System Verilog

SVA

Assignment 4

By: Magdy Ahmed Abbas Abdelhamid

Question 1 & 2

SVA

SVA Assignment

Q1. Write assertions for the following:

1) Write assert statement for signal a is high in a positive edge of a clock then signal b should be high after 2 clock cycles
2) Write a sequence s11a, where signal a is high and signal b is high and then signal c is high 1 to 3 clock cycle later
3) Write a sequence s11b, after 2 positive clock edges, signal b should be low
4) Write a property that check that if sequence s11a occurs then s11b will occur
5) Write a property that checks on a positive clock edge, if signal a is high then signal b should be eventually high from the next clock cycle, then after that signal c should be eventually high starting the same clock cycle in which signal b was high

Q2. Write an assertion to make sure that 3to8 decoder output is only one bit high with every clock cycle.

System Verilog Code

```
// Question 1
  // Requirement 1
  assert property (@(posedge clk) (a |-> ##2 b));   // OR
  assert property (@(posedge clk) (a |=> ##1 b));
  // Requirement 2
  sequence s11a;
    (a && b && c[* 1:3]);
  endsequence
  // Requirement 3
  sequence s11b;
    (##2 (b == 0));
  endsequence
  // Requirement 4
  property p1;
    @(posedge clk) s11a |-> s11b;
  endproperty
  // Requirement 5
  property p2;
    @(posedge clk) (a |=> b[-> 1] |-> c[-> 1]);
  endproperty
// Question 2
  assert property (@(posedge clk) $onehot(out));
```

Question 3

Counter

Q3. Verify the functionality of counter provided in the assignment 3 – extra pdf by adding assertions to the design. The reset of the design has changed to asynchronous.

Assertion to be added in a new file based on the following specs to be checked:

1. When the load control signal is active, then the dout has the value of the din
2. When the load control signal is not active and the enable is off then the dout does not change
3. When the load control signal is not active and the enable is active, and the up_down is high then the dout is incremented.
4. When the load control signal is not active and the enable is active, and the up_down is low then the dout is decremented.
5. When the asynchronous reset is asserted then the counter output is tied to low at the same instant. Note that you can use assert final to sample the new value of a signal.
6. max_count output is high when the counter output is maximum.
7. zero output is high when the counter output is zero.

<u>Requirements for Q3:</u>

1- Create an interface with DUT and TEST modports
2- Create a top module where clock gen will take place and connections of interface
3- Create SVA module and bind it in the top module
4- Create the testbench as instructed in assignment 3 – extra pdf

5- Adjust your verification requirements document in the the functionality check column, add if the requirement is checked against golden model, assertion or both. In case of assertion then mention its label used in your SVA module

Counter Design

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: Counter Design
//
////////////////////////////////////////////////////////////////////////////////

module counter (counter_if.DUT c_if);

always @(posedge c_if.clk or negedge c_if.rst_n) begin
   if (!c_if.rst_n)
      c_if.count_out <= 0;
   else if (!c_if.load_n)
      c_if.count_out <= c_if.data_load;
   else if (c_if.ce)
      if (c_if.up_down)
         c_if.count_out <= c_if.count_out + 1;
      else
         c_if.count_out <= c_if.count_out - 1;
end

assign c_if.max_count = (c_if.count_out == {c_if.WIDTH{1'b1}})? 1:0;
assign c_if.zero = (c_if.count_out == 0)? 1:0;

endmodule
```

Top Module

```
module top;

bit clk;

initial begin
   clk  =  0;
   forever
   #25 clk  =  ~clk;
end
// interface object
counter_if c_if(clk);

// DUT instantiation
counter dut(c_if);

// testbench instantiation
counter_tb test(c_if);

// binding the assertions
bind counter SVA T1(c_if);
endmodule
```

Interface

```
interface counter_if (input bit clk);

// parameter declaration
parameter WIDTH  =  4;

