Project 1

Synchronous FIFO

Name: George Safwat Wasfy Farag

Digital Verification Using SystemVerilog and UVM

Table of Contents

Design without bugs with Assertions:

```
module FIFO(fifo_intf.DUT intf);
localparam max_fifo_addr = $clog2( intf.FIFO_DEPTH);
//ceiling log with base 2
//Depth mem word
reg [ intf.FIFO_WIDTH-1:0] mem [intf.FIFO_DEPTH-1:0];

reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
reg [max_fifo_addr:0] count;
//WRITE
always @(posedge intf.clk or negedge intf.rst_n) begin
    if (!intf.rst_n) begin
        wr_ptr <= 0;
        intf.overflow<=0;
        intf.wr_ack<=0;
    end
    else if(intf.wr_en && intf.rd_en && intf.empty
)begin
        mem[wr_ptr] <= intf.data_in;
        intf.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        intf.overflow<=0;

    end
    else if (intf.wr_en && count < intf.FIFO_DEPTH)
     begin
        mem[wr_ptr] <= intf.data_in;
        intf.wr_ack <= 1;
        wr_ptr <= (wr_ptr == intf.FIFO_DEPTH-1) ? 0 :
wr_ptr + 1;//adding checker to check the wraparound
        wr_ptr <= wr_ptr + 1;
```

```
        intf.overflow<=0;
    end
    else begin
        intf.wr_ack <= 0;
        if (intf.full && intf.wr_en)
            intf.overflow <= 1;
        else
            intf.overflow <= 0;
    end
end
//READ
always @(posedge intf.clk or negedge intf.rst_n) begin
    if (!intf.rst_n) begin
        rd_ptr <= 0;
        intf.underflow<=0;
        intf.data_out<=0;
    end
    else if (intf.wr_en && intf.rd_en && intf.full)
    begin
        intf.data_out <= mem[rd_ptr];
        rd_ptr <= (rd_ptr == intf.FIFO_DEPTH-1) ? 0 :
rd_ptr + 1;//adding checker to check the wraparound
        rd_ptr <= rd_ptr + 1;
        intf.underflow<=0;

    end
    else if (intf.rd_en && count != 0)
     begin
        intf.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        intf.underflow<=0;
    end
```

```
    else if (intf.empty && intf.rd_en) //bug==>adding
underflow here not below
            intf.underflow  <= 1;
        else
            intf.underflow  <= 0;
end

always @(posedge intf.clk or negedge intf.rst_n) begin
    if (!intf.rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({intf.wr_en, intf.rd_en} == 2'b10) &&
!intf.full)
            count <= count + 1;
        else if ( ({intf.wr_en, intf.rd_en} == 2'b01) &&
!intf.empty)
            count <= count - 1; //bug==>adding 2 cases
when write and read enable are asserted
        else if ( ({ intf.wr_en,  intf.rd_en} == 2'b11)
&& intf.full)
            count = count - 1;
                    else if ( ({
intf.wr_en,  intf.rd_en} == 2'b11) && intf.empty)
            count = count + 1;

    end
end


assign intf.full = (count ==  intf.FIFO_DEPTH)? 1 : 0;
assign intf.empty = (count == 0)? 1 : 0;
```

```
assign intf.almostfull = (count ==  intf.FIFO_DEPTH-1)?
1 : 0; //bug FIFO_DEPTH-2 should be -1
assign intf.almostempty = (count == 1)? 1 : 0;

//Add assertions only in simulation, not in synthesis.
`ifdef SIM
//ASSERTIONS
always_comb begin
if(!intf.rst_n) begin
reset: assert final(count == 0 && wr_ptr==0 && rd_ptr==0
);
cvr_reset:cover final(count == 0 && wr_ptr==0 &&
rd_ptr==0);
end
end
property p1;
@(posedge intf.clk) disable iff(!intf.rst_n) (intf.wr_en
&& !intf.full) |=>(intf.wr_ack)
endproperty
wr_ack_assert: assert property(p1);
wr_ack_cvr: cover property(p1);

property p2;
@(posedge intf.clk) disable iff(!intf.rst_n) (intf.wr_en
&& intf.full) |=>(intf.overflow);
endproperty
overflow_assert: assert property(p2);
overflow_cvr: cover property(p2);

property p3;
@(posedge intf.clk) disable iff(!intf.rst_n) (intf.rd_en
&& intf.empty) |=>(intf.underflow);
```

```
endproperty
underflow_assert: assert property(p3);
underflow_cvr: cover property(p3);

property p4;
@(posedge intf.clk) disable iff(!intf.rst_n) (!count) |-
>(intf.empty);
endproperty
empty_assert: assert property(p4);
empty_cvr: cover property(p4);

property p5;
@(posedge intf.clk) disable iff(!intf.rst_n)
(count===intf.FIFO_DEPTH) |->(intf.full);
endproperty
full_assert: assert property(p5);
full_cvr: cover property(p5);

property p6;
@(posedge intf.clk) disable iff(!intf.rst_n)
(count===(intf.FIFO_DEPTH-1)) |->(intf.almostfull);
endproperty
almostfull_assert: assert property(p6);
almostfull_cvr: cover property(p6);

property p7;
@(posedge intf.clk) disable iff(!intf.rst_n) (count===1)
|->(intf.almostempty);
endproperty
almostempty_assert: assert property(p7);
almostempty_cvr: cover property(p7);
```

```
// Write pointer wraparound
property p8;
  @(posedge intf.clk) disable iff(!intf.rst_n)
    (wr_ptr == intf.FIFO_DEPTH-1 && intf.wr_en &&
!intf.full) |=> (wr_ptr == 0);
endproperty
pointer_wraparound_assert_write: assert property(p8);
pointer_wraparound_cvr_write:   cover property(p8);

// Read pointer wraparound
property p9;
  @(posedge intf.clk) disable iff(!intf.rst_n)
    (rd_ptr == intf.FIFO_DEPTH-1 && intf.rd_en &&
!intf.empty) |=> (rd_ptr == 0);
endproperty
pointer_wraparound_assert_read: assert property(p9);
pointer_wraparound_cvr_read:   cover property(p9);


property p10;
@(posedge intf.clk) disable iff(!intf.rst_n)
(wr_ptr<intf.FIFO_DEPTH) && (rd_ptr<intf.FIFO_DEPTH) &&
(count<=intf.FIFO_DEPTH)
;
endproperty
threshold_assert: assert property(p10);
threshold_cvr: cover property(p10);
`endif

endmodule
```

Interface:

```
interface fifo_intf#(parameter FIFO_WIDTH = 16,
parameter FIFO_DEPTH = 8)(clk);
//INPUTS
input bit clk;
bit [FIFO_WIDTH-1:0] data_in;
bit wr_en,rst_n,rd_en;
//OUTPUTS
logic [FIFO_WIDTH-1:0] data_out;
logic
full,empty,almostfull,almostempty,overflow,underflow,wr_
ack;

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

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

modport mon (input
clk,data_in,rst_n,wr_en,rd_en,data_out,wr_ack,overflow,f
ull,
empty,almostfull,almostempty,underflow);

endinterface
```

