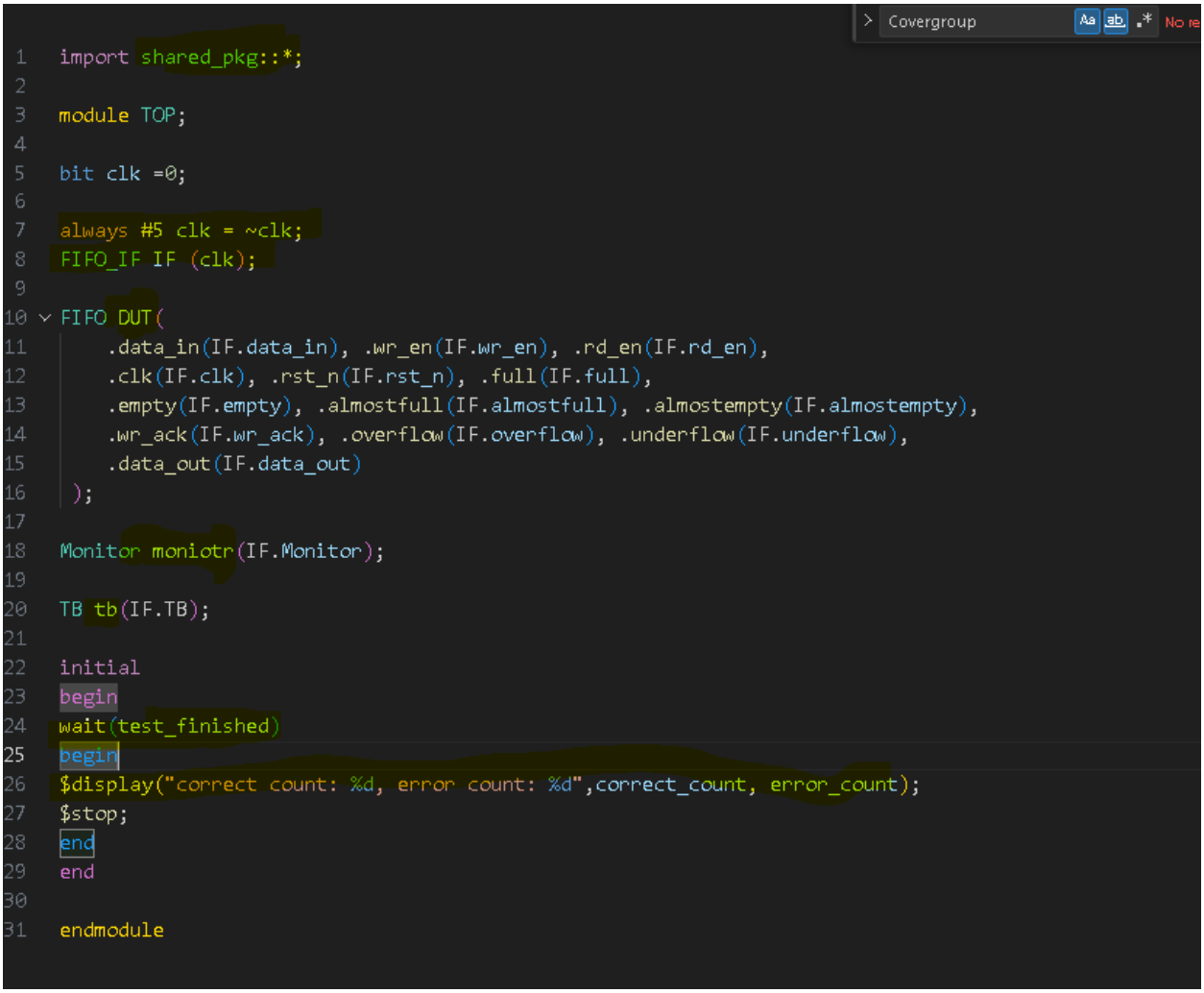
Synchronous FIFO

The top module will generate the clock, pass it to the interface, and the interface will be passed to the DUT, tb, and monitor modules. The tb will reset the DUT and then randomize the inputs. At the end of the test, the tb will assert a signal named test_finished. The signal will be defined as well as the error_count and correct_count in a shared package that you will create named shared_pkg.



The monitor module will do the following:

1. Create objects of 3 different classes (FIFO_transaction, FIFO_scoreboard, FIFO_coverage). These classes will be discussed later in the document

```
//`include "FIFO_IF.sv"
import FIFO_coverage_pkg::*;
import FIFO_transaction_pkg::*;
import FIFO_scoreboard_pkg::*;

module Monitor(FIFO_IF.Monitor IF);
FIFO_coverage fifo_coverage;
FIFO_transaction fifo_transaction;
FIFO_scoreboard fifo_scoreboard;

initial
begin
fifo_transaction = new();
fifo_coverage = new();
fifo_scoreboard = new();
forever begin
  @(posedge IF.clk)
   fifo_transaction.data_in  = IF.data_in;
```