// input & output declaration
logic rst_n, load_n, up_down, ce;
logic [WIDTH – 1: 0] data_load, count_out;
logic max_count, zero;

modport DUT (input clk, rst_n, load_n, up_down, ce, data_load, output count_out, max_count, zero);

modport TEST (output rst_n, load_n, up_down, ce, data_load, input clk, count_out, max_count, zero);

modport SVA (input clk, rst_n, load_n, up_down, ce, data_load, count_out, max_count, zero);

endinterface
```

Package

```
package counter_pkg;

parameter WIDTH = 4;

class counter_transaction;

rand bit rst_n, load_n, up_down, ce;
rand bit [WIDTH - 1: 0] data_load, count_out;

constraint rst_con { rst_n dist {0:= 10, 1:= 90}; }
constraint load_con { load_n dist {0:= 30, 1:= 70}; }
constraint ce_con { ce dist {1:= 70, 0:= 30}; }

covergroup cg ;

load_n_cp: coverpoint load_n {
                        //option. per_instance = 1;
                        option. weight = 0;
                       }
data_load_cp: coverpoint data_load {
                           //option. per_instance = 1;
                           option. weight = 0;
                           }
load_cross: cross load_n_cp, data_load_cp
{ ignore_bins bins_load_n_high = binsof(load_n_cp) intersect {1}; }

count_bins_up_cp: coverpoint count_out iff (rst_n && ce && up_down);
count_bins_up_trans_cp: coverpoint count_out iff (rst_n && ce && up_down)
{ bins max_zero_trans = ({WIDTH{1'b1}} => 0); }

count_bins_dn_cp: coverpoint count_out iff (rst_n && ce && ! up_down);
count_bins_dn_trans_cp: coverpoint count_out iff (rst_n && ce && ! up_down)
{ bins zero_max_trans = ( 0 => {WIDTH{1'b1}} ); }
endgroup

function new;
   cg = new();
endfunction
endclass
endpackage
```

Testbench

```
import counter_pkg: :*;

module counter_tb(counter_if. TEST c_if);

logic max_count_exp, zero_exp;
logic [WIDTH – 1: 0] count_out_exp;
counter_transaction obj  =  new();

// counter declaration
int error_count;
int correct_count;

// Stimulus Generation

initial begin
   error_count  = 0;
   correct_count = 0;
   repeat(10000) begin
      assert(obj. randomize());
      c_if. rst_n  =  obj. rst_n;
      c_if. load_n  =  obj. load_n;
      c_if. up_down  =  obj. up_down;
      c_if. ce  =  obj. ce;
      c_if. data_load  =  obj. data_load;
      check_result(obj);
      obj. count_out  =  c_if. count_out;
      obj. cg. sample();
   end
   $display(" error count is %0d and correct count is %0d", error_count, correct_count);
   $stop;
end

// Checker
task check_result(input counter_transaction obj_c);
   golden_model(obj_c);
      @(negedge c_if. clk);
      if ( (max_count_exp ! ==  c_if. max_count) || (zero_exp! =
                  =  c_if. zero) || (count_out_exp ! ==  c_if. count_out) )
         error_count + +;
      else
         correct_count + +;
endtask
```

```
// Golden Model
task golden_model(input counter_transaction obj_g);
   if (! obj_g. rst_n)
      count_out_exp  =  0;
   else if (! obj_g. load_n)
      count_out_exp  =  obj_g. data_load;
   else if (obj_g. ce) begin
       if (obj_g. up_down)
         count_out_exp  =  count_out_exp  +  1;
      else
         count_out_exp  =  count_out_exp  −  1;
      end
   max_count_exp  =  (count_out_exp  ==  {WIDTH{1'b1}})?  1: 0;
   zero_exp  =  (count_out_exp  ==  0)?  1: 0;
endtask
endmodule
```

SVA

```
module SVA (counter_if. SVA c_if);
// first Assertion
property p1;
@(posedge c_if. clk) disable iff(! c_if. rst_n) (! c_if. load_n) | =>
                                    (c_if. count_out  ===  $past(c_if. data_load));
endproperty
data_load_assertion: assert property(p1);
cover_data_load: cover property(p1);
// second Assertion
property p2;
@(posedge c_if. clk) disable iff(! c_if. rst_n) (c_if. load_n && ! c_if. ce) | =>  $stable(c_if. count_out);
endproperty
not_active_assertion: assert property(p2);
cover_not_active: cover property(p2);
// Third Assertion
property p3;
@(posedge c_if. clk) disable iff(! c_if. rst_n) (c_if. load_n && c_if. ce && c_if. up_down)
                        | =>  (c_if. count_out  ===  $past(c_if. count_out)  +  4'b0001);
endproperty
increment_assertion: assert property(p3);
cover_increment: cover property(p3);
// 4th Assertion
property p4;
@(posedge c_if. clk) disable iff(! c_if. rst_n) (c_if. load_n && c_if. ce && ! c_if. up_down)
                        | =>  (c_if. count_out  ===  $past(c_if. count_out)  −  4'b0001);
endproperty
```