Top:

```
module top();
bit clk;
initial begin
clk=0;
forever #1 clk=~clk;
end

fifo_intf intf(clk);
FIFO DUT(intf);
FIFO_tb TEST(intf);
monitor mon(intf);
endmodule
```

Shared Package:

```
package shared_pkg;
bit test_finished;  // signal refer to the end of the
testbench
int error_count,correct_count;
event trigger;
endpackage
```

Transaction:

```
package transaction_pkg;
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
class FIFO_transaction;
//INPUTS
rand bit clk;
rand bit [FIFO_WIDTH-1:0] data_in;
rand bit wr_en,rst_n,rd_en;
//OUTPUTS
logic [FIFO_WIDTH-1:0] data_out;
logic
full,empty,almostfull,almostempty,overflow,underflow,wr_
ack;
int RD_EN_ON_DIST , WR_EN_ON_DIST;

//CONSTRUCTOR
function new(int Read=30, int Write=70);
RD_EN_ON_DIST=Read;
WR_EN_ON_DIST=Write;
endfunction
//CONSTRAINTS
constraint g{
    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
```

Coverage Collector:

```
package func_pkg;
import transaction_pkg::*;
class FIFO_coverage ;

FIFO_transaction F_cvg_txn=new();

covergroup cvr_grp;

write_enable: coverpoint F_cvg_txn.wr_en {
    bins wr_enable={0,1};
}
read_enable: coverpoint F_cvg_txn.rd_en{
    bins rd_enable={0,1};
}
FULL: coverpoint F_cvg_txn.full{
    bins Full={0,1};
}
EMPTY: coverpoint F_cvg_txn.empty{
    bins Empty={0,1};
}
ALMOSTFULL: coverpoint F_cvg_txn.almostfull{
    bins Almostfull={0,1};
}
ALMOSTEMPTY: coverpoint F_cvg_txn.almostempty{
    bins Almostempty={0,1};
}
OVERFLOW: coverpoint F_cvg_txn.overflow{
    bins Overflow={0,1};
}
UNDERFLOW: coverpoint F_cvg_txn.underflow{
```

```
    bins Underflow={0,1};
}
Write_ack: coverpoint F_cvg_txn.wr_ack{
    bins write_acknowledge={0,1};
}

rd_en_with_wr_enable_full: cross read_enable,
write_enable, FULL {
   ignore_bins rd_en_with_active_full =
       binsof(read_enable) intersect {1} &&
       binsof(FULL) intersect {1};
}

rd_en_with_wr_enable_empty: cross
read_enable,write_enable,EMPTY;
rd_en_with_wr_enable_almostfull: cross read_enable,
write_enable,ALMOSTFULL;
rd_en_with_wr_enable_almostempty: cross
read_enable,write_enable,ALMOSTEMPTY;

rd_en_with_wr_enable_underflow: cross read_enable,
write_enable, UNDERFLOW {
   ignore_bins read_with_underflow =
       binsof(read_enable) intersect {0} &&
       binsof(UNDERFLOW) intersect {1};
}

rd_en_with_wr_enable_overflow: cross read_enable,
write_enable, OVERFLOW {
   ignore_bins write_with_overflow =
       binsof(write_enable) intersect {0} &&
       binsof(OVERFLOW) intersect {1};
```

```
}

rd_en_with_wr_enable_wr_ack: cross read_enable,
write_enable, Write_ack {
   ignore_bins write_with_wr_ack =
       binsof(write_enable) intersect {0} &&
       binsof(Write_ack) intersect {1};
}

endgroup
//CONSTRUCTOR
function new();
cvr_grp=new();
//F_cvg_txn = new();
endfunction

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

endclass
endpackage
```

Testbench:

```
import scoreboard_pkg::*;
import transaction_pkg::*;
import shared_pkg::*;
module FIFO_tb(fifo_intf.TEST intf);
FIFO_transaction fifo_tr;
integer i;

initial begin
    fifo_tr=new();
            intf.rst_n = 1;
            intf.rst_n = 0;
            -> trigger;
         @(negedge intf.clk);
            intf.rst_n = 1;
        //READ
        for (i=0;i<10000;i=i+1)begin
        assert(fifo_tr.randomize());
        intf.data_in=fifo_tr.data_in;
        intf.rst_n = fifo_tr.rst_n;
        intf.rd_en = fifo_tr.rd_en;
        intf.wr_en = fifo_tr.wr_en;
        -> trigger;
        $display("rd_en=%d,wr_en=%d",fifo_tr.rd_en,fifo_tr.wr_e
n);
        @(negedge intf.clk);
        end
    test_finished=1;
    -> trigger;
    end
endmodule
```