2. It will have an initial block and inside it a forever loop that waits for negedge clock at the start of the loop and then sample the data of the interface and assign it to the data variables of the object of class FIFO_transaction. And then after that there will be fork join, where 2 processes will run, the first one is calling a method named sample_data of the object of class FIFO_coverage and the second process is calling a method named check_data of the object of class FIFO_scoreboard.

```
initial
begin
fifo_transaction = new();
fifo__coverage = new();
fifo_scoreboard = new();
forever begin
  @(posedge IF.clk)
    fifo_transaction.data_in  = IF.data_in;

    fifo_transaction.rst_n = IF.rst_n;

    fifo_transaction.wr_en = IF.wr_en;
    fifo_transaction.rd_en = IF.rd_en;


  @(negedge IF.clk) ;
    fifo_transaction.wr_ack = IF.wr_ack;

    fifo_transaction.overflow  = IF.overflow;
    fifo_transaction.underflow = IF.underflow;

    fifo_transaction.data_out = IF.data_out;
    fifo_transaction.full  = IF.full;
    fifo_transaction.empty = IF.empty;

    fifo_transaction.almostfull = IF.almostfull;
    fifo_transaction.almostempty = IF.almostempty;

    fifo_transaction.underflow  = IF.underflow;
  fork

     fifo__coverage.sample_data(fifo_transaction);
     begin

     fifo_scoreboard.check_data(fifo_transaction);
     end
  join

  end
```

2. Create a package in a new file that will have a class named FIFO_transaction

a. Inside of this class add the FIFO inputs and outputs as class variables of the class as well as adding 2 integers (RD_EN_ON_DIST & WR_EN_ON_DIST)

b. Add a constructor that takes 2 inputs and override the values of RD_EN_ON_DIST and WR_EN_ON_DIST, let the default of RD_EN_ON_DIST be 30 and WR_EN_ON_DIST be 70

c. Add the following 3 constraint blocks

1. Assert reset less often
2. Constraint the write enable to be high with distribution of the value WR_EN_ON_DIST 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

```
package FIFO_transaction_pkg;


class FIFO_transaction;
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

    int WR_EN_ON_DIST;
    int RD_EN_ON_DIST;

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


    logic [FIFO_WIDTH-1:0] data_out;

    logic wr_ack, overflow;
    logic full, empty, almostfull, almostempty, underflow;


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

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

endfunction

constraint cons {
    rst_n dist{1'b1:=98, 1'b0:=2};
    wr_en dist{1'b1:=WR_EN_ON_DIST,1'b0:=100-WR_EN_ON_DIST};
    rd_en dist{1'b1:=RD_EN_ON_DIST,1'b0:=100-RD_EN_ON_DIST};
};


endclass

endpackage
```

3. Create a package in a new file that will have a class for functional coverage collection named FIFO_coverage

a. Import the previous package (Add the import statement after the package declaration)

b. The class will have an object of the class FIFO_transaction named F_cvg_txn.

c. Create a covergroup. The coverage needed is 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.

d. Create a void function inside it named sample_data that takes one input named F_txn. This input is an object of class FIFO_transaction. This function will do the following

1. Assign F_txn to F_cvg_txn

2. Trigger the sampling of the covergroup using the .sample method

```
package FIFO_coverage_pkg;
import FIFO_transaction_pkg::*;

class FIFO_coverage;
FIFO_transaction F_cvg_txn;

//virtual FIFO_IF F_cvg_txn;
/*
 logic wr_ack, overflow;
 logic full, empty, almostfull, almostempty, underflow;
*/
covergroup cg;
option.per_instance = 1;
wr_en_cp: coverpoint F_cvg_txn.wr_en;
rd_en_cp: coverpoint F_cvg_txn.rd_en;

wr_ack_cp: coverpoint F_cvg_txn.wr_ack;

overflow_cp: coverpoint F_cvg_txn.overflow;
underflow_cp: coverpoint F_cvg_txn.underflow;

full_cp: coverpoint F_cvg_txn.full;
empty_cp: coverpoint F_cvg_txn.empty;

almostfull_cp: coverpoint F_cvg_txn.almostfull;
almostempty_cp: coverpoint F_cvg_txn.almostempty;

wr_rd_ack: cross wr_en_cp,rd_en_cp,wr_ack_cp
{
    ignore_bins ign_ack_without_wr = binsof(wr_en_cp) intersect {0} && binsof(wr_ack_cp) intersect{1};
}

wr_rd_overflow:cross wr_en_cp,rd_en_cp,overflow_cp
{
    ignore_bins ign_overflow_without_wr = binsof(wr_en_cp) intersect {0} && binsof(overflow_cp) intersect{1};
}

wr_rd_underflow:cross wr_en_cp,rd_en_cp,underflow_cp
{
    ignore_bins ign_underflow_without_rd = binsof(rd_en_cp) intersect {0} && binsof(underflow_cp) intersect{1
}

wr_rd_full:cross wr_en_cp,rd_en_cp,full_cp
```

```
wr__rd__full:cross wr_en_cp,rd_en_cp,full_cp
{
    ignore_bins ign_full_without_wr = binsof(wr_en_cp) intersect {0} && binsof(full_cp) intersect{1};
    ignore_bins ign_full_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsof(ful
}
wr__rd__empty:cross wr_en_cp,rd_en_cp,empty_cp
{
        ignore_bins ign_empty_without_rd = binsof(rd_en_cp) intersect {0} && binsof(empty_cp) intersect{1};
        ignore_bins ign_empty_with_wr_rd = binsof(wr_en_cp) intersect {1} && binsof(rd_en_cp) intersect {1} && binsc
}


wr__rd__almostfull:cross wr_en_cp,rd_en_cp,almostfull_cp;
wr__rd__almostempty:cross wr_en_cp,rd_en_cp,almostempty_cp;


endgroup

function new();
cg= new();
endfunction

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

function void stop();
cg.stop();
endfunction


endclass
```