```
decrement_assertion: assert property(p4);
cover_decrement: cover property(p4);
// 5th Assertion (using immediate assertion)
always @(*) begin
   if(! c_if. rst_n)
   reset_assertion: assert final (c_if. count_out  ===  0);
end
// 6th assertion
property p6;
@(posedge c_if. clk) (c_if. count_out  ===  {c_if. WIDTH{1'b1}}) |−>  c_if. max_count;
endproperty
max_value_assertion: assert property(p6);
max_value_cover: cover property(p6);
// 7th assertion
property p7;
@(posedge c_if. clk) (c_if. count_out  ===  0) |−>  c_if. zero;
endproperty
zero_assertion: assert property(p7);
zero_cover: cover property(p7);
endmodule
```

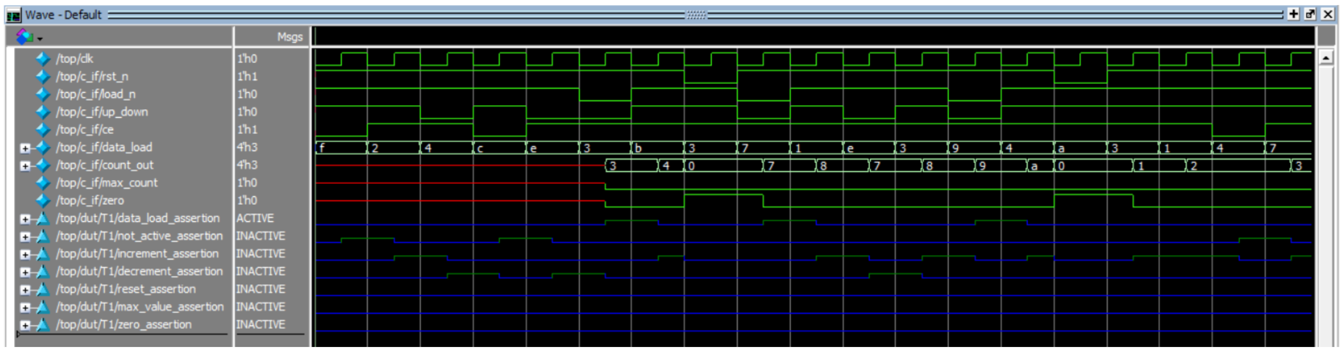
Do File

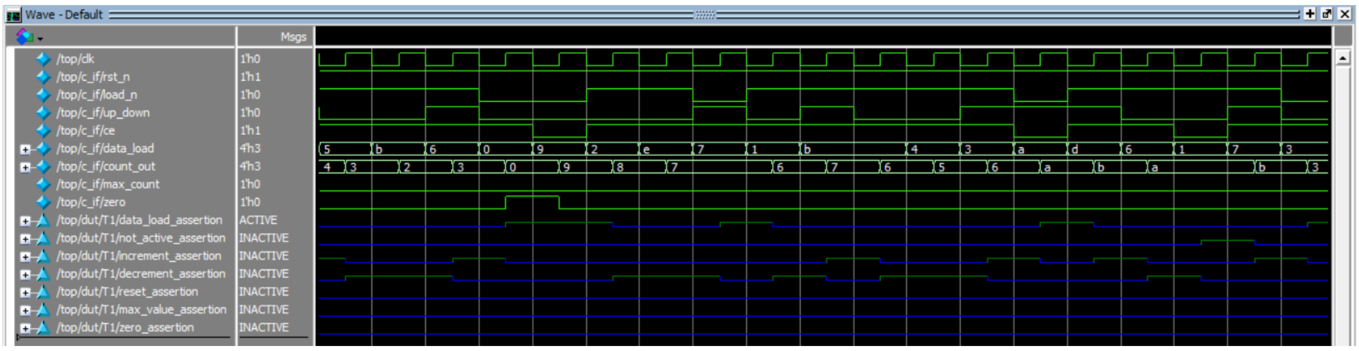
```
vlib work
vlog counter. sv package. sv SVA. sv testbench. sv interface. sv TOP. sv  + cover
vsim  − voptargs = +acc work. top  − cover
add wave  *
add wave  − position insertpoint  \
sim:/top/c_if/rst_n \
sim:/top/c_if/load_n \
sim:/top/c_if/up_down \
sim:/top/c_if/ce \
sim:/top/c_if/data_load \
sim:/top/c_if/count_out \
sim:/top/c_if/max_count \
sim:/top/c_if/zero
add wave /top/dut/T1/data_load_assertion /top/dut/T1/not_active_assertion /top/dut/T1