Scoreboard:

```
package scoreboard_pkg;
import transaction_pkg::*;
import shared_pkg::*;
import func_pkg::*;
class FIFO_scoreboard;
localparam max_fifo_addr = $clog2(FIFO_DEPTH);
logic [FIFO_WIDTH-1:0] data_out_ref;
logic
full_ref,empty_ref,almostfull_ref,almostempty_ref,overflow_ref,
underflow_ref,wr_ack_ref;
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;

function void comb_flags();
   full_ref       = (count == FIFO_DEPTH) ? 1 : 0;
   empty_ref      = (count == 0) ? 1 : 0;
   almostfull_ref = (count == FIFO_DEPTH-1) ? 1 : 0;
   almostempty_ref= (count == 1) ? 1 : 0;
endfunction



function void check_data(FIFO_transaction tr);
      // Run the reference model
      reference_model(tr);
      if((data_out_ref==tr.data_out)&&({full_ref,empty_ref,almo
stfull_ref,almostempty_ref,underflow_ref,overflow_ref}=={tr.ful
l,tr.empty,tr.almostfull,tr.almostempty,tr.underflow,tr.overflo
w}))
      begin
            correct_count++;
$display("rst_n=%0d data_in=%0d data_out=%0d data_out_ref=%0d
wr_ack=%0d wr_ack_ref=%0d overflow=%0d overflow_ref=%0d
```

```
full=%0d full_ref=%0d empty=%0d empty_ref=%0d almostfull=%0d
almostfull_ref=%0d almostempty=%0d almostempty_ref=%0d
underflow=%0d underflow_ref=%0d wr_en=%0d rd_en=%0d count=%0d",
         tr.rst_n, tr.data_in, tr.data_out, data_out_ref,
         tr.wr_ack, wr_ack_ref, tr.overflow, overflow_ref,
         tr.full, full_ref, tr.empty, empty_ref,
         tr.almostfull, almostfull_ref, tr.almostempty,
almostempty_ref,
         tr.underflow, underflow_ref, tr.wr_en, tr.rd_en,
count);


      end
      else begin
             error_count++;
            $display("Error ==> correct_count=%d,
error_count=%d, data_in=%d, data_out_ref=%d, tr.data_out=%d
",correct_count,error_count,tr.data_in,data_out_ref,tr.data_out
);
      end
endfunction

function void reference_model(FIFO_transaction ref_tr);


//REFRENCE FOR WRITE

if (!ref_tr.rst_n) begin
            wr_ptr=0;
            full_ref=0;
            wr_ack_ref=0;
            empty_ref=1;
            overflow_ref=0;

      end
      else if (ref_tr.wr_en && count < FIFO_DEPTH)
```

```
      begin
            mem[wr_ptr] = ref_tr.data_in;
            wr_ack_ref = 1;
            wr_ptr <= wr_ptr + 1;
            overflow_ref=0;
      end
      else if(ref_tr.wr_en && ref_tr.rd_en && empty_ref )begin
            mem[wr_ptr] = ref_tr.data_in;
            wr_ack_ref = 1;
            wr_ptr <= wr_ptr + 1;
            wr_ptr <= (wr_ptr == FIFO_DEPTH-1) ? 0 : wr_ptr +
1;
            overflow_ref=0;


      end
      else begin
            wr_ack_ref = 0;
            if (full_ref & ref_tr.wr_en)
                  overflow_ref = 1;
            else
                  overflow_ref = 0;
      end



//REFRENCE FOR READ
      if (!ref_tr.rst_n) begin
            rd_ptr = 0;
            data_out_ref = 0;
            empty_ref = 1;
            almostempty_ref = 0;
            underflow_ref = 0;
      end
      else if (ref_tr.rd_en && count != 0) begin
            data_out_ref = mem[rd_ptr];
```

```
            rd_ptr = rd_ptr + 1;
            underflow_ref=0;
      end
      else if(ref_tr.wr_en && ref_tr.rd_en  && full_ref )begin
            data_out_ref = mem[rd_ptr];
            rd_ptr = rd_ptr + 1;
            rd_ptr <= (rd_ptr == FIFO_DEPTH-1) ? 0 : rd_ptr +
1;
            underflow_ref=0;

      end
      else  begin
            if(empty_ref && ref_tr.rd_en)
                  underflow_ref  = 1;
        else
                   underflow_ref  = 0;
      end
//COUNTER
if (!ref_tr.rst_n) begin
            count = 0;
      end
      else begin
            if    ( ({ref_tr.wr_en, ref_tr.rd_en} == 2'b10) &&
!full_ref)
                  count = count + 1;
            else if ( ({ref_tr.wr_en, ref_tr.rd_en} == 2'b01)
&& !empty_ref)
                  count = count - 1;
            else if     ( ({ref_tr.wr_en, ref_tr.rd_en} ==
2'b11) && empty_ref)
                  count = count + 1;
            else if     ( ({ref_tr.wr_en, ref_tr.rd_en} ==
2'b11) && full_ref)
                  count = count - 1;

      end
```

```
      //calling flags
            comb_flags();
endfunction
endclass
endpackage
```

Monitor:

```
import shared_pkg::*;
import transaction_pkg::*;
import scoreboard_pkg::*;
import func_pkg::*;
module monitor(fifo_intf.mon intf);

FIFO_transaction tr= new();
FIFO_scoreboard sb= new();
FIFO_coverage cov= new();

initial begin
    forever begin

        wait (trigger.triggered);
            @(negedge intf.clk);
            //input
            tr.data_in=intf.data_in;
            tr.wr_en=intf.wr_en;
            tr.rst_n=intf.rst_n;
            tr.rd_en=intf.rd_en;
            //output
            tr.data_out=intf.data_out;
            tr.full=intf.full;
            tr.empty=intf.empty;
            tr.almostfull=intf.almostfull;
            tr.almostempty=intf.almostempty;
            tr.overflow=intf.overflow;
```

```
            tr.underflow=intf.underflow;
            tr.wr_ack=intf.wr_ack;
fork

        //PROCESS_1
        begin
        cov.sample_data(tr);
        end
        //PROCESS_2
        begin
        sb.check_data(tr);
        end

join
    if (test_finished==1)begin
    //$display("tr.data_in=%d,intf.data_in=%d,tr.data_out=%d,in
tf.data_out=%d",tr.data_in,intf.data_in,tr.data_out,intf.data_o
ut);
     $display("error_count=%d,
correct_count=%d",error_count,correct_count);
     $stop;
                end
        end
    end
endmodule
```

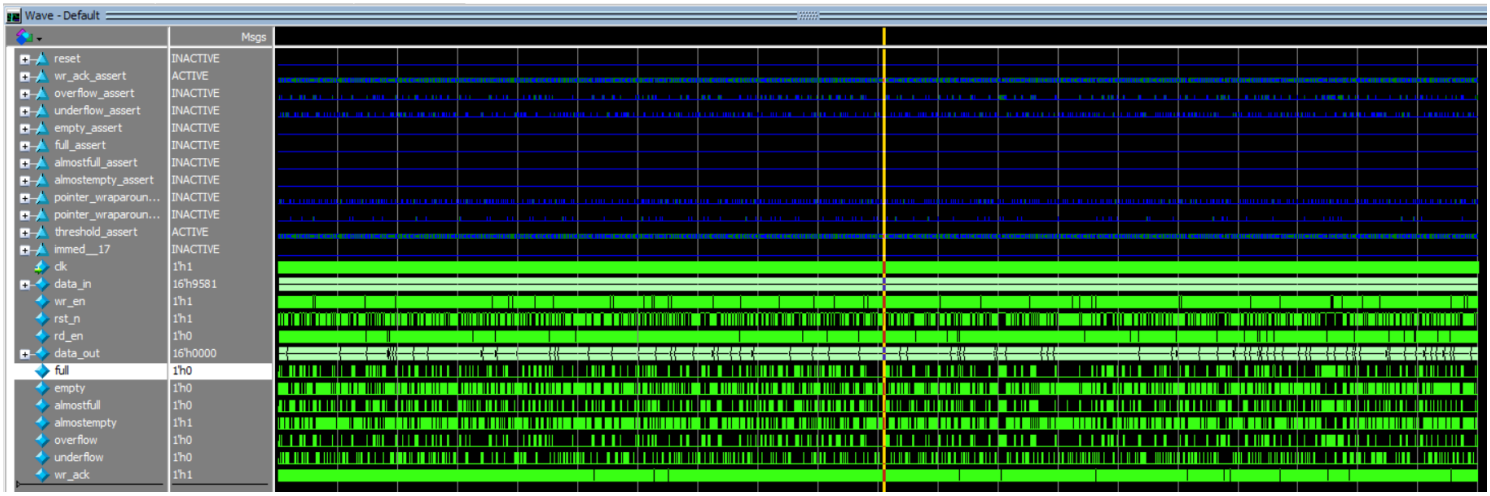
Transcript:

```
# rst_n=0 data_in=20724 data_out=0 data_out_ref=0 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=1 empty_ref=1 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0 under
flow_ref=0 wr_en=0 rd_en=0 count=0
# rd_en=1,wr_en=0
# rst_n=1 data_in=19169 data_out=0 data_out_ref=0 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=1 empty_ref=1 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=1 under
flow_ref=1 wr_en=0 rd_en=1 count=0
# rd_en=1,wr_en=1
# rst_n=1 data_in=36425 data_out=0 data_out_ref=0 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=1 almostempty_ref=1 underflow=1 under
flow_ref=1 wr_en=1 rd_en=1 count=1
# rd_en=0,wr_en=1
# rst_n=1 data_in=41006 data_out=0 data_out_ref=0 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0 under
flow_ref=0 wr_en=1 rd_en=0 count=2
# rd_en=0,wr_en=1
# rst_n=1 data_in=40898 data_out=0 data_out_ref=0 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0 under
flow_ref=0 wr_en=1 rd_en=0 count=3
# rd_en=0,wr_en=1
# rst_n=1 data_in=29577 data_out=0 data_out_ref=0 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0 under
flow_ref=0 wr_en=1 rd_en=0 count=4
# rd_en=0,wr_en=0
# rst_n=1 data_in=27097 data_out=0 data_out_ref=0 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0 under
flow_ref=0 wr_en=0 rd_en=0 count=4
# rd_en=1,wr_en=1
# rst_n=1 data_in=1496 data_out=36425 data_out_ref=36425 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=
0 underflow_ref=0 wr_en=1 rd_en=1 count=4
# rd_en=0,wr_en=1
# rst_n=1 data_in=28280 data_out=36425 data_out_ref=36425 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow
=0 underflow_ref=0 wr_en=1 rd_en=0 count=5
# rd_en=0,wr_en=0
# rst_n=1 data_in=30072 data_out=36425 data_out_ref=36425 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow
=0 underflow_ref=0 wr_en=0 rd_en=0 count=5
# rd_en=0,wr_en=0
# rst_n=1 data_in=337 data_out=36425 data_out_ref=36425 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow=0
 underflow_ref=0 wr_en=0 rd_en=0 count=5
# rd_en=0,wr_en=1
# rst_n=1 data_in=46918 data_out=36425 data_out_ref=36425 wr_ack=1 wr_ack_ref=1 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow
=0 underflow_ref=0 wr_en=1 rd_en=0 count=6
# rd_en=1,wr_en=0
# rst_n=1 data_in=46649 data_out=41006 data_out_ref=41006 wr_ack=0 wr_ack_ref=0 overflow=0 overflow_ref=0 full=0 full_ref=0 empty=0 empty_ref=0 almostfull=0 almostfull_ref=0 almostempty=0 almostempty_ref=0 underflow
=0 underflow_ref=0 wr_en=0 rd_en=1 count=5
```

