FIFO VERIFICATION PROJECT

Hussien Mohamed Hussien

Description about fifo:

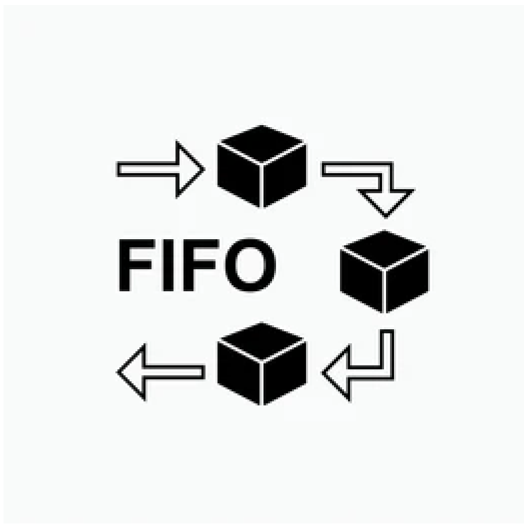
A FIFO (First In, First Out) buffer is a type of data structure used in digital systems to manage and store data. It operates based on the principle that the first data item entered into the buffer is the first one to be removed, resembling a queue. FIFOs are commonly used in hardware designs and digital circuits to handle data transfer between two systems or components that operate at different speeds, ensuring smooth data flow without loss.

Functionality:

- **Data Storage**: A FIFO holds data in a sequential manner, storing it in the order it was received.
- **Controlled Flow**: The buffer maintains two pointers—one for the write (input) side and one for the read (output) side. Data is written into the buffer at the write pointer and read out at the read pointer.
- **Synchronization**: In systems with clock domain crossing, FIFOs help by storing data temporarily until it can be safely passed to a different clock domain.
- **Overflow and Underflow Protection**: FIFOs can manage situations where data input exceeds the available space (overflow) or when data is being read from an empty buffer (underflow), usually by asserting status flags or error signals.

Usage:

- **Data Communication**: FIFOs are essential in communication systems, especially for data buffering between devices that produce and consume data at different rates, such as processors and peripheral devices.
- **Pipelining**: In pipeline architectures, FIFOs enable data buffering between processing stages.
- **Clock Domain Crossing**: FIFOs are used to safely transfer data between circuits operating under different clock frequencies.
- **Audio and Video Processing**: In multimedia applications, FIFOs help in managing streams of audio or video data, ensuring a smooth flow from one processing stage to another.



Design before correction :

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(inter.DUT intt);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
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 intt.clk or negedge intt.rst_n) begin
    if (!intt.rst_n) begin
        wr_ptr <= 0;
    end
    else if (intt.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= intt.data_in;
        intt.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        intt.wr_ack <= 0;
        if (intt.full & intt.wr_en)
            intt.overflow <= 1;
        else
            intt.overflow <= 0;
    end
end

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

always @(posedge intt.clk or negedge intt.rst_n) begin
    if (!intt.rst_n) begin
        count <= 0;
    end
    else begin
```

```
        if  ( ({intt.wr_en, intt.rd_en} == 2'b10) && !intt.full)
            count <= count + 1;
        else if ( ({intt.wr_en, intt.rd_en} == 2'b01) && !intt.empty)
            count <= count - 1;
    end
end

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

endmodule
```

Verification plan :

Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
FIFO_1	the reset is asserted and all the outputs should be asserted to zero except the empty flag	directed at the start of the simulation	cross coverage between 3 signals which are write enable, read enable and each output control signals (outputs except data_out) to make sure that all combinations of write and read enable took place in all state of the FIFO.	function checker at the score class
FIFO_2	the wr_en is high while the rd_en is low 9 cycles while the data_in is randomized the values of the data_in will be stored in a queue then the wr_en is low and the rd_en is high 9 cycles and data out will be checked with the values in the queue this scenario wil be repeated 5000 times	randomized during the simulation	cross coverage between 3 signals which are write enable, read enable and each output control signals (outputs except data_out) to make sure that all combinations of write and read enable took place in all state of the FIFO.	function checker at the score class
FIFO_3	all inputs will be randomized 5000 times under the following constraints 1. Assert reset less often 2. Constraint the write enable to be high with distribution of the value WR_EN_ON_DIST(70) and to be low with 100-WR_EN_ON_DIST 3. Constraint the read enable the same as write enable but using RD_EN_ON_DIST(30)	randomized during the simulation	cross coverage between 3 signals which are write enable, read enable and each output control signals (outputs except data_out) to make sure that all combinations of write and read enable took place in all state of the FIFO.	function checker at the score class

Interface :

```
interface inter(clk);
    parameter FIFO_WIDTH = 16;
    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 DUT (
    input clk,data_in,rst_n,wr_en,rd_en,
    output data_out,wr_ack,overflow,full,empty,almostfull,almostempty,underflow
    );
    modport TB (
    input clk,rd_en,data_out,wr_ack,overflow,full,empty,almostfull,almostempty,underflow,
    output data_in,rst_n,wr_en
    );
    modport MONITOR (
    input
clk,rd_en,data_out,wr_ack,overflow,full,empty,almostfull,almostempty,underflow,data_in,rst_n,
wr_en
    );
endinterface //inter
```

classes that will be used :

Transaction class :

```
package trans;
class FIFO_transaction;
    parameter FIFO_WIDTH = 16;
    rand logic [FIFO_WIDTH-1:0] data_in;
    rand logic  rst_n, wr_en, rd_en;
    logic clk;
    logic [FIFO_WIDTH-1:0] data_out;
    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;
    integer RD_EN_ON_DIST ,  WR_EN_ON_DIST   ;
    function new(int rd_dst = 30 , wr_dst = 70 );
        this.RD_EN_ON_DIST=rd_dst;
        this.WR_EN_ON_DIST=wr_dst;
    endfunction //new()
    constraint rando{
        rst_n dist {1:=90,0:=10};
        wr_en dist {1:=(WR_EN_ON_DIST),0:=(100-WR_EN_ON_DIST)};
        rd_en dist {1:=(RD_EN_ON_DIST),0:=(100-RD_EN_ON_DIST)};
    }
endclass //
endpackage
```

Scoreboard class :