                    /increment_assertion /top/dut/T1/decrement_assertion /top/dut/T1/reset_assertion
                    /top/dut/T1/max_value_assertion /top/dut/T1/zero_assertion
coverage save testbench. ucdb  − onexit
run  − all
```

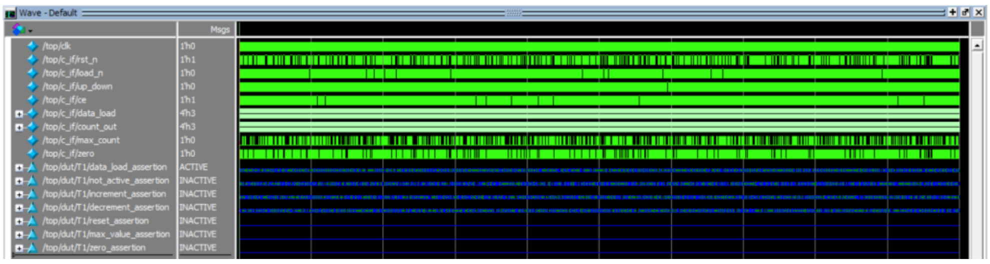
Counter Verification Requirements

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
COUNTER_1	When the reset is asserted, the output counter value should be low, zero output is high and max_count output is 0	Directed at the start of the sim, then randomized with constraint that it is mostly off	-	Output Checked against golden model & also check it using reset_assertion which is the 5th assertion in the SVA module
COUNTER_2	When the load is asserted (active low), the output count_out should take the value of the data_load input	Randomization under constraints on the load signal to be off 70% of the time	Included in coverpoints for load and data load with weight option = 0. Cross coverage between the data load when load is asserted	Output Checked against golden model & also check it using data_load_assertion which is the 1st assertion in the SVA module