4. Create a package in a new file named FIFO_scoreboard

a. Import the FIFO_transaction package.

b. Add variables for the data_out_ref, full_ref, etc. to be used in the reference_model function

c. Create a function named check_data that takes one input which of type FIFO_transaction

1. Inside this function, call another function named reference_model that you will create and pass to it the same object that you have received

2. Reference_model function will check the input values from the input object and assign values to the class properties data_out_ref, full_ref, etc.

3. After the reference_model function returns, you will compare the reference outputs calculated with the outputs of the object received. Increment the error_count or correct_count. Also, display a message if error occurs.

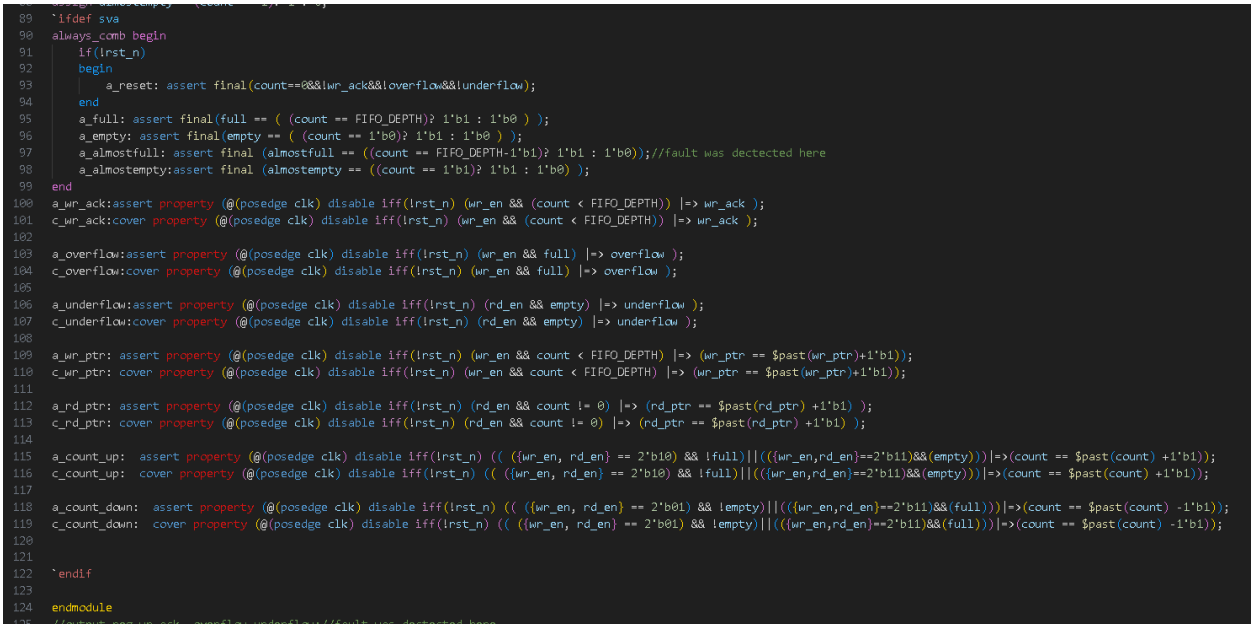
```
package FIFO_scoreboard_pkg;
import shared_pkg::*;
import FIFO_transaction_pkg::*;
class FIFO_scoreboard;
    localparam FIFO_WIDTH = 16;
    localparam FIFO_DEPTH = 8;
   localparam max_fifo_addr = $clog2(FIFO_DEPTH);


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

    logic [max_fifo_addr-1:0] wr_ptr_ref;
    logic [max_fifo_addr-1:0] rd_ptr_ref;
    logic [max_fifo_addr:0] count_ref;


    logic [FIFO_WIDTH-1 : 0] mem_ref [FIFO_DEPTH];


function new();
count_ref=0;


endfunction

function void reference_model(FIFO_transaction trans);

    if (!trans.rst_n) begin
        wr_ptr_ref = 0;
        wr_ack_ref=0;
        overflow_ref=0;//fault was dectected here
        data_out_ref= 0;//fault was dectected here

    end
    else if (trans.wr_en && count_ref < 8) begin
        mem_ref[wr_ptr_ref] = trans.data_in;
        wr_ack_ref = 1;
        wr_ptr_ref = wr_ptr_ref + 1;
        overflow_ref = 0;//fault was dectected here
    end
    else begin
```