```
package score;
    import trans::*;
    class FIFO_scoreboard;
        parameter FIFO_WIDTH = 16;
        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;
        static integer error_count=0 , right_count=0 , size ;
        logic [FIFO_WIDTH-1:0] data_mem [$] ;
        static bit test_finished;
        bit next_write,next_read;
        task check_data (FIFO_transaction trans1);
            reference_model(trans1);
            if (!trans1.rst_n) begin
                if (trans1.wr_ack!=wr_ack_ref || trans1.overflow!=overflow_ref || trans1.full
!= full_ref || trans1.empty != empty_ref || trans1.almostfull != almostfull_ref ||
trans1.almostempty != almostempty_ref) begin
                error_count=error_count+1;
                $display("error at reset");
                if (trans1.wr_ack!=wr_ack_ref) begin
                $display("wr_ack=%0b , wr_ack_ref=%0b at
%0t",trans1.wr_ack,wr_ack_ref,$time);
                end
                if (trans1.overflow!=overflow_ref) begin
                $display("overflow=%0b , overflow_ref=%0b at
%0t",trans1.overflow,overflow_ref,$time);
                end
                if (trans1.full!=full_ref) begin
                $display("full=%0b , full_ref=%0b at %0t",trans1.full,full_ref,$time);
                end
                if (trans1.empty!=empty_ref) begin
                $display("empty=%0b , empty_ref=%0b at
%0t",trans1.empty,empty_ref,$time);
                end
                if (trans1.almostfull!=almostfull_ref) begin
                $display("full=%0b , full=%0b at %0t",trans1.full,full_ref,$time);
                end
                if (trans1.almostempty!=almostempty_ref) begin
                $display("almostempty=%0b , almostempty_ref=%0b at
%0t",trans1.almostempty,almostempty_ref,$time);
                end
            end
            else begin
                right_count = right_count+1;
            end
            end
            else begin
```

```
                if (trans1.wr_ack!=wr_ack_ref || trans1.overflow!=overflow_ref ||
trans1.data_out!=data_out_ref || trans1.full != full_ref || trans1.empty != empty_ref ||
trans1.almostfull != almostfull_ref || trans1.almostempty != almostempty_ref) begin
                error_count=error_count+1;
                $display("error at no reset");
                if (trans1.wr_ack!=wr_ack_ref) begin
                $display("wr_ack=%0b , wr_ack_ref=%0b , size=%0d at
%0t",trans1.wr_ack,wr_ack_ref,data_mem.size(),$time);
                end
                if (trans1.overflow!=overflow_ref) begin
                $display("overflow=%0b , overflow_ref=%0b at
%0t",trans1.overflow,overflow_ref,$time);
                end
                if (trans1.data_out!=data_out_ref) begin
                $display("dataout=%0d , dataout_ref=%0d, size=%0d at
%0t",trans1.data_out,data_out_ref,data_mem.size(),$time);
                end
                if (trans1.full!=full_ref) begin
                $display("full=%0b , full_ref=%0b at %0t",trans1.full,full_ref,$time);
                end
                if (trans1.empty!=empty_ref) begin
                $display("empty=%0b , empty_ref=%0b , size=%0d at
%0t",trans1.empty,empty_ref,data_mem.size(),$time);
                end
                if (trans1.almostfull!=almostfull_ref) begin
                $display("almostfull=%0b , almostfull_ref=%0b at
%0t",trans1.almostfull,almostfull_ref,$time);
                end
                if (trans1.almostempty!=almostempty_ref) begin
                $display("almostempty=%0b , almostempty_ref=%0b , size=%0d at
%0t",trans1.almostempty,almostempty_ref,data_mem.size(),$time);
                end
            end
            else begin
                right_count = right_count+1;
            end
            end
        endtask

        task reference_model(FIFO_transaction trans2);
        if (!trans2.rst_n) begin
            full_ref=0;
            almostfull_ref=0;
            empty_ref=1;
            almostempty_ref=0;
            overflow_ref=0;
            underflow_ref=0;
            data_mem.delete();
            wr_ack_ref=0;
            next_read=0;
            next_write=0;
        end
        else
        begin
            size=data_mem.size();
            // read
            if (trans2.rd_en && size>0) begin
                data_out_ref=data_mem.pop_front();
            end

             if (trans2.wr_en && size<8) begin
                data_mem.push_back(trans2.data_in);
                wr_ack_ref=1;
            end
            else
            begin
                wr_ack_ref=0;
            end
            if(trans2.wr_en&&next_write)
            begin
                overflow_ref=1;
            end
            else begin
                overflow_ref=0;
            end
            if(trans2.rd_en&&next_read)
            begin
                underflow_ref=1;
            end
            else begin
                underflow_ref=0;
            end
            if (data_mem.size==7) begin
                almostfull_ref=1;
            end
            else begin
                almostfull_ref=0;
            end
            if (data_mem.size==8) begin
                full_ref=1;
                next_write=1;
            end
            else begin
                full_ref=0;
                next_write=0;
            end
            if (data_mem.size==0) begin
                empty_ref=1;
                next_read=1;
            end
            else begin
                empty_ref=0;
                next_read=0;
            end
            if (data_mem.size==1) begin
                almostempty_ref=1;
            end
            else begin
                almostempty_ref=0;
            end
        end
        endtask
    endclass //
endpackage
```

cover class:

```
package coverr;
    import trans::*;
    FIFO_transaction F_cvg_txn = new();
    class FIFO_coverage;
        covergroup p1 ;
            rd_enable: coverpoint F_cvg_txn.rd_en iff(F_cvg_txn.rst_n);
            wr_enable: coverpoint F_cvg_txn.wr_en iff(F_cvg_txn.rst_n);
            wr_acknowledge: coverpoint F_cvg_txn.wr_ack iff(F_cvg_txn.rst_n);
            overfloww: coverpoint F_cvg_txn.overflow iff(F_cvg_txn.rst_n);
            underfloww: coverpoint F_cvg_txn.underflow iff(F_cvg_txn.rst_n);
            fulll: coverpoint F_cvg_txn.full iff(F_cvg_txn.rst_n);
            almostfulll: coverpoint F_cvg_txn.almostfull iff(F_cvg_txn.rst_n);
            emptyy: coverpoint F_cvg_txn.empty iff(F_cvg_txn.rst_n);
            almostemptyy: coverpoint F_cvg_txn.almostempty iff(F_cvg_txn.rst_n);
            rd_enable_with_empty:cross rd_enable,emptyy;
            rd_enable_with_almostempty:cross rd_enable,almostemptyy;
            rd_enable_with_underflow : cross rd_enable,underfloww {
                option.cross_auto_bin_max=0;
                bins rd_on_under_off = binsof(rd_enable) intersect {1} && binsof(underfloww)
intersect {0};
                bins rd_off_under_off = binsof(rd_enable) intersect {0} && binsof(underfloww)
intersect {0};
                bins rd_on_under_on = binsof(rd_enable) intersect {1} && binsof(underfloww)
intersect {1};
            }
            wr_enable_with_full : cross wr_enable,fulll;
            wr_enable_with_almostfull : cross wr_enable,almostfulll;
            wr_enable_with_overflow : cross wr_enable,overfloww{
                option.cross_auto_bin_max=0;
                bins wr_on_over_off = binsof(wr_enable) intersect {1} && binsof(overfloww)
intersect {0};
```

```
                bins wr_off_over_off = binsof(wr_enable) intersect {0} && binsof(overfloww)
intersect {0};
                bins wr_on_over_on = binsof(wr_enable) intersect {1} && binsof(overfloww)
intersect {1};
            }
            wr_enable_with_wr_acknowledge : cross wr_enable,wr_acknowledge{
                option.cross_auto_bin_max=0;
                bins wr_on_wr_acknowledge_off = binsof(wr_enable) intersect {1} &&
binsof(wr_acknowledge) intersect {0};
                bins wr_off_wr_acknowledge_off = binsof(wr_enable) intersect {0} &&
binsof(wr_acknowledge) intersect {0};
                bins wr_on_wr_acknowledge_on = binsof(wr_enable) intersect {1} &&
binsof(wr_acknowledge) intersect {1};
            }
        endgroup

        function new();
            p1=new();
        endfunction //new()

        function void sample_data(FIFO_transaction F_txn);
            F_cvg_txn=F_txn;
            p1.sample();
        endfunction

    endclass //className

endpackage
```

Testbench module :

```
import score::*;
import trans::*;
module fifo_tb (inter.TB intt);
    FIFO_transaction c=new();
    FIFO_scoreboard cc=new();
    initial begin
        intt.rst_n=0;
        intt.data_in=0;
        intt.wr_en=0;
        intt.rd_en=0;
       repeat(2)
        @(negedge intt.clk);

        intt.rst_n=1;
        c.rand_mode(0);
        c.data_in.rand_mode(1);
        c.constraint_mode(0);
        repeat(5000) begin
            intt.wr_en=1;
            intt.rd_en=0;
            repeat(9) begin
                assert (c.randomize);
                intt.data_in=c.data_in;
                @(negedge intt.clk);
            end
            intt.wr_en=0;
            intt.rd_en=1;
            repeat(9) begin
                @(negedge intt.clk);
            end
        end
        c.rand_mode(0);
        c.rand_mode(1);
        c.constraint_mode(1);
        repeat (5000) begin
            assert (c.randomize);
            intt.wr_en=c.wr_en;
            intt.rd_en=c.rd_en;
            intt.data_in=c.data_in;
            intt.rst_n=c.rst_n;
            @(negedge intt.clk);
        end
        cc.test_finished=1;
    end
endmodule
```

Monitor module:

```
import score::*;
import coverr::*;
import trans::*;
module fifo_monitor (inter.MONITOR intt);
    FIFO_transaction obj_trans;
    FIFO_scoreboard obj2_score;
    FIFO_coverage obj3_cover;
    always @(*) begin
        obj_trans.rst_n=intt.rst_n;
            obj_trans.data_in=intt.data_in;
            obj_trans.wr_en=intt.wr_en;
            obj_trans.rd_en=intt.rd_en;
            obj_trans.data_out=intt.data_out;
            obj_trans.full=intt.full;
            obj_trans.almostfull=intt.almostfull;
            obj_trans.empty=intt.empty;
            obj_trans.almostempty=intt.almostempty;
            obj_trans.wr_ack=intt.wr_ack;
            obj_trans.overflow=intt.overflow;
            obj_trans.underflow=intt.underflow;
    end
    initial begin
        obj_trans=new();
        obj2_score=new();
        obj3_cover=new();
        forever begin
            @(negedge intt.clk);
            // obj_trans.rst_n=intt.rst_n;
            // obj_trans.data_in=intt.data_in;
            // obj_trans.wr_en=intt.wr_en;
            // obj_trans.rd_en=intt.rd_en;
            // obj_trans.data_out=intt.data_out;
            // obj_trans.full=intt.full;
            // obj_trans.almostfull=intt.almostfull;
            // obj_trans.empty=intt.almostempty;
            // obj_trans.almostempty=intt.almostempty;
            // obj_trans.wr_ack=intt.wr_ack;
            // obj_trans.overflow=intt.overflow;
            // obj_trans.underflow=intt.underflow;
            fork
                //sample
                begin
                   obj3_cover.sample_data(obj_trans);
                end
                begin
                    obj2_score.check_data(obj_trans);
                end
            join
            if (obj2_score.test_finished) begin
                $display("error_count=%0d
,  right_count=%0d",obj2_score.error_count,obj2_score.right_count);
                $stop;
            end
        end
    end
endmodule
```

Top module :