COUNTER_3	When the enable is asserted (considering that the reset & load signals are deasserted), up_down is high then the output count_out will be incremented	Randomization under constraints on the enable signal to be active 70% of the time	Included in a coverpoint for count out if the reset is deasserted, enable is active and up_down is high. Another coverpoint for count out if the reset is deasserted, enable is active and up_down is high with transition bin to check when overflow occurs	Output Checked against golden model & also check it using increment_assertion which is the 3rd assertion in the SVA module
COUNTER_4	When the enable is asserted, up_down is low then the output count_out will be decremented	Randomization under constraints on the enable signal to be active 70% of the time	Included in a coverpoint for count out if the reset is deasserted, enable is active and up_down is low. Another coverpoint for count out if the reset is deasserted, enable is active and up_down is low with transition bin to check when underflow occurs	Output Checked against golden model & also check it using decrement_assertion which is the 4th assertion in the SVA module

Waveform

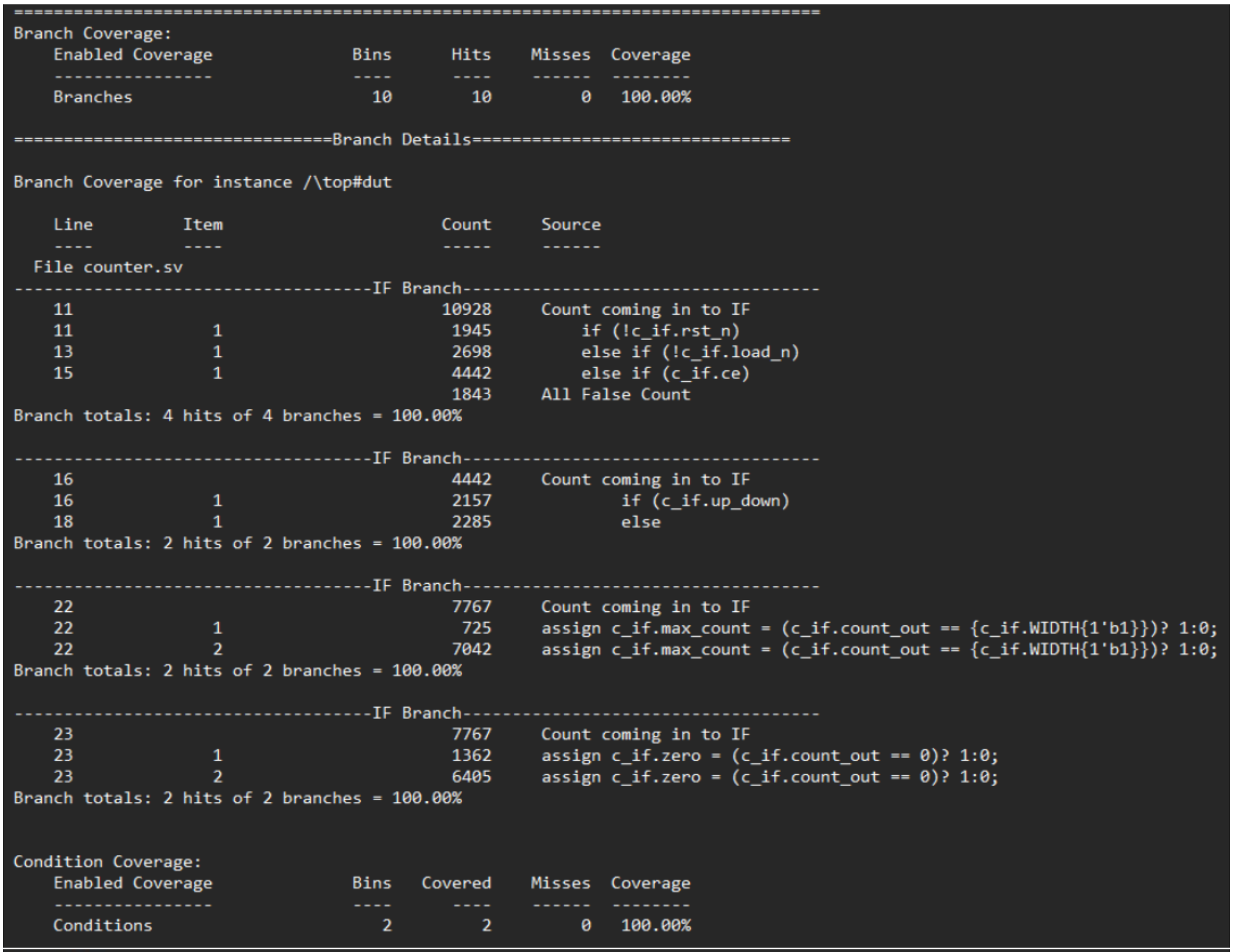