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


    if (!trans.rst_n) begin
        rd_ptr_ref = 0;
        data_out_ref= 0;//fault was dectected here
        underflow_ref=0;//fault was dectected here
    end
    else if (trans.rd_en && count_ref != 0) begin
        data_out_ref = mem_ref[rd_ptr_ref];
        rd_ptr_ref = rd_ptr_ref + 1;
        underflow_ref=0;//fault was dectected here
    end
    else begin
    if(empty_ref&&trans.rd_en)
        underflow_ref=1;
    else
        underflow_ref=0;
    end


    if (!trans.rst_n) begin
        count_ref = 0;
        data_out_ref= 0;//fault was dectected here
    end
    else begin
        if  ( ({trans.wr_en, trans.rd_en} == 2'b10) && !full_ref) // if wren=1 and count_ref != 8
            count_ref = count_ref + 1;
        else if ( ({trans.wr_en, trans.rd_en} == 2'b01) && !empty_ref)// if rden=1 and count_ref != 0
            count_ref = count_ref - 1;
            else if ({trans.wr_en,trans.rd_en}==2'b11) begin//fault was dectected here
            if(full_ref)
```

```
            count_ref = count_ref - 1;
            else if ({trans.wr_en,trans.rd_en}==2'b11) begin//fault was dectected here
            if(full_ref)
            count_ref = count_ref-1;
            else if (empty_ref)
            count_ref = count_ref+1;
            end
    end

  full_ref = (count_ref == FIFO_DEPTH)? 1 : 0;
  empty_ref = (count_ref == 0)? 1 : 0;
  almostfull_ref = (count_ref == FIFO_DEPTH-1)? 1 : 0; //fault was dectected here
  almostempty_ref = (count_ref == 1)? 1 : 0;

  //assign underflow_ref = (empty_ref && rd_en)? 1 : 0; //fault was dectected here

  endfunction


  function void check_data(FIFO_transaction trans);

  reference_model(trans);

  assert(trans.data_out === data_out_ref && trans.wr_ack === wr_ack_ref && trans.overflow === overflow_ref && trans.fu
  && trans.almostempty === almostempty_ref && underflow_ref === trans.underflow)
  begin
      correct_count++;
  end
  else
  begin
  $error("test failed!");
      error_count++;
  end



  endfunction
```

```
    end


    endfunction


    endclass


    endpackage
```

5. Open the design file and add assertions to the FIFO inside the design file



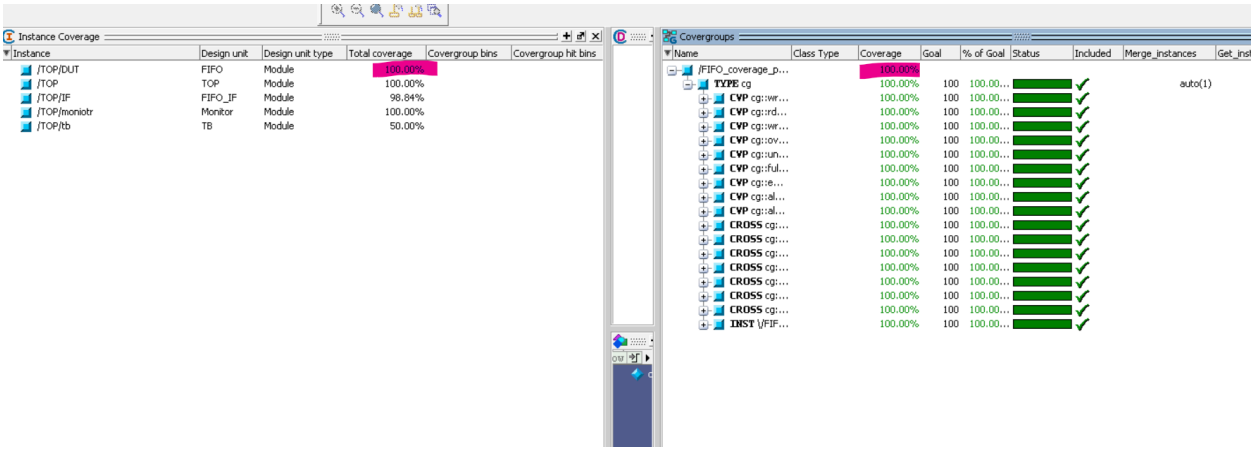
```
`ifdef sva
always_comb begin
    if(!rst_n)
    begin
        a_reset: assert final(count==0&&!wr_ack&&!overflow&&!underflow);
    end
    a_full: assert final(full == ( (count == FIFO_DEPTH)? 1'b1 : 1'b0 ) );
    a_empty: assert final(empty == ( (count == 1'b0)? 1'b1 : 1'b0 ) );
    a_almostfull: assert final (almostfull == ((count == FIFO_DEPTH-1'b1)? 1'b1 : 1'b0));//fault was dectected here
    a_almostempty:assert final (almostempty == ((count == 1'b1)? 1'b1 : 1'b0) );
end
a_wr_ack:assert property (@(posedge clk) disable iff(!rst_n) (wr_en && (count < FIFO_DEPTH)) |=> wr_ack );
c_wr_ack:cover property (@(posedge clk) disable iff(!rst_n) (wr_en && (count < FIFO_DEPTH)) |=> wr_ack );

a_overflow:assert property (@(posedge clk) disable iff(!rst_n) (wr_en && full) |=> overflow );
c_overflow:cover property (@(posedge clk) disable iff(!rst_n) (wr_en && full) |=> overflow );

