempty2;
 @(posedge FIFO_IF.clk)  (count != 1)  |->  (FIFO_IF.almostempty == 0);
endproperty
almostempty2_p: assert property (almostempty2);
almostempty2_c: cover  property (almostempty2);

//EXTRA
// compination assertions between empty and almostempty
property empty_almostempty1;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.almostempty && (!(FIFO_IF.wr_en)&& (FIFO_IF.rd_en)) )  |-> ##1  (FIFO_IF.empty);
endproperty
empty_almostempty1_p: assert property (empty_almostempty1);
empty_almostempty1_c: cover  property (empty_almostempty1);


property empty_almostempty2;
 @(posedge FIFO_IF.clk) disable iff (!FIFO_IF.rst_n) (FIFO_IF.empty && ((FIFO_IF.wr_en)&& (!FIFO_IF.rd_en)) )  |-> ##1  (FIFO_IF.almostempty);
endproperty
empty_almostempty2_p: assert property (empty_almostempty2);
empty_almostempty2_c: cover  property (empty_almostempty2);

`endif

endmodule

```

6- DO File

```
vlib work
vlog -f sourcefile.txt +cover -covercells +define+SIM
vsim -voptargs=+acc work.top -cover +define+SIM

add wave -position insertpoint sim:/top/DUT/FIFO_IF/*
add wave -position insertpoint sim:/top/DUT/*

add wave /top/DUT/max_fifo_addr_p /top/DUT/wr_ptr1_p /top/DUT/wr_ptr2_p
add wave /top/DUT/rd_ptr1_p /top/DUT/rd_ptr2_p /top/DUT/counter1_p
add wave /top/DUT/counter2_p /top/DUT/counter3_p /top/DUT/counter4_p /top/DUT/counter5_p
add wave /top/DUT/wr_ack1_p /top/DUT/wr_ack2_p /top/DUT/wr_ack3_p /top/DUT/overflow1_p
add wave /top/DUT/overflow2_p /top/DUT/overflow3_p /top/DUT/underflow1_p /top/DUT/underflow2_p
add wave /top/DUT/underflow3_p /top/DUT/full1_p /top/DUT/full2_p /top/DUT/almostfull1_p
add wave /top/DUT/almostfull2_p /top/DUT/full_almostfull1_p /top/DUT/full_almostfull2_p
add wave /top/DUT/empty1_p /top/DUT/empty2_p /top/DUT/almostempty1_p /top/DUT/almostempty2_p
add wave /top/DUT/empty_almostempty1_p /top/DUT/empty_almostempty2_p

coverage save top.ucdb -onexit
run -all

#vcover report top.ucdb -details -annotate -all -output code_coverage_rpt.txt

```

7- Code Coverage

Toggle Coverage: -

```
Toggle Coverage      =     100.00% (20 of 20 bins)
```

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

Statements Coverage: -

```
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      36        36         0   100.00%
```

Branch Coverage: -

```
  Enabled Coverage              Bins      Hits    Misses  Coverage
  ----------------              ----      ----    ------  --------
  Branches                        37        37         0   100.00%
```

Also, I will attach the complete report in zip file if you want to see more details about coverage.

8- Functional Coverage

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

```
Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        29        na        na        na
            Covergroup Bins       100       100         0   100.00%
```

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_coverage_pkg/FIFO_coverage		100.00%							
TYPE covgrp		100.00%	100	100.00...		✓	auto(1)		
CVP covgrp::RST		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__0#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__1#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.almostempty__2#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__3#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__4#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.almostfull__5#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__6#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__7#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.empty__8#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__9#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__10#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.full__11#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__12#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__13#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.underflow__14#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__15#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__16#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.overflow__17#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_en__18#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.rd_en__19#}		100.00%	100	100.00...		✓			
CVP covgrp::{#F_cvg_txn.wr_ack__20#}		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_wr_ack		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_overflow		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_underflow		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_full		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_empty		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_almostfull		100.00%	100	100.00...		✓			
CROSS covgrp::wr_en_rd_en_almostempty		100.00%	100	100.00...		✓			

Also, I will attach the complete report in zip file if you want to see more details about coverage.

9- Assertions Coverage