```
module fifo_top ();
    bit clk;
    initial begin
        forever begin
            #10;
            clk=!clk;
        end
    end
    inter inter(clk);
    FIFO_corrected DUT (inter);
    fifo_tb TB(inter);
    fifo_monitor MONITOR (inter);
endmodule
```

bugs found:

first bug :

Underflow should be sequential not compinational .

second bug :

Almostfull flag should be high at fifo_depth-2.

third bug :

reset isn’t applied on the following flags wr_ack,overflow,underflow.

fourth bug :

Underflow and overflow should be 0 when a successful read and write operation respectively occurs.

fifth bug :

at the case of both write and read occurs counter flag should behave differently.

Correct design :

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO_corrected(inter.DUT intt);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
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 intt.clk or negedge intt.rst_n) begin
    if (!intt.rst_n) begin
        wr_ptr <= 0;
        //
        intt.overflow <= 0;
        intt.wr_ack <= 0;
    end
    else if (intt.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= intt.data_in;
        intt.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        //
        intt.overflow <= 0;
    end
    else begin
        intt.wr_ack <= 0;
        if (intt.full && intt.wr_en) begin
            intt.overflow <= 1;
        end

        else begin
            intt.overflow <= 0;
        end

    end
end

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

        else begin
            intt.underflow <= 0;
        end

    end
end

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

assign intt.full = (count == FIFO_DEPTH)? 1 : 0;
assign intt.empty = (count == 0)? 1 : 0;
assign intt.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //
assign intt.almostempty = (count == 1)? 1 : 0;
///////////////////////////////////////////////////////
`ifdef assertion
always_comb begin : blockName
    if (!intt.rst_n) begin
        RESET:assert final(count==3'b0 && !intt.full && intt.empty && !intt.almostempty &&
!intt.almostfull && !rd_ptr && !wr_ptr && !intt.overflow && !intt.underflow );
    end
end
full : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) (count)==FIFO_DEPTH |->
intt.full==1'b1);
empty : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) (count)==3'b0 |->
intt.empty==1'b1);
almostfull : assert property (@(posedge intt.clk) disable iff(!intt.rst_n)
(count)==FIFO_DEPTH-1'b1 |-> intt.almostfull==1'b1);
almostempty : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) (count)==3'b1 |->
intt.almostempty==1);
COUNTER_UP : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.wr_en==1'b1
&& intt.rd_en==1'b0 && !intt.full |=> count==$past(count)+1'b1);
COUNTER_DOWN : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.wr_en==1'b0
&& intt.rd_en==1'b1 && !intt.empty |=> count==$past(count)-1'b1);
COUNTER_FULL : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.wr_en==1'b1
&& intt.rd_en==1'b1 && intt.full |=> count==$past(count)-1'b1);
COUNTER_EMPTY : assert property (@(posedge intt.clk) disable iff(!intt.rst_n)
intt.wr_en==1'b1 && intt.rd_en==1'b1 && intt.empty |=> count==$past(count)+1'b1);
OVERFLOW : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.wr_en==1'b1 &&
intt.full |=> intt.overflow==1);
UNDERFLOW : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.rd_en==1'b1 &&
intt.empty |=> intt.underflow==1);
RD_POINTER : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.rd_en==1'b1
&& count != 1'b0 |=> rd_ptr==$past(rd_ptr)+1'b1);
WR_POINTER : assert property (@(posedge intt.clk) disable iff(!intt.rst_n) intt.wr_en==1'b1
&& count < FIFO_DEPTH |=> wr_ptr==$past(wr_ptr)+1'b1);
`endif
endmodule
```

do file :

```
vlib work
vlog +define+assertion -f src_files.list +cover -covercells
vsim -voptargs=+acc work.fifo_top -cover
add wave -position insertpoint  \
sim:/fifo_top/inter/almostempty \
sim:/fifo_top/inter/almostfull \
sim:/fifo_top/inter/clk \
sim:/fifo_top/inter/data_in \
sim:/fifo_top/inter/data_out \
sim:/fifo_top/inter/empty \
sim:/fifo_top/inter/full \
sim:/fifo_top/inter/overflow \
sim:/fifo_top/inter/rd_en \
sim:/fifo_top/inter/rst_n \
sim:/fifo_top/inter/underflow \
sim:/fifo_top/inter/wr_ack \
sim:/fifo_top/inter/wr_en
add wave -position insertpoint  \
sim:/fifo_top/DUT/count
run -all
coverage save fifo.ucdb -onexit
#vcover report fifo.ucdb -details -all -output coverage_rpt_fifo.txt
```

Src_files.list :