a_underflow:assert property (@(posedge clk) disable iff(!rst_n) (rd_en && empty) |=> underflow );
c_underflow:cover property (@(posedge clk) disable iff(!rst_n) (rd_en && empty) |=> underflow );

a_wr_ptr: assert property (@(posedge clk) disable iff(!rst_n) (wr_en && count < FIFO_DEPTH) |=> (wr_ptr == $past(wr_ptr)+1'b1));
c_wr_ptr: cover property (@(posedge clk) disable iff(!rst_n) (wr_en && count < FIFO_DEPTH) |=> (wr_ptr == $past(wr_ptr)+1'b1));

a_rd_ptr: assert property (@(posedge clk) disable iff(!rst_n) (rd_en && count != 0) |=> (rd_ptr == $past(rd_ptr) +1'b1) );
c_rd_ptr: cover property (@(posedge clk) disable iff(!rst_n) (rd_en && count != 0) |=> (rd_ptr == $past(rd_ptr) +1'b1) );

a_count_up:  assert property (@(posedge clk) disable iff(!rst_n) (( ({wr_en, rd_en} == 2'b10) && !full)||(({wr_en,rd_en}==2'b11)&&(empty)))|=>(count == $past(count) +1'b1));
c_count_up:  cover property (@(posedge clk) disable iff(!rst_n) (( ({wr_en, rd_en} == 2'b10) && !full)||(({wr_en,rd_en}==2'b11)&&(empty)))|=>(count == $past(count) +1'b1));

a_count_down:  assert property (@(posedge clk) disable iff(!rst_n) (( ({wr_en, rd_en} == 2'b01) && !empty)||(({wr_en,rd_en}==2'b11)&&(full)))|=>(count == $past(count) -1'b1));
c_count_down:  cover property (@(posedge clk) disable iff(!rst_n) (( ({wr_en, rd_en} == 2'b01) && !empty)||(({wr_en,rd_en}==2'b11)&&(full)))|=>(count == $past(count) -1'b1));


`endif

endmodule
```

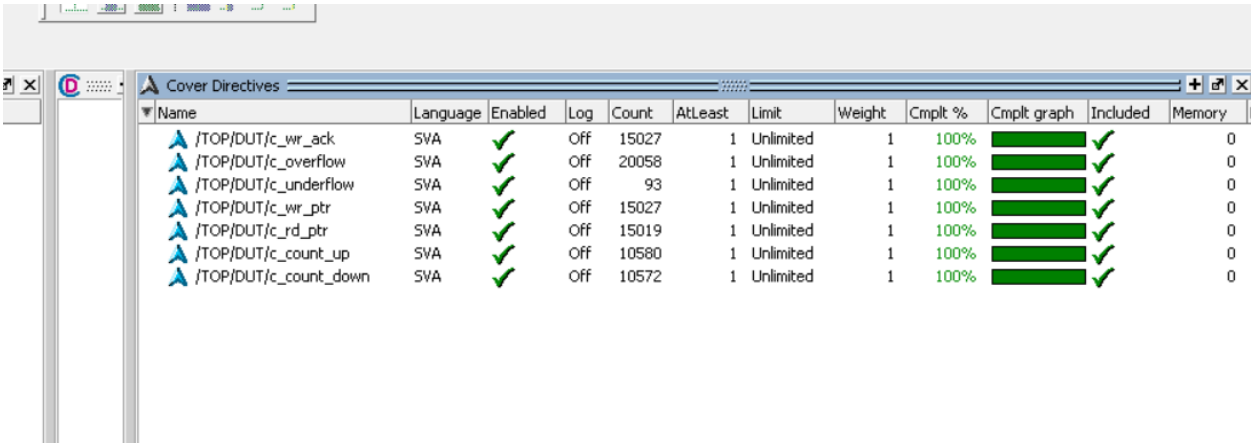
RUN.Do

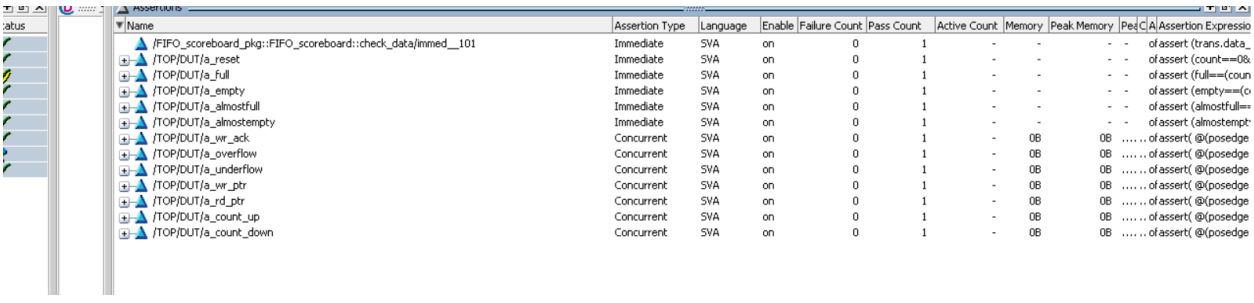
```
vlib work
vlog +fcover +cover +define+sva +acc -f source.list -covercells
vsim -voptargs=+acc -cover work.TOP