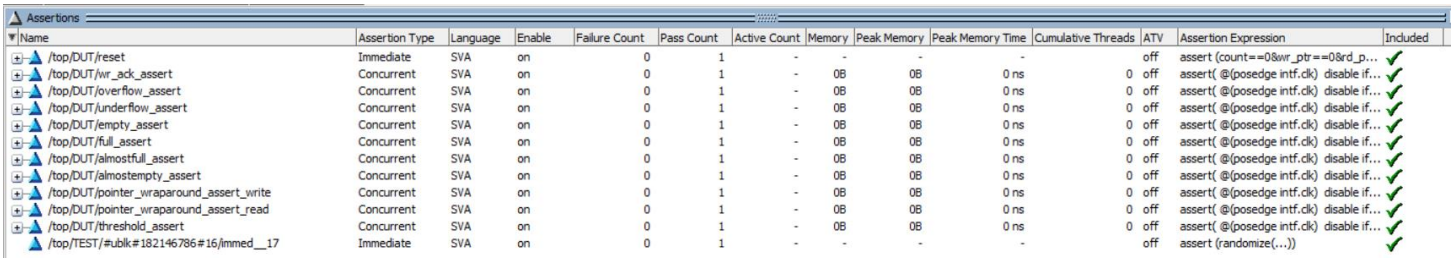
Do file:

```
vlib work
vlog +define+SIM shared_pkg.sv top.sv interface.sv FIFO.sv  FIFO_TRANSACTIONS.sv func_cov_collection.sv  Monitor.sv  scoreboard.sv testbench_fifo.sv +cover -covercells
vsim -voptargs=+acc work.top -cover
coverage save FIFO.ucdb -onexit -du work.FIFO
add wave /top/DUT/reset /top/DUT/wr_ack_assert /top/DUT/overflow_assert /top/DUT/underflow_assert /top/DUT/empty_assert /top/DUT/full_assert /top/DUT/almostfull_assert
/top/DUT/almostempty_assert /top/DUT/pointer_wraparound_assert_write /top/DUT/pointer_wraparound_assert_read /top/DUT/threshold_assert /top/TEST/#ublk#182146786#16/immed__17
add wave -position insertpoint sim:/top/intf/*
run -all
coverage report -detail -cvg -directive -comments -file F_cover_fifo.txt -noa -all
quit -sim
vcover report FIFO.ucdb -details -all -output coverage_rpt_FIFO.txt
```

Simulation:

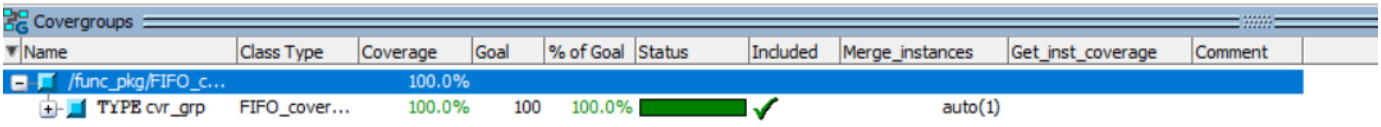


Assertions:



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/reset	Immediate	SVA	on	0	1	-	-	-	-		off	assert (count==0&wr_ptr==0&rd_p...	✓
/top/DUT/wr_ack_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/overflow_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/underflow_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/empty_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/full_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/almostfull_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/almostempty_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/pointer_wraparound_assert_write	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/pointer_wraparound_assert_read	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/DUT/threshold_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge intf.clk) disable if...	✓
/top/TEST/#ublk#182146786#16/immed__17	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓

Functional coverage:



Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/func_pkg/FIFO_c...		100.0%							
TYPE cvr_grp	FIFO_cover...	100.0%	100	100.0%	✓		auto(1)		

Reports:

Code coverage & Assertions:

Statement details:

```
Coverage Report by file with details

================================================================================
=== File: FIFO.sv
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           40        40         0     100.0

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

Statement Coverage for file FIFO.sv --

    1                                           module FIFO(fifo_intf.DUT intf);
    2                                           localparam max_fifo_addr = $clog2( intf.FIFO_DEPTH); //ceiling log with base 2
    3                                           //Depth mem word
    4                                           reg [ intf.FIFO_WIDTH-1:0] mem [intf.FIFO_DEPTH-1:0];
    5
    6                                           reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    7                                           reg [max_fifo_addr:0] count;
    8                                           //WRITE
    9               1                   10872   always @(posedge intf.clk or negedge intf.rst_n) begin
    10                                             if (!intf.rst_n) begin
    11              1                    1860                wr_ptr <= 0;
    12              1                    1860                intf.overflow<=0;
    13              1                    1860                intf.wr_ack<=0;
    14                                             end
    15                                             else if(intf.wr_en && intf.rd_en && intf.empty )begin
    16              1                     308                mem[wr_ptr] <= intf.data_in;
    17              1                     308                intf.wr_ack <= 1;
    18              1                     308                wr_ptr <= wr_ptr + 1;
    19              1                     308                intf.overflow<=0;
    20
    21                                             end
    22                                             else if (intf.wr_en && count < intf.FIFO_DEPTH)
    23                                              begin
    24              1                    5278                mem[wr_ptr] <= intf.data_in;
```

```
25          1                5278                intf.wr_ack <= 1;
26          1                5278                wr_ptr <= (wr_ptr == intf.FIFO_DEPTH-1) ? 0 : wr_ptr + 1;//adding checker to check the wraparound
27          1                5278                wr_ptr <= wr_ptr + 1;
28          1                5278                intf.overflow<=0;
29                                           end
30                                           else begin
31          1                3426                intf.wr_ack <= 0;
32                                               if (intf.full && intf.wr_en)
33          1                 753                       intf.overflow <= 1;
34                                               else
35          1                2673                       intf.overflow <= 0;
36                                           end
37                                       end
38                                       //READ
39          1               10872        always @(posedge intf.clk or negedge intf.rst_n) begin
40                                           if (!intf.rst_n) begin
41          1                1860                rd_ptr <= 0;
42          1                1860                intf.underflow<=0;
43          1                1860                intf.data_out<=0;
44                                           end
45                                           else if (intf.wr_en && intf.rd_en && intf.full)
46                                           begin
47          1                 237                intf.data_out <= mem[rd_ptr];
48          1                 237                rd_ptr <= (rd_ptr == intf.FIFO_DEPTH-1) ? 0 : rd_ptr + 1;//adding checker to check the wraparound
49          1                 237                rd_ptr <= rd_ptr + 1;
50          1                 237                intf.underflow<=0;
51
52                                           end
53                                           else if (intf.rd_en && count != 0)
54                                            begin
55          1                2012                intf.data_out <= mem[rd_ptr];
56          1                2012                rd_ptr <= rd_ptr + 1;
57          1                2012                intf.underflow<=0;
58                                           end
59                                           else if (intf.empty && intf.rd_en) //bug==>adding underflow here not below
60          1                 459                       intf.underflow  <= 1;
61                                                else
```

```
62          1                6304                    intf.underflow  <= 0;
63                                           end
64
65          1                9950        always @(posedge intf.clk or negedge intf.rst_n) begin
66                                           if (!intf.rst_n) begin
67          1                1824                    count <= 0;
68                                           end
69                                           else begin
70                                                   if      ( ({intf.wr_en, intf.rd_en} == 2'b10) && !intf.full)
71          1                3916                       count <= count + 1;
72                                                   else if ( ({intf.wr_en, intf.rd_en} == 2'b01) && !intf.empty)
73          1                 650                       count <= count - 1; //bug==>adding 2 cases when write and read enable are asserted
74                                                   else if ( ({ intf.wr_en,  intf.rd_en} == 2'b11) && intf.full)
75          1                 237                    count = count - 1;
76                                                    else if ( ({ intf.wr_en,  intf.rd_en} == 2'b11) && intf.empty)
77          1                 308                    count = count + 1;
78
79                                           end
80                                       end
81
82
83          1                5926        assign intf.full = (count ==  intf.FIFO_DEPTH)? 1 : 0;
84          1                5926        assign intf.empty = (count == 0)? 1 : 0;
85          1                5926        assign intf.almostfull = (count ==  intf.FIFO_DEPTH-1)? 1 : 0; //bug FIFO_DEPTH-2 should be -1
86          1                5926        assign intf.almostempty = (count == 1)? 1 : 0;
87
88                                       //Add assertions only in simulation, not in synthesis.
89                                       `ifdef SIM
90                                       //ASSERTIONS
91          1                9587        always_comb begin
92                                       if(!intf.rst_n) begin
93                                       reset: assert final(count == 0 && wr_ptr==0 && rd_ptr==0 );
94                                       cvr_reset:cover final(count == 0 && wr_ptr==0 && rd_ptr==0);
95                                       end
96                                       end
97                                       property p1;
98                                       @(posedge intf.clk) disable iff(!intf.rst_n) (intf.wr_en && !intf.full) |=>(intf.wr_ack)
```

```
99                                          endproperty
100                                         wr_ack_assert: assert property(p1);
101                                         wr_ack_cvr: cover property(p1);
102
103                                         property p2;
104                                         @(posedge intf.clk) disable iff(!intf.rst_n) (intf.wr_en && intf.full) |=>(intf.overflow);
105                                         endproperty
106                                         overflow_assert: assert property(p2);
107                                         overflow_cvr: cover property(p2);
108
109                                         property p3;
110                                         @(posedge intf.clk) disable iff(!intf.rst_n) (intf.rd_en && intf.empty) |=>(intf.underflow);
111                                         endproperty
112                                         underflow_assert: assert property(p3);
113                                         underflow_cvr: cover property(p3);
114
115                                         property p4;
116                                         @(posedge intf.clk) disable iff(!intf.rst_n) (!count) |->(intf.empty);
117                                         endproperty
118                                         empty_assert: assert property(p4);
119                                         empty_cvr: cover property(p4);
120
121                                         property p5;
122                                         @(posedge intf.clk) disable iff(!intf.rst_n) (count===intf.FIFO_DEPTH) |->(intf.full);
123                                         endproperty
124                                         full_assert: assert property(p5);
125                                         full_cvr: cover property(p5);
126
127                                         property p6;
128                                         @(posedge intf.clk) disable iff(!intf.rst_n) (count===(intf.FIFO_DEPTH-1)) |->(intf.almostfull);
129                                         endproperty
130                                         almostfull_assert: assert property(p6);
131                                         almostfull_cvr: cover property(p6);
132
133                                         property p7;
134                                         @(posedge intf.clk) disable iff(!intf.rst_n) (count===1) |->(intf.almostempty);
135                                         endproperty
136                                         almostempty_assert: assert property(p7);
137                                         almostempty_cvr: cover property(p7);
```

```
    138
    139                                         // Write pointer wraparound
    140                                         property p8;
    141                                           @(posedge intf.clk) disable iff(!intf.rst_n)
    142                                             (wr_ptr == intf.FIFO_DEPTH-1 && intf.wr_en && !intf.full) |=> (wr_ptr == 0);
    143                                         endproperty
    144                                         pointer_wraparound_assert_write: assert property(p8);
    145                                         pointer_wraparound_cvr_write:   cover property(p8);
    146
    147                                         // Read pointer wraparound
    148                                         property p9;
    149                                           @(posedge intf.clk) disable iff(!intf.rst_n)
    150                                             (rd_ptr == intf.FIFO_DEPTH-1 && intf.rd_en && !intf.empty) |=> (rd_ptr == 0);
    151                                         endproperty
    152                                         pointer_wraparound_assert_read: assert property(p9);
    153                                         pointer_wraparound_cvr_read:   cover property(p9);
    154
    155
    156                                         property p10;
    157                                         @(posedge intf.clk) disable iff(!intf.rst_n) (intf.wr_en && !intf.full) |=>(intf.wr_ack);
    158                                         endproperty
    159                                         threshold_assert: assert property(p10);
    160                                         threshold_cvr: cover property(p10);
    161                                         `endif
    162
    163                                         endmodule

Branch Coverage:
    Enabled Coverage           Active      Hits    Misses % Covered
    ----------------           ------      ----    ------ ---------
    Branches                       32        32         0     100.0
```