fifo_interface.sv

FIFO_corrected.sv

trans_pck.sv

score_pck.sv

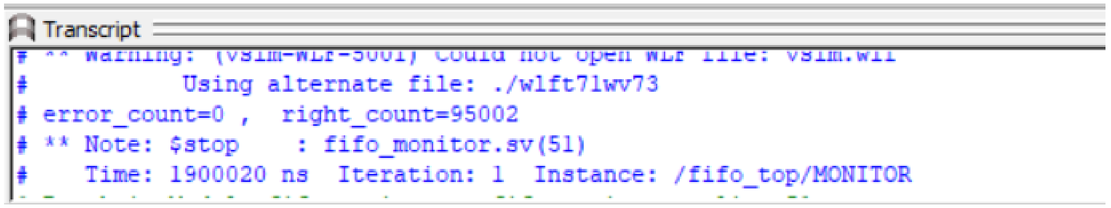
cover_pck.sv

fifo_monitor.sv

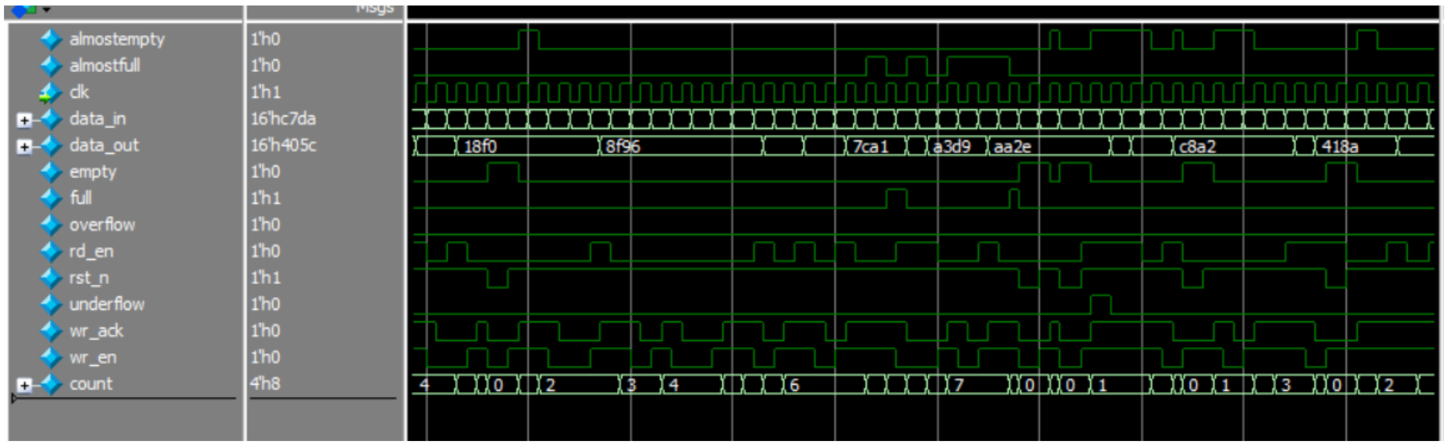
fifo_tb.sv

fifo_top.sv

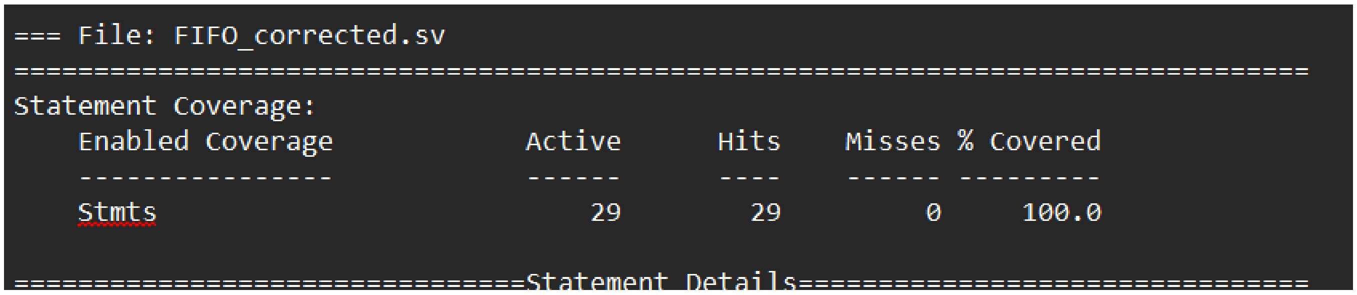
Waveform and transcript :



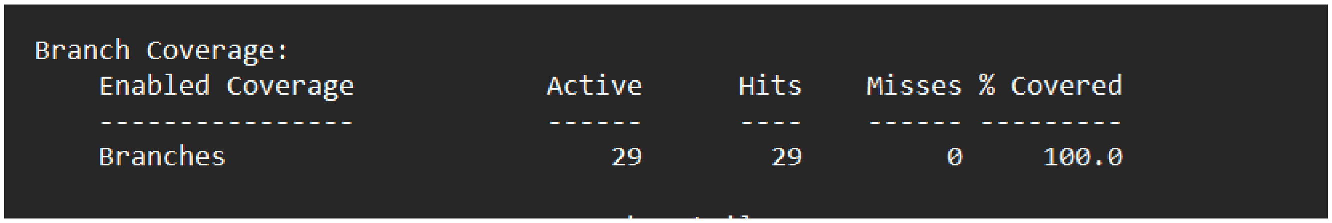
```
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#           Using alternate file: ./wlft7lwv73
# error_count=0 ,  right_count=95002
# ** Note: $stop    : fifo_monitor.sv(51)
#    Time: 1900020 ns  Iteration: 1  Instance: /fifo_top/MONITOR
```



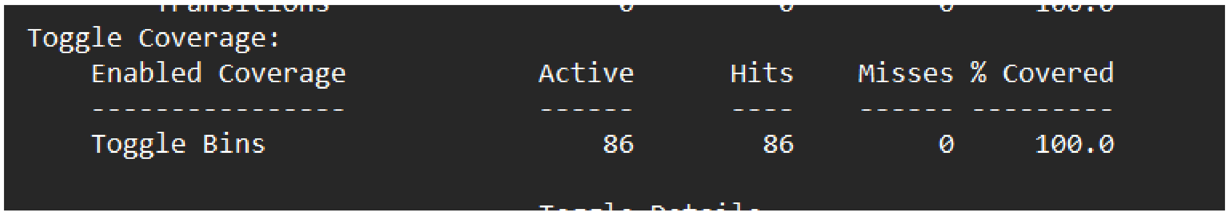
Coverage :