coverage save report.ucdb -onexit -du FIFO

add wave -noupdate /TOP/IF/FIFO_WIDTH
add wave -noupdate /TOP/IF/FIFO_DEPTH
add wave -noupdate /TOP/IF/clk
add wave -noupdate /TOP/IF/data_in
add wave -noupdate /TOP/IF/rst_n
add wave -noupdate /TOP/IF/wr_en
add wave -noupdate /TOP/IF/rd_en
add wave -noupdate /TOP/IF/data_out
add wave -noupdate /TOP/moniotr/fifo_scoreboard.data_out_ref
add wave -noupdate /TOP/IF/wr_ack
add wave -noupdate /TOP/IF/overflow
add wave -noupdate /TOP/IF/full
add wave -noupdate /TOP/IF/empty
add wave -noupdate /TOP/IF/almostfull
add wave -noupdate /TOP/IF/underflow
add wave -noupdate -color Gold /TOP/DUT/wr_ptr
add wave -noupdate -color Gold /TOP/DUT/rd_ptr
add wave -noupdate -color Gold /TOP/DUT/count
add wave -noupdate -color Salmon /TOP/IF/almostempty
add wave -noupdate /TOP/moniotr/fifo_scoreboard
add wave -noupdate /TOP/moniotr/fifo_scoreboard.underflow_ref
add wave -noupdate /TOP/DUT/mem
add wave -noupdate /TOP/moniotr/fifo_transaction
add wave -noupdate -expand -group Assertions /TOP/DUT/a_wr_ack
add wave -noupdate -expand -group Assertions /TOP/DUT/a_overflow
add wave -noupdate -expand -group Assertions /TOP/DUT/a_underflow
add wave -noupdate -expand -group Assertions /TOP/DUT/a_wr_ptr
add wave -noupdate -expand -group Assertions /TOP/DUT/a_rd_ptr
add wave -noupdate -expand -group Assertions /TOP/DUT/a_count_up
add wave -noupdate -expand -group Assertions /TOP/DUT/a_count_down
add wave -noupdate -expand -group Coverage /TOP/DUT/c_wr_ack
add wave -noupdate -expand -group Coverage /TOP/DUT/c_overflow
add wave -noupdate -expand -group Coverage /TOP/DUT/c_underflow
add wave -noupdate -expand -group Coverage /TOP/DUT/c_wr_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_rd_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_up
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_down
```

```
add wave -noupdate -expand -group Coverage /TOP/DUT/c_wr_ack
add wave -noupdate -expand -group Coverage /TOP/DUT/c_overflow
add wave -noupdate -expand -group Coverage /TOP/DUT/c_underflow
add wave -noupdate -expand -group Coverage /TOP/DUT/c_wr_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_rd_ptr
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_up
add wave -noupdate -expand -group Coverage /TOP/DUT/c_count_down

add wave -noupdate -group {Immediate Assertions} /TOP/DUT/a_reset
add wave -noupdate -group {Immediate Assertions} /TOP/DUT/a_full
add wave -noupdate -group {Immediate Assertions} /TOP/DUT/a_empty
add wave -noupdate -group {Immediate Assertions} /TOP/DUT/a_almostfull
add wave -noupdate -group {Immediate Assertions} /TOP/DUT/a_almostempty


run -all

coverage exclude -du FIFO -togglenode rst_n

#vcover report report.ucdb -details -annotate -all -output coverage_report.txt
```

Coverage Reports:



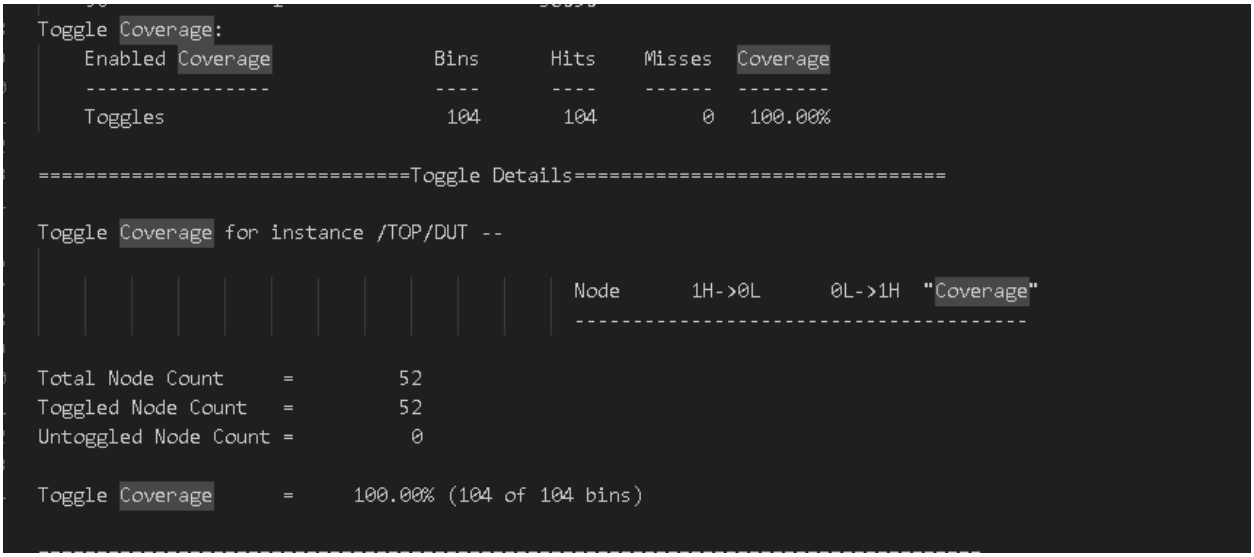