```
Wave - Default                               Msgs
/top/clk                                     1'h0
/top/c_if/rst_n                              1'h1
/top/c_if/load_n                             1'h0
/top/c_if/up_down                            1'h0
/top/c_if/ce                                 1'h1
/top/c_if/data_load                          4'h3
/top/c_if/count_out                          4'h3
/top/c_if/max_count                          1'h0
/top/c_if/zero                               1'h0
/top/dut/T1/data_load_assertion              ACTIVE
/top/dut/T1/not_active_assertion             INACTIVE
/top/dut/T1/increment_assertion              INACTIVE
/top/dut/T1/decrement_assertion              INACTIVE
/top/dut/T1/reset_assertion                  INACTIVE
/top/dut/T1/max_value_assertion              INACTIVE
/top/dut/T1/zero_assertion                   INACTIVE
```


```
Wave - Default                               Msgs
/top/clk                                     1'h0
/top/c_if/rst_n                              1'h1
/top/c_if/load_n                             1'h0
/top/c_if/up_down                            1'h0
/top/c_if/ce                                 1'h1
/top/c_if/data_load                          4'h3
/top/c_if/count_out                          4'h3
/top/c_if/max_count                          1'h0
/top/c_if/zero                               1'h0
/top/dut/T1/data_load_assertion              ACTIVE
/top/dut/T1/not_active_assertion             INACTIVE
/top/dut/T1/increment_assertion              INACTIVE
/top/dut/T1/decrement_assertion              INACTIVE
/top/dut/T1/reset_assertion                  INACTIVE
/top/dut/T1/max_value_assertion              INACTIVE
/top/dut/T1/zero_assertion                   INACTIVE
```

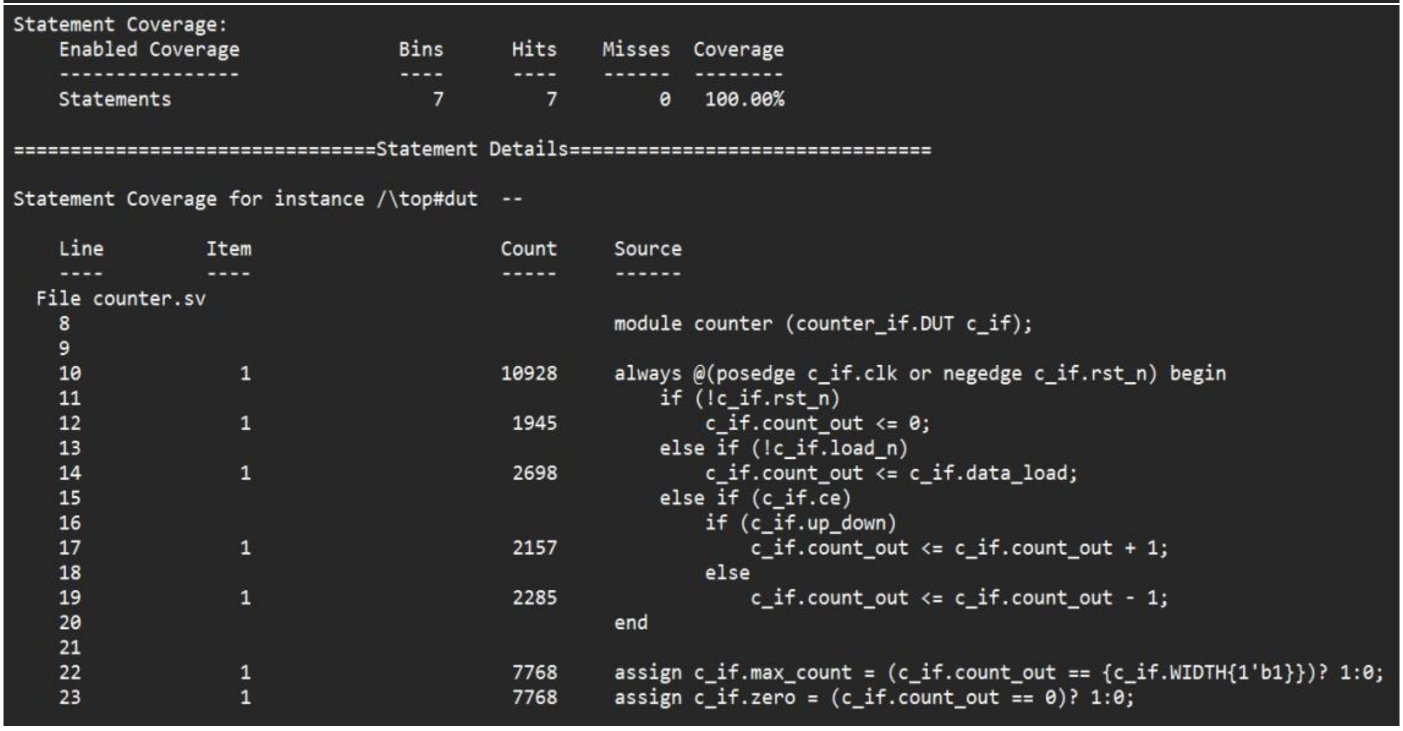