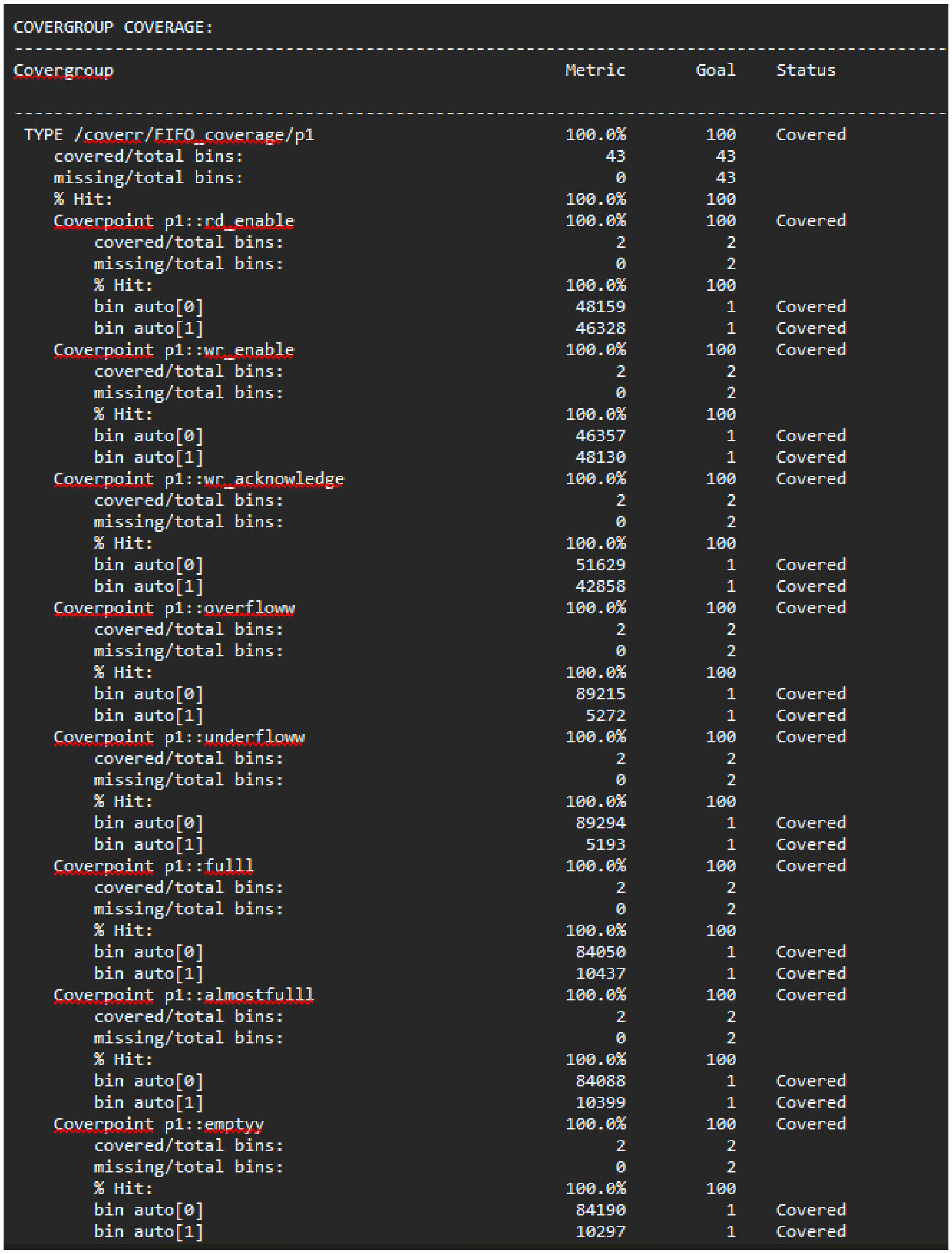
```
=== File: FIFO_corrected.sv
====================================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                             29        29         0     100.0

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



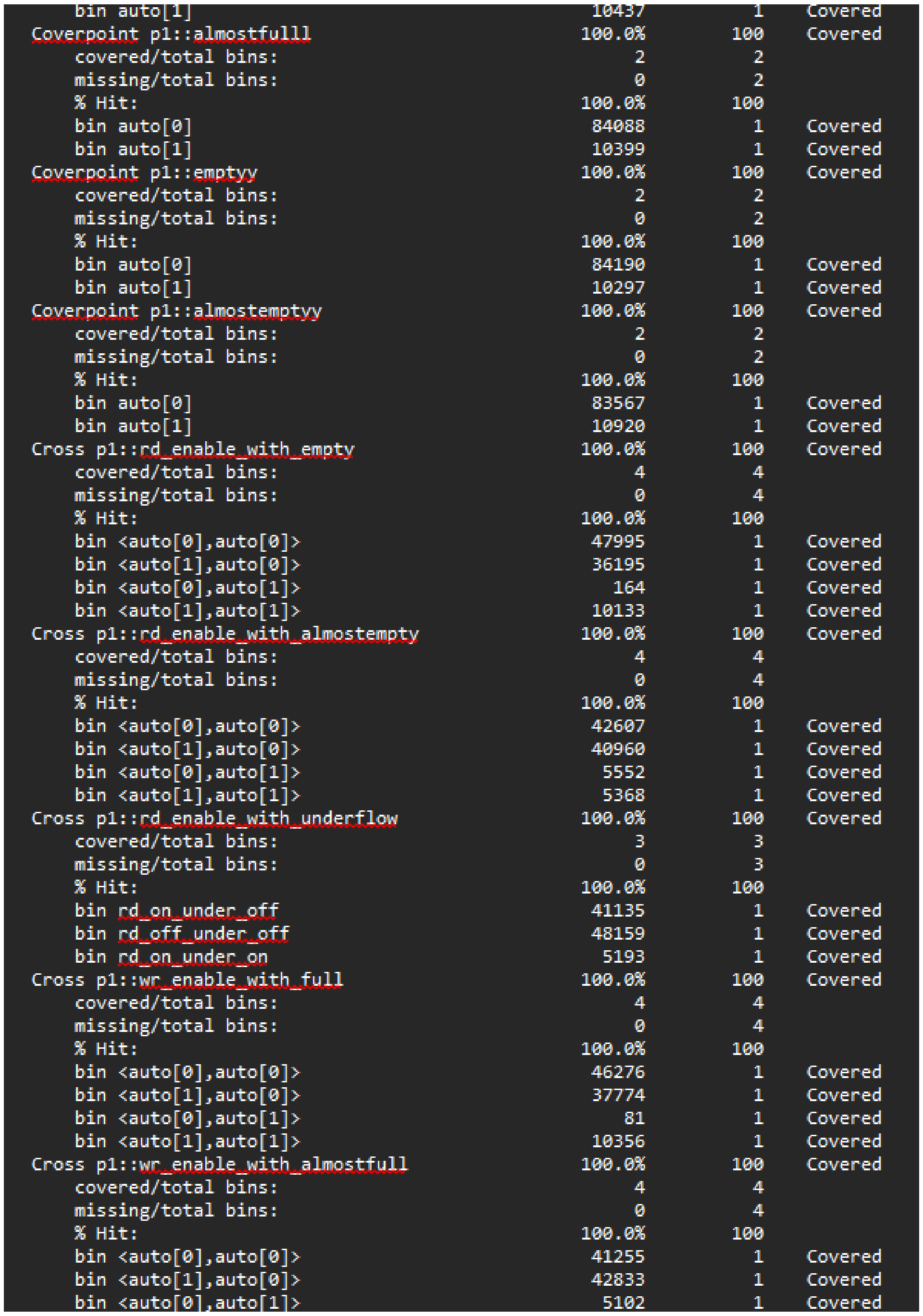
```
Branch Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Branches                          29        29         0     100.0