```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        29        29         0   100.00%

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

Branch Coverage for instance /TOP/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    25                                     50203     Count coming in to IF
    25              1                          2
    31              1                      15027
    37              1                      35174
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    39                                     35174     Count coming in to IF
```

```
140
141
142 ∨ Condition Coverage:
143       Enabled Coverage              Bins   Covered    Misses  Coverage
144       ----------------              ----      ----    ------  --------
145       Conditions                      20        20         0   100.00%
146
147   ================================Condition Details================================
148
149 ∨ Condition Coverage for instance /TOP/DUT --
150
```

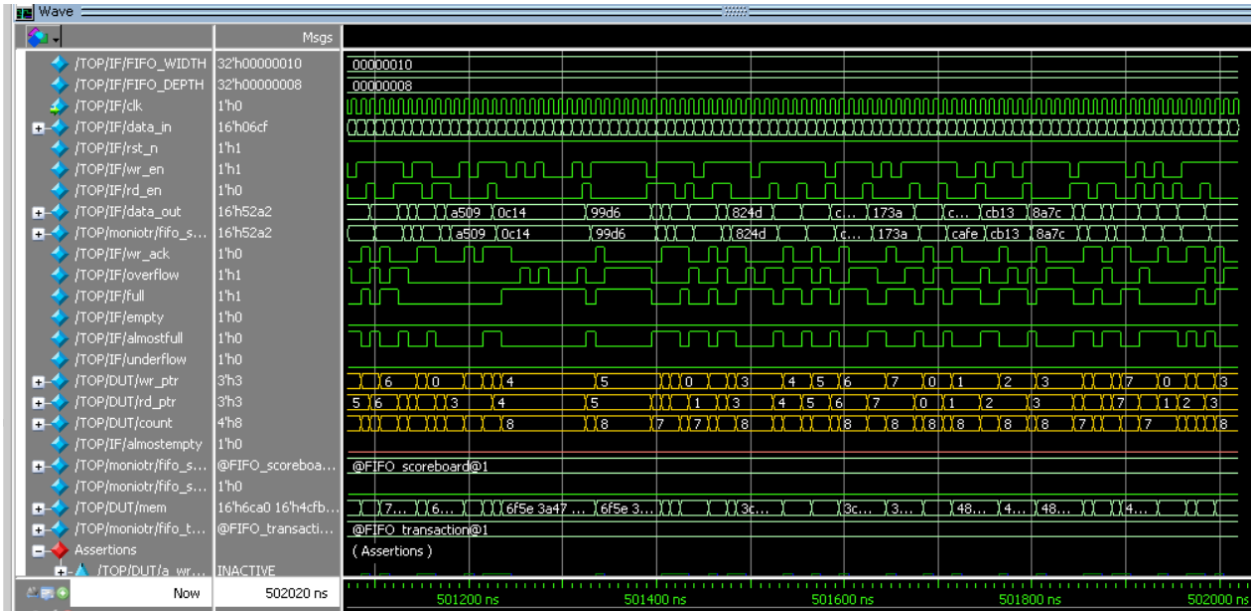
```
12   /TOP/DUT/c_count_down                    FIFO   Verilog  SVA  FIFO.sv(119)    10372 Covered
13 ∨ Statement Coverage:
14       Enabled Coverage              Bins      Hits    Misses  Coverage
15       ----------------              ----      ----    ------  --------
16       Statements                      33        33         0   100.00%
17
18   ================================Statement Details================================
19
20 ∨ Statement Coverage for instance /TOP/DUT --
21
22       Line         Item                      Count     Source
23       ----         ----                      -----     ------
24 ∨   File FIFO.sv
25       24              1                      50203
```

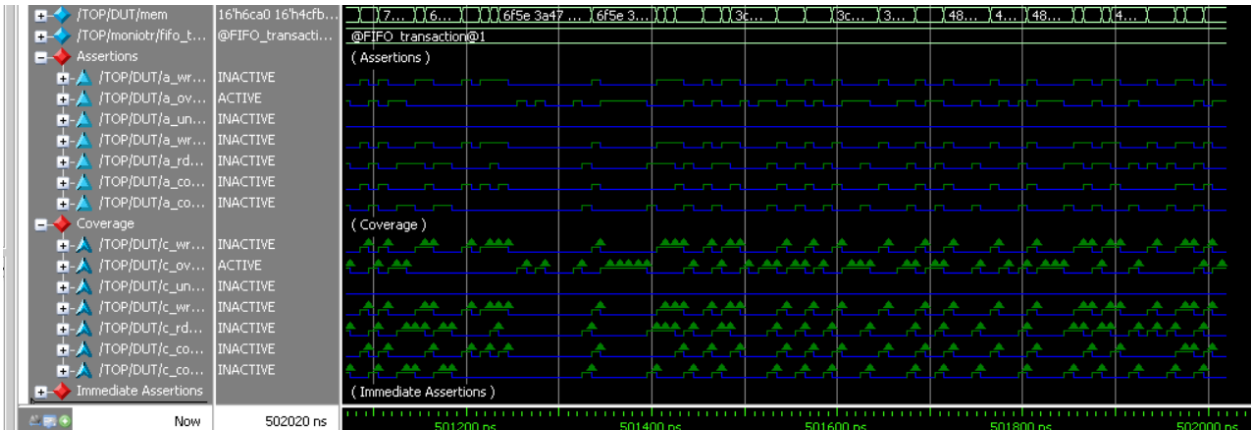

Toggle Coverage:
Enabled Coverage Bins Hits Misses Coverage
Toggles 104 104 0 100.00%
==========Toggle Details==========
Toggle Coverage for instance /TOP/DUT --
Node 1H->0L 0L->1H "Coverage"
Total Node Count = 52
Toggled Node Count = 52
Untoggled Node Count = 0
Toggle Coverage = 100.00% (104 of 104 bins)