```
Wave - Default                               Msgs
/top/clk                                     1'h0
/top/c_if/rst_n                              1'h1
/top/c_if/load_n                             1'h0
/top/c_if/up_down                            1'h0
/top/c_if/ce                                 1'h1
/top/c_if/data_load                          4'h3
/top/c_if/count_out                          4'h3
/top/c_if/max_count                          1'h0
/top/c_if/zero                               1'h0
/top/dut/T1/data_load_assertion              ACTIVE
/top/dut/T1/not_active_assertion             INACTIVE
/top/dut/T1/increment_assertion              INACTIVE
/top/dut/T1/decrement_assertion              INACTIVE
/top/dut/T1/reset_assertion                  INACTIVE
/top/dut/T1/max_value_assertion              INACTIVE
/top/dut/T1/zero_assertion                   INACTIVE
```

```
# error count is 0 and correct count is 10000
```

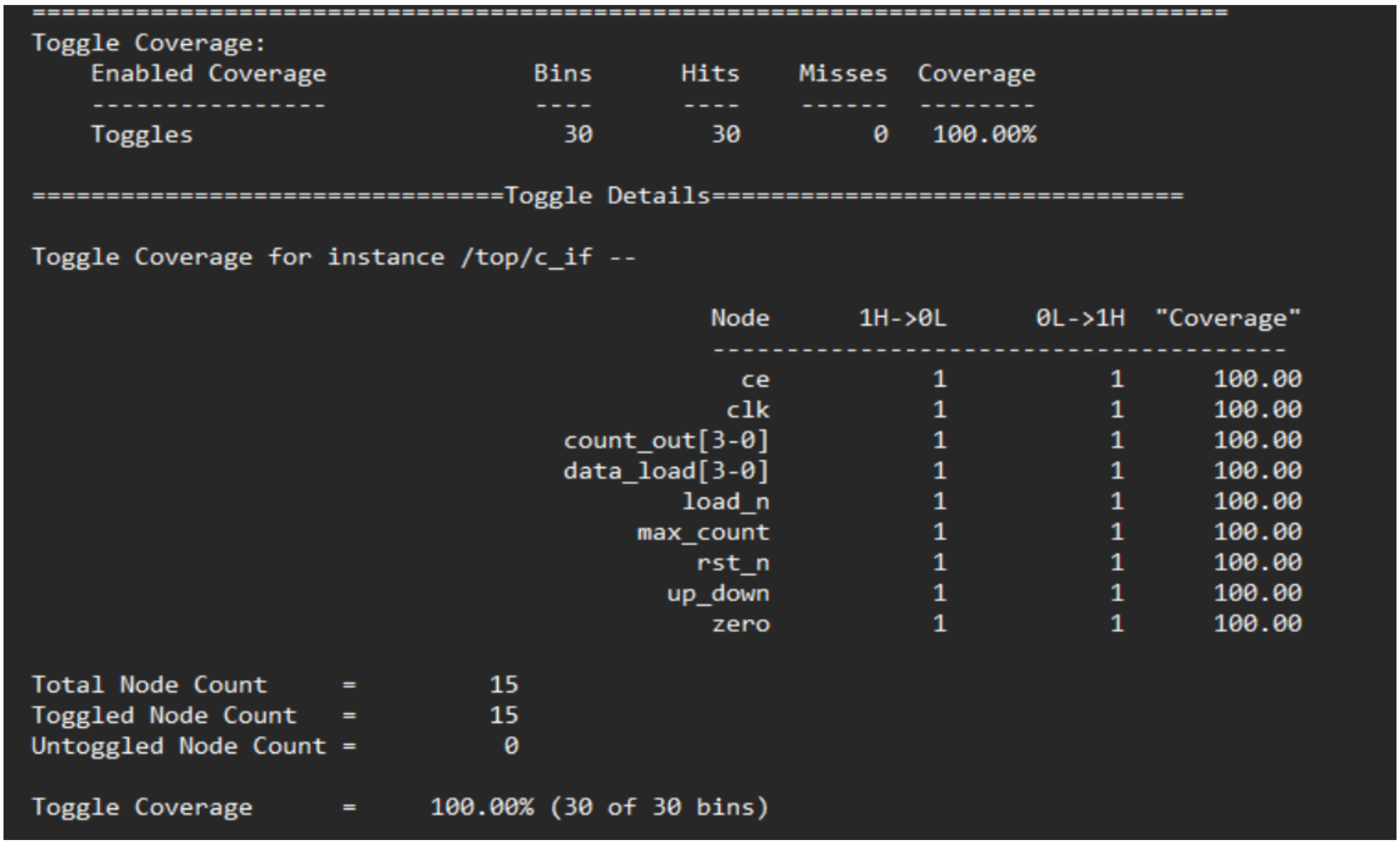
Code Coverage