Branch details:

```
================================Branch Details================================

Branch Coverage for file FIFO.sv --

------------------------------------IF Branch------------------------------------
    10                                  10872     Count coming in to IF
    10              1                    1860        if (!intf.rst_n) begin
    15              1                     308        else if(intf.wr_en && intf.rd_en && intf.empty )begin
    22              1                    5278        else if (intf.wr_en && count < intf.FIFO_DEPTH)
    30              1                    3426        else begin
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    26                                   5278     Count coming in to IF
    26              1                     375                wr_ptr <= (wr_ptr == intf.FIFO_DEPTH-1) ? 0 : wr_ptr + 1;//adding checker to check the wraparound
    26              2                    4903                wr_ptr <= (wr_ptr == intf.FIFO_DEPTH-1) ? 0 : wr_ptr + 1;//adding checker to check the wraparound
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    32                                   3426     Count coming in to IF
    32              1                     753                if (intf.full && intf.wr_en)
    34              1                    2673                else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    40                                  10872     Count coming in to IF
    40              1                    1860        if (!intf.rst_n) begin
    45              1                     237        else if (intf.wr_en && intf.rd_en && intf.full)
    53              1                    2012        else if (intf.rd_en && count != 0)
    59              1                     459        else if (intf.empty && intf.rd_en) //bug==>adding underflow here not below
    61              1                    6304             else
Branch totals: 5 hits of 5 branches = 100.0%
```

```
------------------------------------IF Branch------------------------------------
    48                                    237     Count coming in to IF
    48              1                      25                rd_ptr <= (rd_ptr == intf.FIFO_DEPTH-1) ? 0 : rd_ptr + 1;//adding checker to check the wraparound
    48              2                     212                rd_ptr <= (rd_ptr == intf.FIFO_DEPTH-1) ? 0 : rd_ptr + 1;//adding checker to check the wraparound
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    66                                   9950     Count coming in to IF
    66              1                    1824        if (!intf.rst_n) begin
    69              1                    8126        else begin
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    70                                   8126     Count coming in to IF
    70              1                    3916                if      ( ({intf.wr_en, intf.rd_en} == 2'b10) && !intf.full)
    72              1                     650                else if ( ({intf.wr_en, intf.rd_en} == 2'b01) && !intf.empty)
    74              1                     237                else if ( ({ intf.wr_en,  intf.rd_en} == 2'b11) && intf.full)
    76              1                     308                                else if ( ({ intf.wr_en,  intf.rd_en} == 2'b11) && intf.empty)
                                          3015     All False Count
Branch totals: 5 hits of 5 branches = 100.0%

------------------------------------IF Branch------------------------------------
    83                                   5925     Count coming in to IF
    83              1                     433     assign intf.full = (count ==  intf.FIFO_DEPTH)? 1 : 0;
    83              2                    5492     assign intf.full = (count ==  intf.FIFO_DEPTH)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    84                                   5925     Count coming in to IF
    84              1                     954     assign intf.empty = (count == 0)? 1 : 0;
    84              2                    4971     assign intf.empty = (count == 0)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.0%
```

```
------------------------------------IF Branch------------------------------------
    85                                   5925     Count coming in to IF
    85              1                     605     assign intf.almostfull = (count ==  intf.FIFO_DEPTH-1)? 1 : 0; //bug FIFO_DEPTH-2 should be -1
    85              2                    5320     assign intf.almostfull = (count ==  intf.FIFO_DEPTH-1)? 1 : 0; //bug FIFO_DEPTH-2 should be -1
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    86                                   5925     Count coming in to IF
    86              1                    1065     assign intf.almostempty = (count == 1)? 1 : 0;
    86              2                    4860     assign intf.almostempty = (count == 1)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    92                                   9587     Count coming in to IF
    92              1                    1699     if(!intf.rst_n) begin
                                         7888     All False Count
Branch totals: 2 hits of 2 branches = 100.0%


Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------   -------    ------ ---------
    FEC Condition Terms             26        26         0     100.0
```