```
Directive Coverage:
    Directives                    24        24         0   100.00%
```

```
 TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 24
```

```
Assertion Coverage:
    Assertions                    31        31         0   100.00%
```

Name	Language	Enabled	Log	Count	AtLe	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/wr_ptr2_c	SVA	✓	Off	396	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/rd_ptr2_c	SVA	✓	Off	281	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/counter2_c	SVA	✓	Off	274	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/counter3_c	SVA	✓	Off	88	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/counter4_c	SVA	✓	Off	77	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/counter5_c	SVA	✓	Off	6	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/wr_ack2_c	SVA	✓	Off	396	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/wr_ack3_c	SVA	✓	Off	576	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/overflow2_c	SVA	✓	Off	288	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/overflow3_c	SVA	✓	Off	396	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/underflow2_c	SVA	✓	Off	10	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/underflow3_c	SVA	✓	Off	281	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/full1_c	SVA	✓	Off	403	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/full2_c	SVA	✓	Off	601	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/almostfull1_c	SVA	✓	Off	270	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/almostfull2_c	SVA	✓	Off	734	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/full_almostfull1_c	SVA	✓	Off	130	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/full_almostfull2_c	SVA	✓	Off	120	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/empty1_c	SVA	✓	Off	43	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/empty2_c	SVA	✓	Off	961	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/almostempty1_c	SVA	✓	Off	38	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/almostempty2_c	SVA	✓	Off	966	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/empty_almostempty1_c	SVA	✓	Off	3	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/empty_almostempty2_c	SVA	✓	Off	19	1	Unlimited	1	100%		✓	0	0	0 ns	0

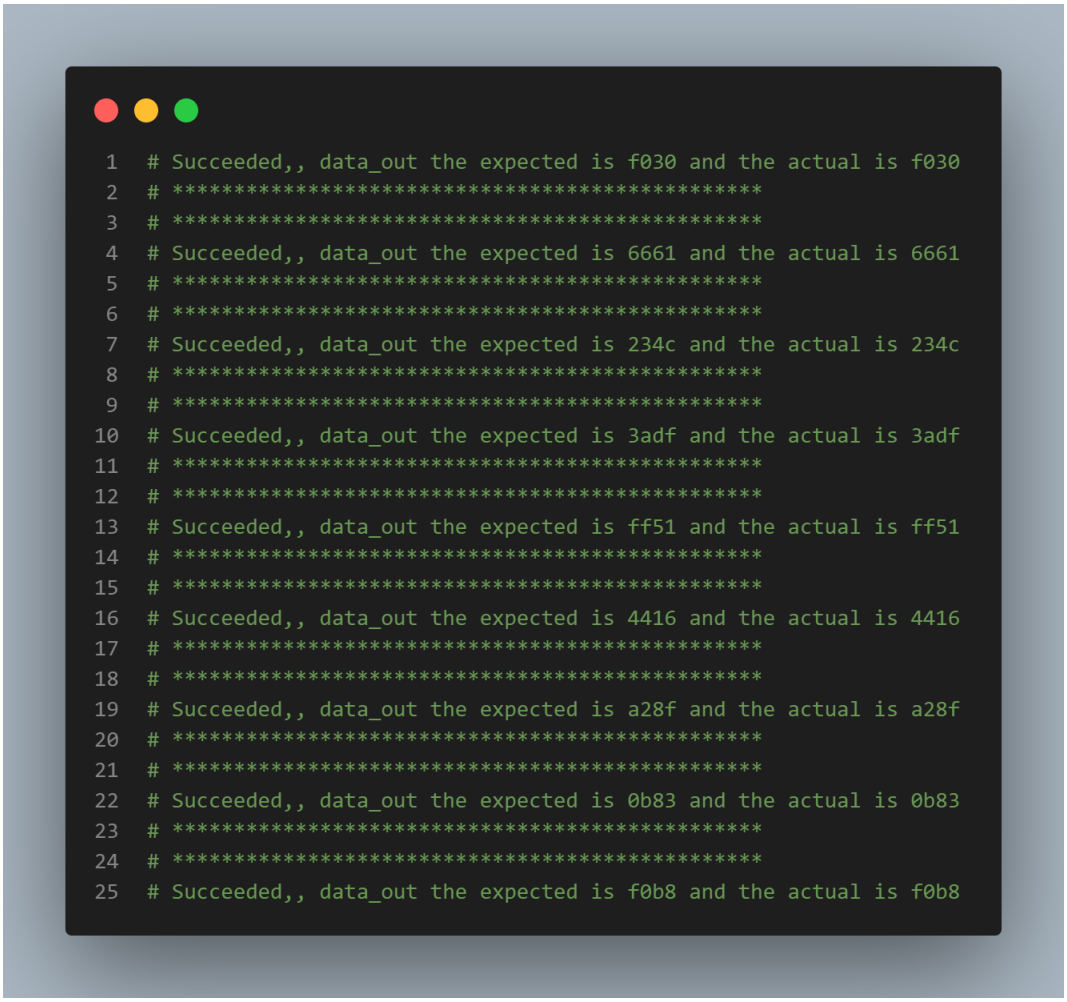
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/top/DUT/max_fifo_addr_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (3==$clog2(FIFO_IF.FIFO_...	✓
/top/DUT/wr_ptr1_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (wr_ptr==0)	✓
/top/DUT/wr_ptr2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/rd_ptr1_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (rd_ptr==0)	✓
/top/DUT/rd_ptr2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/counter1	Immediate	SVA	on	0	1	-	-	-	-		off	assert (count==0)	✓
/top/DUT/counter2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/counter3_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/counter4_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/counter5_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/wr_ack1_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~FIFO_IF.wr_ack)	✓
/top/DUT/wr_ack2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/wr_ack3_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/overflow1_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~FIFO_IF.overflow)	✓
/top/DUT/overflow2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/overflow3_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/underflow1_c	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~FIFO_IF.underflow)	✓
/top/DUT/underflow2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/underflow3_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/full1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/full2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/almostfull1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/almostfull2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/full_almostfull1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/full_almostfull2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/empty1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/empty2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/almostempty1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/almostempty2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) (c...	✓
/top/DUT/empty_almostempty1_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/DUT/empty_almostempty2_p	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge FIFO_IF.clk) di...	✓
/top/TB/#ublk#182146242#15/immed__16	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/TB/RST/immed__39	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓

Also, I will attach the complete report in zip file if you want to see more details about coverage.

10- Simulation

Some of transcript when score_board_check=0