```



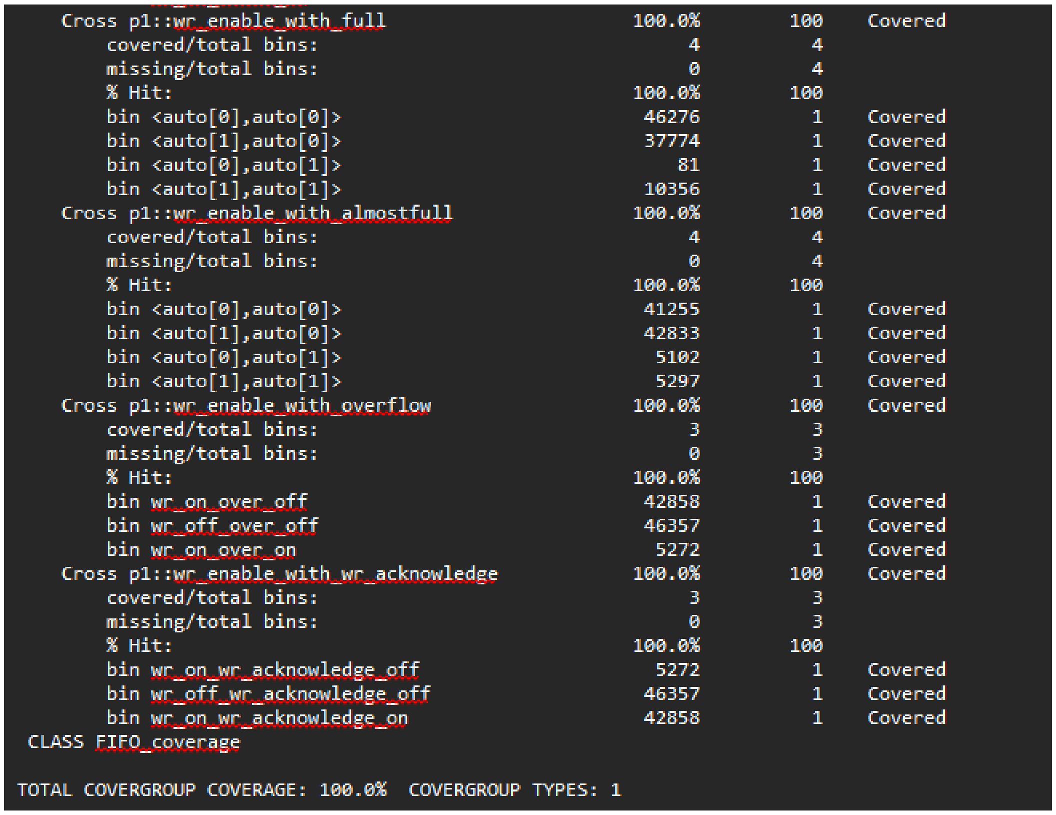
```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       86        86         0     100.0
```



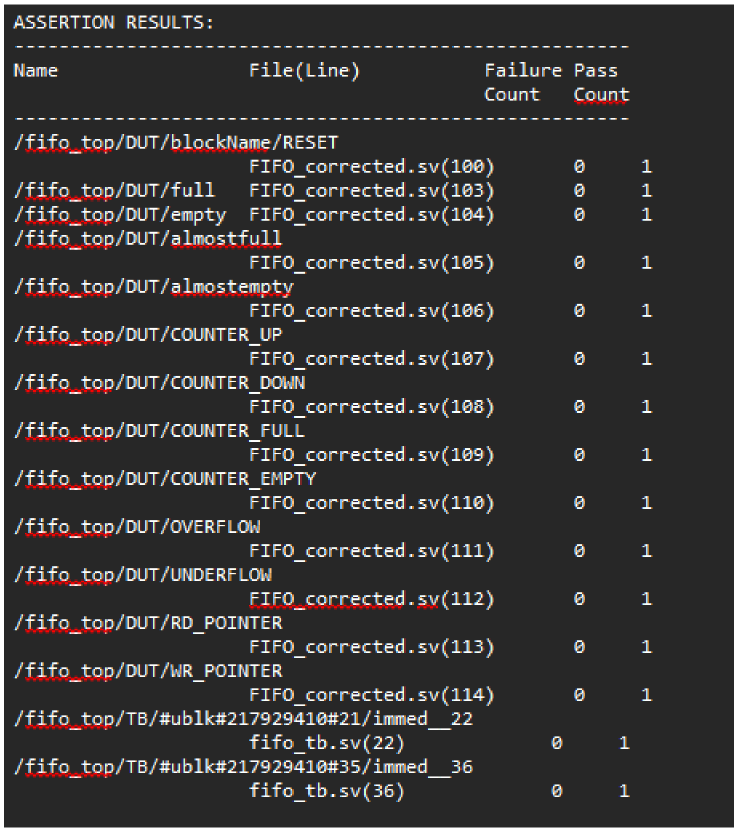
```
COVERGROUP COVERAGE:
-----------------------------------------------------------------------------------------------
Covergroup                                             Metric       Goal    Status