Condition details:

```
================================Condition Details================================

Condition Coverage for file FIFO.sv --

----------------Focused Condition View-------------------
Line       15 Item    1  ((intf.wr_en && intf.rd_en) && intf.empty)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
  intf.wr_en         Y
  intf.rd_en         Y
  intf.empty         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.wr_en_0          -
  Row   2:          1  intf.wr_en_1          (intf.empty && intf.rd_en)
  Row   3:          1  intf.rd_en_0          intf.wr_en
  Row   4:          1  intf.rd_en_1          (intf.empty && intf.wr_en)
  Row   5:          1  intf.empty_0          (intf.wr_en && intf.rd_en)
  Row   6:          1  intf.empty_1          (intf.wr_en && intf.rd_en)

----------------Focused Condition View-------------------
Line       22 Item    1  (intf.wr_en && (count < intf.FIFO_DEPTH))
Condition totals: 2 of 2 input terms covered = 100.0%

                Input Term   Covered  Reason for no coverage   Hint
               -----------  --------  ----------------------  --------------
                intf.wr_en         Y
  (count < intf.FIFO_DEPTH)        Y

     Rows:       Hits  FEC Target                  Non-masking condition(s)
 ---------  ---------  --------------------        -------------------------
  Row   1:          1  intf.wr_en_0                -
  Row   2:          1  intf.wr_en_1                (count < intf.FIFO_DEPTH)
  Row   3:          1  (count < intf.FIFO_DEPTH)_0  intf.wr_en
  Row   4:          1  (count < intf.FIFO_DEPTH)_1  intf.wr_en
```

```
----------------Focused Condition View-------------------
Line       32 Item    1  (intf.full && intf.wr_en)
Condition totals: 2 of 2 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  -------------
   intf.full         Y
  intf.wr_en         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.full_0           -
  Row   2:          1  intf.full_1           intf.wr_en
  Row   3:          1  intf.wr_en_0          intf.full
  Row   4:          1  intf.wr_en_1          intf.full

----------------Focused Condition View-------------------
Line       45 Item    1  ((intf.wr_en && intf.rd_en) && intf.full)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  -------------
  intf.wr_en         Y
  intf.rd_en         Y
   intf.full         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.wr_en_0          -
  Row   2:          1  intf.wr_en_1          (intf.full && intf.rd_en)
  Row   3:          1  intf.rd_en_0          intf.wr_en
  Row   4:          1  intf.rd_en_1          (intf.full && intf.wr_en)
  Row   5:          1  intf.full_0           (intf.wr_en && intf.rd_en)
  Row   6:          1  intf.full_1           (intf.wr_en && intf.rd_en)
```

```
----------------Focused Condition View-------------------
Line       53 Item    1  (intf.rd_en && (count != 0))
Condition totals: 2 of 2 input terms covered = 100.0%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
    intf.rd_en         Y
  (count != 0)         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  intf.rd_en_0         -
  Row   2:          1  intf.rd_en_1         (count != 0)
  Row   3:          1  (count != 0)_0       intf.rd_en
  Row   4:          1  (count != 0)_1       intf.rd_en

----------------Focused Condition View-------------------
Line       59 Item    1  (intf.empty && intf.rd_en)
Condition totals: 2 of 2 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
  intf.empty         Y
  intf.rd_en         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  intf.empty_0         -
  Row   2:          1  intf.empty_1         intf.rd_en
  Row   3:          1  intf.rd_en_0         intf.empty
  Row   4:          1  intf.rd_en_1         intf.empty
```

```
----------------Focused Condition View-------------------
Line       70 Item    1  ((~intf.rd_en && intf.wr_en) && ~intf.full)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  -------------
  intf.rd_en         Y
  intf.wr_en         Y
   intf.full         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.rd_en_0          (~intf.full && intf.wr_en)
  Row   2:          1  intf.rd_en_1          -
  Row   3:          1  intf.wr_en_0          ~intf.rd_en
  Row   4:          1  intf.wr_en_1          (~intf.full && ~intf.rd_en)
  Row   5:          1  intf.full_0           (~intf.rd_en && intf.wr_en)
  Row   6:          1  intf.full_1           (~intf.rd_en && intf.wr_en)

----------------Focused Condition View-------------------
Line       72 Item    1  ((intf.rd_en && ~intf.wr_en) && ~intf.empty)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  -------------
  intf.rd_en         Y
  intf.wr_en         Y
  intf.empty         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.rd_en_0          -
  Row   2:          1  intf.rd_en_1          (~intf.empty && ~intf.wr_en)
  Row   3:          1  intf.wr_en_0          (~intf.empty && intf.rd_en)
  Row   4:          1  intf.wr_en_1          intf.rd_en
  Row   5:          1  intf.empty_0          (intf.rd_en && ~intf.wr_en)
  Row   6:          1  intf.empty_1          (intf.rd_en && ~intf.wr_en)
```

```
----------------Focused Condition View-------------------
Line       74 Item    1  ((intf.rd_en && intf.wr_en) && intf.full)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
  intf.rd_en         Y
  intf.wr_en         Y
   intf.full         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.rd_en_0          -
  Row   2:          1  intf.rd_en_1          (intf.full && intf.wr_en)
  Row   3:          1  intf.wr_en_0          intf.rd_en
  Row   4:          1  intf.wr_en_1          (intf.full && intf.rd_en)
  Row   5:          1  intf.full_0           (intf.rd_en && intf.wr_en)
  Row   6:          1  intf.full_1           (intf.rd_en && intf.wr_en)

----------------Focused Condition View-------------------
Line       76 Item    1  ((intf.rd_en && intf.wr_en) && intf.empty)
Condition totals: 3 of 3 input terms covered = 100.0%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
  intf.rd_en         Y
  intf.wr_en         Y
  intf.empty         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  intf.rd_en_0          -
  Row   2:          1  intf.rd_en_1          (intf.empty && intf.wr_en)
  Row   3:          1  intf.wr_en_0          intf.rd_en
  Row   4:          1  intf.wr_en_1          (intf.empty && intf.rd_en)
  Row   5:          1  intf.empty_0          (intf.rd_en && intf.wr_en)
  Row   6:          1  intf.empty_1          (intf.rd_en && intf.wr_en)


Expression Coverage:
    Enabled Coverage           Active   Covered    Misses % Covered
    ----------------           ------      ----    ------ ---------
    FEC Expression Terms            0         0         0     100.0
FSM Coverage:
    Enabled Coverage           Active      Hits    Misses % Covered
    ----------------           ------      ----    ------ ---------
    FSMs                                                      100.0
        States                      0         0         0     100.0
        Transitions                 0         0         0     100.0
Toggle Coverage:
    Enabled Coverage           Active      Hits    Misses % Covered
    ----------------           ------      ----    ------ ---------
    Toggle Bins                   20        20         0     100.0
```

Toggle details:

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

Toggle Coverage for File FIFO.sv --

       Line                                  Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          6                              wr_ptr[2]           1           1      100.00
          6                              wr_ptr[1]           1           1      100.00
          6                              wr_ptr[0]           1           1      100.00
          6                              rd_ptr[2]           1           1      100.00
          6                              rd_ptr[1]           1           1      100.00
          6                              rd_ptr[0]           1           1      100.00
          7                               count[3]           1           1      100.00
          7                               count[2]           1           1      100.00
          7                               count[1]           1           1      100.00
          7                               count[0]           1           1      100.00

Total Node Count     =         10
Toggled Node Count   =         10
Untoggled Node Count =          0

Toggle Coverage      =      100.0% (20 of 20 bins)
```

Assertions:

```
DIRECTIVE COVERAGE:
---------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Count Status
                                         Unit   UnitType
---------------------------------------------------------------------------------------------
/\top#DUT /cvr_reset                     FIFO   Verilog  SVA  FIFO.sv(94)       870 Covered
/\top#DUT /wr_ack_cvr                    FIFO   Verilog  SVA  FIFO.sv(101)     5077 Covered
/\top#DUT /overflow_cvr                  FIFO   Verilog  SVA  FIFO.sv(107)      684 Covered
/\top#DUT /underflow_cvr                 FIFO   Verilog  SVA  FIFO.sv(113)      406 Covered
/\top#DUT /empty_cvr                     FIFO   Verilog  SVA  FIFO.sv(119)     1398 Covered
/\top#DUT /full_cvr                      FIFO   Verilog  SVA  FIFO.sv(125)     1051 Covered
/\top#DUT /almostfull_cvr                FIFO   Verilog  SVA  FIFO.sv(131)      862 Covered
/\top#DUT /almostempty_cvr               FIFO   Verilog  SVA  FIFO.sv(137)     1554 Covered
/\top#DUT /pointer_wraparound_cvr_write  FIFO   Verilog  SVA  FIFO.sv(145)      347 Covered
/\top#DUT /pointer_wraparound_cvr_read   FIFO   Verilog  SVA  FIFO.sv(153)       76 Covered
/\top#DUT /threshold_cvr                 FIFO   Verilog  SVA  FIFO.sv(160)     5077 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 11