```
# Correct_count = 288 ,,, Error_count = 0
```



```
# Succeeded,, data_out the expected is f030 and the actual is f030
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is 6661 and the actual is 6661
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is 234c and the actual is 234c
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is 3adf and the actual is 3adf
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is ff51 and the actual is ff51
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is 4416 and the actual is 4416
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is a28f and the actual is a28f
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is 0b83 and the actual is 0b83
# ************************************************
# ************************************************
# Succeeded,, data_out the expected is f0b8 and the actual is f0b8
```

Some of transcript when score_board_check=1

```
# ************************************************
# Correct_count = 7316 ,,, Error_count = 0
```

```
# Succeeded,, data_out the expected is a28f and the actual is a28f
# ***********************************************
# ***********************************************
# Succeeded,, wr_ack the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, overflow the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, underflow the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, almostempty the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, empty the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, almostfull the expected is 1 and the actual is 1
# ***********************************************
# ***********************************************
# Succeeded,, full the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, data_out the expected is 0b83 and the actual is 0b83
# ***********************************************
# ***********************************************
# Succeeded,, wr_ack the expected is 1 and the actual is 1
# ***********************************************
# ***********************************************
# Succeeded,, overflow the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, underflow the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, almostempty the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, empty the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, almostfull the expected is 0 and the actual is 0
# ***********************************************
# ***********************************************
# Succeeded,, full the expected is 1 and the actual is 1
# ***********************************************
# ***********************************************
```

Some of Waveform with Assertions