-----------------------------------------------------------------------------------------------
 TYPE /coverr/FIFO_coverage/p1                         100.0%        100    Covered
    covered/total bins:                                    43         43
    missing/total bins:                                     0         43
    % Hit:                                             100.0%        100
    Coverpoint p1::rd_enable                           100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     48159          1    Covered
        bin auto[1]                                     46328          1    Covered
    Coverpoint p1::wr_enable                           100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     46357          1    Covered
        bin auto[1]                                     48130          1    Covered
    Coverpoint p1::wr_acknowledge                      100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     51629          1    Covered
        bin auto[1]                                     42858          1    Covered
    Coverpoint p1::overfloww                           100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     89215          1    Covered
        bin auto[1]                                      5272          1    Covered
    Coverpoint p1::underfloww                          100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     89294          1    Covered
        bin auto[1]                                      5193          1    Covered
    Coverpoint p1::fulll                               100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     84050          1    Covered
        bin auto[1]                                     10437          1    Covered
    Coverpoint p1::almostfulll                         100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     84088          1    Covered
        bin auto[1]                                     10399          1    Covered
    Coverpoint p1::emptyy                              100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin auto[0]                                     84190          1    Covered
        bin auto[1]                                     10297          1    Covered
```



```
        bin auto[1]                                  10437           1     Covered
    Coverpoint p1::almostfulll                      100.0%         100     Covered
        covered/total bins:                              2           2
        missing/total bins:                              0           2
        % Hit:                                      100.0%         100
        bin auto[0]                                  84088           1     Covered
        bin auto[1]                                  10399           1     Covered
    Coverpoint p1::emptyy                           100.0%         100     Covered
        covered/total bins:                              2           2
        missing/total bins:                              0           2
        % Hit:                                      100.0%         100
        bin auto[0]                                  84190           1     Covered
        bin auto[1]                                  10297           1     Covered
    Coverpoint p1::almostemptyy                     100.0%         100     Covered
        covered/total bins:                              2           2
        missing/total bins:                              0           2
        % Hit:                                      100.0%         100
        bin auto[0]                                  83567           1     Covered
        bin auto[1]                                  10920           1     Covered
    Cross p1::rd_enable_with_empty                  100.0%         100     Covered
        covered/total bins:                              4           4
        missing/total bins:                              0           4
        % Hit:                                      100.0%         100
        bin <auto[0],auto[0]>                        47995           1     Covered
        bin <auto[1],auto[0]>                        36195           1     Covered
        bin <auto[0],auto[1]>                          164           1     Covered
        bin <auto[1],auto[1]>                        10133           1     Covered
    Cross p1::rd_enable_with_almostempty            100.0%         100     Covered
        covered/total bins:                              4           4
        missing/total bins:                              0           4
        % Hit:                                      100.0%         100
        bin <auto[0],auto[0]>                        42607           1     Covered
        bin <auto[1],auto[0]>                        40960           1     Covered
        bin <auto[0],auto[1]>                         5552           1     Covered
        bin <auto[1],auto[1]>                         5368           1     Covered
    Cross p1::rd_enable_with_underflow              100.0%         100     Covered
        covered/total bins:                              3           3
        missing/total bins:                              0           3
        % Hit:                                      100.0%         100
        bin rd_on_under_off                          41135           1     Covered
        bin rd_off_under_off                         48159           1     Covered
        bin rd_on_under_on                            5193           1     Covered
    Cross p1::wr_enable_with_full                   100.0%         100     Covered
        covered/total bins:                              4           4
        missing/total bins:                              0           4
        % Hit:                                      100.0%         100
        bin <auto[0],auto[0]>                        46276           1     Covered
        bin <auto[1],auto[0]>                        37774           1     Covered
        bin <auto[0],auto[1]>                           81           1     Covered
        bin <auto[1],auto[1]>                        10356           1     Covered
    Cross p1::wr_enable_with_almostfull             100.0%         100     Covered
        covered/total bins:                              4           4
        missing/total bins:                              0           4
        % Hit:                                      100.0%         100
        bin <auto[0],auto[0]>                        41255           1     Covered
        bin <auto[1],auto[0]>                        42833           1     Covered
        bin <auto[0],auto[1]>                         5102           1     Covered
```



```
    Cross p1::wr_enable_with_full                      100.0%        100    Covered
        covered/total bins:                                 4          4
        missing/total bins:                                 0          4
        % Hit:                                         100.0%        100
        bin <auto[0],auto[0]>                           46276          1    Covered
        bin <auto[1],auto[0]>                           37774          1    Covered
        bin <auto[0],auto[1]>                              81          1    Covered
        bin <auto[1],auto[1]>                           10356          1    Covered
    Cross p1::wr_enable_with_almostfull                100.0%        100    Covered
        covered/total bins:                                 4          4
        missing/total bins:                                 0          4
        % Hit:                                         100.0%        100
        bin <auto[0],auto[0]>                           41255          1    Covered
        bin <auto[1],auto[0]>                           42833          1    Covered
        bin <auto[0],auto[1]>                            5102          1    Covered
        bin <auto[1],auto[1]>                            5297          1    Covered
    Cross p1::wr_enable_with_overflow                  100.0%        100    Covered
        covered/total bins:                                 3          3
        missing/total bins:                                 0          3
        % Hit:                                         100.0%        100
        bin wr_on_over_off                              42858          1    Covered
        bin wr_off_over_off                             46357          1    Covered
        bin wr_on_over_on                                5272          1    Covered
    Cross p1::wr_enable_with_wr_acknowledge            100.0%        100    Covered
        covered/total bins:                                 3          3
        missing/total bins:                                 0          3
        % Hit:                                         100.0%        100
        bin wr_on_wr_acknowledge_off                     5272          1    Covered
        bin wr_off_wr_acknowledge_off                   46357          1    Covered
        bin wr_on_wr_acknowledge_on                     42858          1    Covered
 CLASS FIFO_coverage

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```



```
ASSERTION RESULTS:
-------------------------------------------------------
Name                 File(Line)           Failure Pass
                                          Count   Count
-------------------------------------------------------
/fifo_top/DUT/blockName/RESET
                     FIFO_corrected.sv(100)       0     1
/fifo_top/DUT/full   FIFO_corrected.sv(103)       0     1
/fifo_top/DUT/empty  FIFO_corrected.sv(104)       0     1
/fifo_top/DUT/almostfull
                     FIFO_corrected.sv(105)       0     1
/fifo_top/DUT/almostempty
                     FIFO_corrected.sv(106)       0     1
/fifo_top/DUT/COUNTER_UP
                     FIFO_corrected.sv(107)       0     1
/fifo_top/DUT/COUNTER_DOWN
                     FIFO_corrected.sv(108)       0     1
/fifo_top/DUT/COUNTER_FULL
                     FIFO_corrected.sv(109)       0     1
/fifo_top/DUT/COUNTER_EMPTY
                     FIFO_corrected.sv(110)       0     1
/fifo_top/DUT/OVERFLOW
                     FIFO_corrected.sv(111)       0     1
/fifo_top/DUT/UNDERFLOW
                     FIFO_corrected.sv(112)       0     1
/fifo_top/DUT/RD_POINTER
                     FIFO_corrected.sv(113)       0     1
/fifo_top/DUT/WR_POINTER
                     FIFO_corrected.sv(114)       0     1
/fifo_top/TB/#ublk#217929410#21/immed__22
                     fifo_tb.sv(22)             0     1
/fifo_top/TB/#ublk#217929410#35/immed__36
                     fifo_tb.sv(36)             0     1
```

THANKS