ASSERTION RESULTS:
-------------------------------------------------------
Name                 File(Line)          Failure Pass
                                         Count   Count
-------------------------------------------------------
/\top#DUT /reset     FIFO.sv(93)               0     1
/\top#DUT /wr_ack_assert
                     FIFO.sv(100)              0     1
/\top#DUT /overflow_assert
                     FIFO.sv(106)              0     1
/\top#DUT /underflow_assert
                     FIFO.sv(112)              0     1
/\top#DUT /empty_assert
                     FIFO.sv(118)              0     1
/\top#DUT /full_assert
                     FIFO.sv(124)              0     1
/\top#DUT /almostfull_assert
                     FIFO.sv(130)              0     1
/\top#DUT /almostempty_assert
                     FIFO.sv(136)              0     1
/\top#DUT /pointer_wraparound_assert_write
                     FIFO.sv(144)              0     1
/\top#DUT /pointer_wraparound_assert_read
                     FIFO.sv(152)              0     1
/\top#DUT /threshold_assert
                     FIFO.sv(159)              0     1

Total Coverage By File (code coverage only, filtered view): 100.0%
```

Functional Coverage:

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

----------------------------------------------------------------------------------
------------
 TYPE /func_pkg/FIFO_coverage/cvr_grp                  100.0%        100    Covered
    covered/total bins:                                    12         12
    missing/total bins:                                     0         12
    % Hit:                                             100.0%        100
    Coverpoint cvr_grp::write_enable                   100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin wr_enable                                   10002          1    Covered
    Coverpoint cvr_grp::read_enable                    100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin rd_enable                                   10002          1    Covered
    Coverpoint cvr_grp::FULL                           100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Full                                        10002          1    Covered
    Coverpoint cvr_grp::EMPTY                          100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Empty                                       10002          1    Covered
    Coverpoint cvr_grp::ALMOSTFULL                     100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Almostfull                                  10002          1    Covered
    Coverpoint cvr_grp::ALMOSTEMPTY                    100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Almostempty                                 10002          1    Covered
    Coverpoint cvr_grp::OVERFLOW                       100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Overflow                                    10002          1    Covered
    Coverpoint cvr_grp::UNDERFLOW                      100.0%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                         100.0%        100
        bin Underflow                                   10002          1    Covered
    Coverpoint cvr_grp::Write_ack                      100.0%        100    Covered
```

```
        covered/total bins:                                1          1
        missing/total bins:                                0          1
        % Hit:                                         100.0%        100
        bin write_acknowledge                           10002          1    Covered
    Cross cvr_grp::rd_en_with_wr_enable_full [2]         0.0%        100    ZERO
        covered/total bins:                                0          0
        missing/total bins:                                0          0
        % Hit:                                         100.0%        100
        ignore_bin rd_en_with_active_full               10002
Occurred
    Cross cvr_grp::rd_en_with_wr_enable_empty          100.0%        100    Covered
        covered/total bins:                                1          1
        missing/total bins:                                0          1
        % Hit:                                         100.0%        100
        bin <rd_enable,wr_enable,Empty>                 10002          1    Covered
    Cross cvr_grp::rd_en_with_wr_enable_almostfull     100.0%        100    Covered
        covered/total bins:                                1          1
        missing/total bins:                                0          1
        % Hit:                                         100.0%        100
        bin <rd_enable,wr_enable,Almostfull>            10002          1    Covered
    Cross cvr_grp::rd_en_with_wr_enable_almostempty
                                                       100.0%        100    Covered
        covered/total bins:                                1          1
        missing/total bins:                                0          1
        % Hit:                                         100.0%        100
        bin <rd_enable,wr_enable,Almostempty>           10002          1    Covered
    Cross cvr_grp::rd_en_with_wr_enable_underflow [2]
                                                         0.0%        100    ZERO
        covered/total bins:                                0          0
        missing/total bins:                                0          0
        % Hit:                                         100.0%        100
        ignore_bin read_with_underflow                  10002
Occurred
    Cross cvr_grp::rd_en_with_wr_enable_overflow [2]
                                                         0.0%        100    ZERO
        covered/total bins:                                0          0
        missing/total bins:                                0          0
        % Hit:                                         100.0%        100
        ignore_bin write_with_overflow                  10002
Occurred
    Cross cvr_grp::rd_en_with_wr_enable_wr_ack [2]       0.0%        100    ZERO
        covered/total bins:                                0          0
        missing/total bins:                                0          0
        % Hit:                                         100.0%        100
        ignore_bin write_with_wr_ack                    10002
Occurred
 CLASS FIFO_coverage

 [2] - Does not contribute coverage as the item is empty

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
```

Name	Design Unit	Design UnitType	Lang	File(Line)	Count	Status
/top/DUT/cvr_reset	FIFO	Verilog	SVA	FIFO.sv(94)	870	Covered
/top/DUT/wr_ack_cvr	FIFO	Verilog	SVA	FIFO.sv(101)	5077	Covered
/top/DUT/overflow_cvr	FIFO	Verilog	SVA	FIFO.sv(107)	684	Covered
/top/DUT/underflow_cvr	FIFO	Verilog	SVA	FIFO.sv(113)	406	Covered
/top/DUT/empty_cvr	FIFO	Verilog	SVA	FIFO.sv(119)	1398	Covered
/top/DUT/full_cvr	FIFO	Verilog	SVA	FIFO.sv(125)	1051	Covered
/top/DUT/almostfull_cvr	FIFO	Verilog	SVA	FIFO.sv(131)	862	Covered
/top/DUT/almostempty_cvr	FIFO	Verilog	SVA	FIFO.sv(137)	1554	Covered
/top/DUT/pointer_wraparound_cvr_write	FIFO	Verilog	SVA	FIFO.sv(145)	347	Covered
/top/DUT/pointer_wraparound_cvr_read	FIFO	Verilog	SVA	FIFO.sv(153)	76	Covered
/top/DUT/threshold_cvr	FIFO	Verilog	SVA	FIFO.sv(160)	5077	Covered

TOTAL DIRECTIVE COVERAGE: 100.0% COVERS: 11

Verification Plan:

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	If rst_n high therefore all FIFO locations will be reseted & the output signals should be zero	we make directed testing for the reset signal at the beginning of testbench initialize the FIFO & then randomize it inside loops by constraining it to be active less often	No Functional Coverage for reset signal	Output Checked against zero in check_data function
FIFO_2	If write enable asserted and read enable deasserted we check the FIFO if it is not full so we will write data_in inside the FIFO	Randomization under constraints on the data_in & write enable	we include cover point for the wr_en signal & for full signal	Output Checked against check_data function
FIFO_3	If write enable deasserted and active read asserted we check the FIFO if it is not empty so we will read from the FIFO & get the value to be stored in data_out signal	Randomization under constraints for read enable signal	we include cover point for the rd_en signal & for empty signal	Output Checked against check_data function
FIFO_4	When read_enable and write_enable are asserted together, the outcome is determined by the status flags. If the full flag is asserted, the design gives priority to the read operation. If the empty flag is asserted, the design instead gives priority to the write operation. In cases where neither flag is asserted, the read and write operations are carried out concurrently.	NO Randomization occur for the output flags	We defined nine coverpoints to monitor the activity of read_enable, write_enable, and each of the output flags. In addition, we introduced seven cross coverage points to capture specific combinations (bins) that occur between the read and write enables and the output flags.	Output Checked against check_data function