```
================================================================================
=== Instance: /FIFO_coverage_pkg
=== Design Unit: work.FIFO_coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        16        na        na        na
            Covergroup Bins        62        62         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                             Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pkg/FIFO_coverage/cg            100.00%        100          -    Covered
    covered/total bins:                                  62         62          -
    missing/total bins:                                   0         62          -
    % Hit:                                           100.00%        100          -
    Coverpoint wr_en_cp                              100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                       100.00%        100          -
        bin auto[0]                                    15116          1          -    Covered
        bin auto[1]                                    35086          1          -    Covered
    Coverpoint rd_en_cp                              100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                       100.00%        100          -
        bin auto[0]                                    35090          1          -    Covered
        bin auto[1]                                    15112          1          -    Covered
    Coverpoint wr_ack_cp                             100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
        % Hit:                                       100.00%        100          -
        bin auto[0]                                    35175          1          -    Covered
        bin auto[1]                                    15027          1          -    Covered
    Coverpoint overflow_cp                           100.00%        100          -    Covered
        covered/total bins:                                2          2          -
        missing/total bins:                                0          2          -
```

Results:

```
# Loading work.FIFO_coverage_pkg(fast)
# Loading work.monitor_sv_unit(fast)
# Loading work.Monitor(fast)
# Loading work.TB(fast)
# correct count:       50202, error count:           0
# ** Note: $stop    : TOP.sv(27)
#    Time: 502020 ns  Iteration: 2  Instance: /TOP
# Break in Module TOP at TOP.sv line 27
VSIM 3> coverage report -detail -output cov.txt

VSIM 4>
```


Wave
Msgs
/TOP/IF/FIFO_WIDTH
32'h00000010
/TOP/IF/FIFO_DEPTH
32'h00000008
/TOP/IF/clk
1'h0
/TOP/IF/data_in
16'h06cf
/TOP/IF/rst_n
1'h1
/TOP/IF/wr_en
1'h1
/TOP/IF/rd_en
1'h0
/TOP/IF/data_out
16'h52a2
/TOP/moniotr/fifo_s...
16'h52a2
/TOP/IF/wr_ack
1'h0
/TOP/IF/overflow
1'h1
/TOP/IF/full
1'h1
/TOP/IF/empty
1'h0
/TOP/IF/almostfull
1'h0
/TOP/IF/underflow
1'h0
/TOP/DUT/wr_ptr
3'h3
/TOP/DUT/rd_ptr
3'h3
/TOP/DUT/count
4'h8
/TOP/IF/almostempty
1'h0
/TOP/moniotr/fifo_s...
@FIFO_scoreboa...
/TOP/moniotr/fifo_s...
1'h0
/TOP/DUT/mem
16'h6ca0 16'h4cfb...
/TOP/moniotr/fifo_t...
@FIFO_transacti...
Assertions
/TOP/DUT/a_wr...
INACTIVE
00000010
00000008
@FIFO_scoreboard@1
@FIFO_transaction@1
(Assertions)
Now
502020 ns
501200 ns
501400 ns
501600 ns
501800 ns
502000 ns


/TOP/DUT/mem
16'h6ca0 16'h4cfb...
/TOP/moniotr/fifo_t...
@FIFO_transacti...
Assertions
/TOP/DUT/a_wr...
INACTIVE
/TOP/DUT/a_ov...
ACTIVE
/TOP/DUT/a_un...
INACTIVE
/TOP/DUT/a_wr...
INACTIVE
/TOP/DUT/a_rd...
INACTIVE
/TOP/DUT/a_co...
INACTIVE
/TOP/DUT/a_co...
INACTIVE
Coverage
/TOP/DUT/c_wr...
INACTIVE
/TOP/DUT/c_ov...
ACTIVE
/TOP/DUT/c_un...
INACTIVE
/TOP/DUT/c_wr...
INACTIVE
/TOP/DUT/c_rd...
INACTIVE
/TOP/DUT/c_co...
INACTIVE
/TOP/DUT/c_co...
INACTIVE
Immediate Assertions
@FIFO_transaction@1
(Assertions)
(Coverage)
(Immediate Assertions)
Now
502020 ns