```
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for instance /\top#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File counter.sv
------------------------------------IF Branch------------------------------------
    11                                     10928     Count coming in to IF
    11              1                       1945         if (!c_if.rst_n)
    13              1                       2698         else if (!c_if.load_n)
    15              1                       4442         else if (c_if.ce)
                                            1843     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    16                                      4442     Count coming in to IF
    16              1                       2157             if (c_if.up_down)
    18              1                       2285             else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    22                                      7767     Count coming in to IF
    22              1                        725     assign c_if.max_count = (c_if.count_out == {c_if.WIDTH{1'b1}})? 1:0;
    22              2                       7042     assign c_if.max_count = (c_if.count_out == {c_if.WIDTH{1'b1}})? 1:0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    23                                      7767     Count coming in to IF
    23              1                       1362     assign c_if.zero = (c_if.count_out == 0)? 1:0;
    23              2                       6405     assign c_if.zero = (c_if.count_out == 0)? 1:0;
Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       2         2         0   100.00%
```

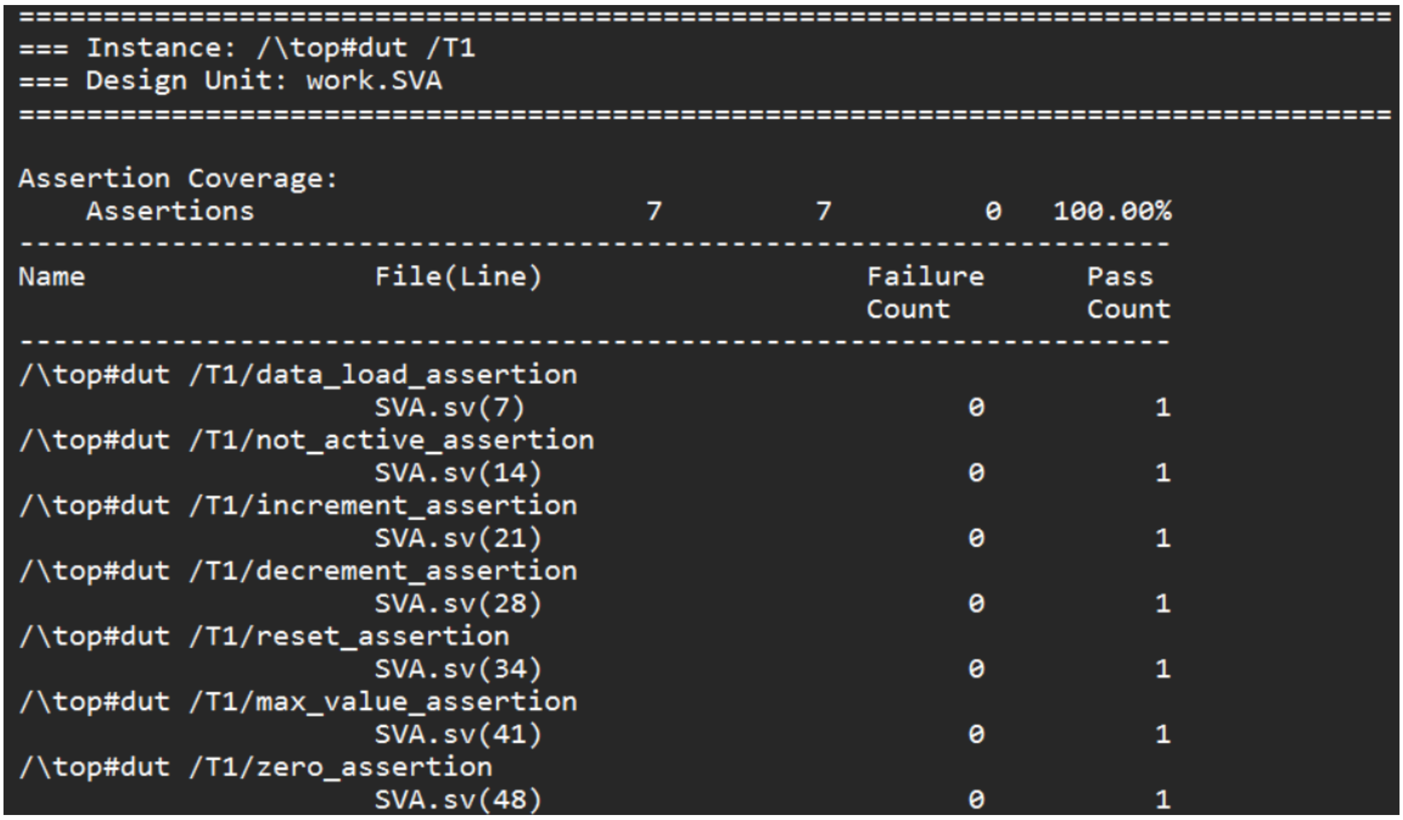


```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       7         7         0   100.00%

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

Statement Coverage for instance /\top#dut  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File counter.sv
    8                                                module counter (counter_if.DUT c_if);
    9
    10              1                      10928     always @(posedge c_if.clk or negedge c_if.rst_n) begin
    11                                                   if (!c_if.rst_n)
    12              1                       1945             c_if.count_out <= 0;
    13                                                   else if (!c_if.load_n)
    14              1                       2698             c_if.count_out <= c_if.data_load;
    15                                                   else if (c_if.ce)
    16                                                       if (c_if.up_down)
    17              1                       2157                 c_if.count_out <= c_if.count_out + 1;
    18                                                       else
    19              1                       2285                 c_if.count_out <= c_if.count_out - 1;
    20                                               end
    21
    22              1                       7768     assign c_if.max_count = (c_if.count_out == {c_if.WIDTH{1'b1}})? 1:0;
    23              1                       7768     assign c_if.zero = (c_if.count_out == 0)? 1:0;
```



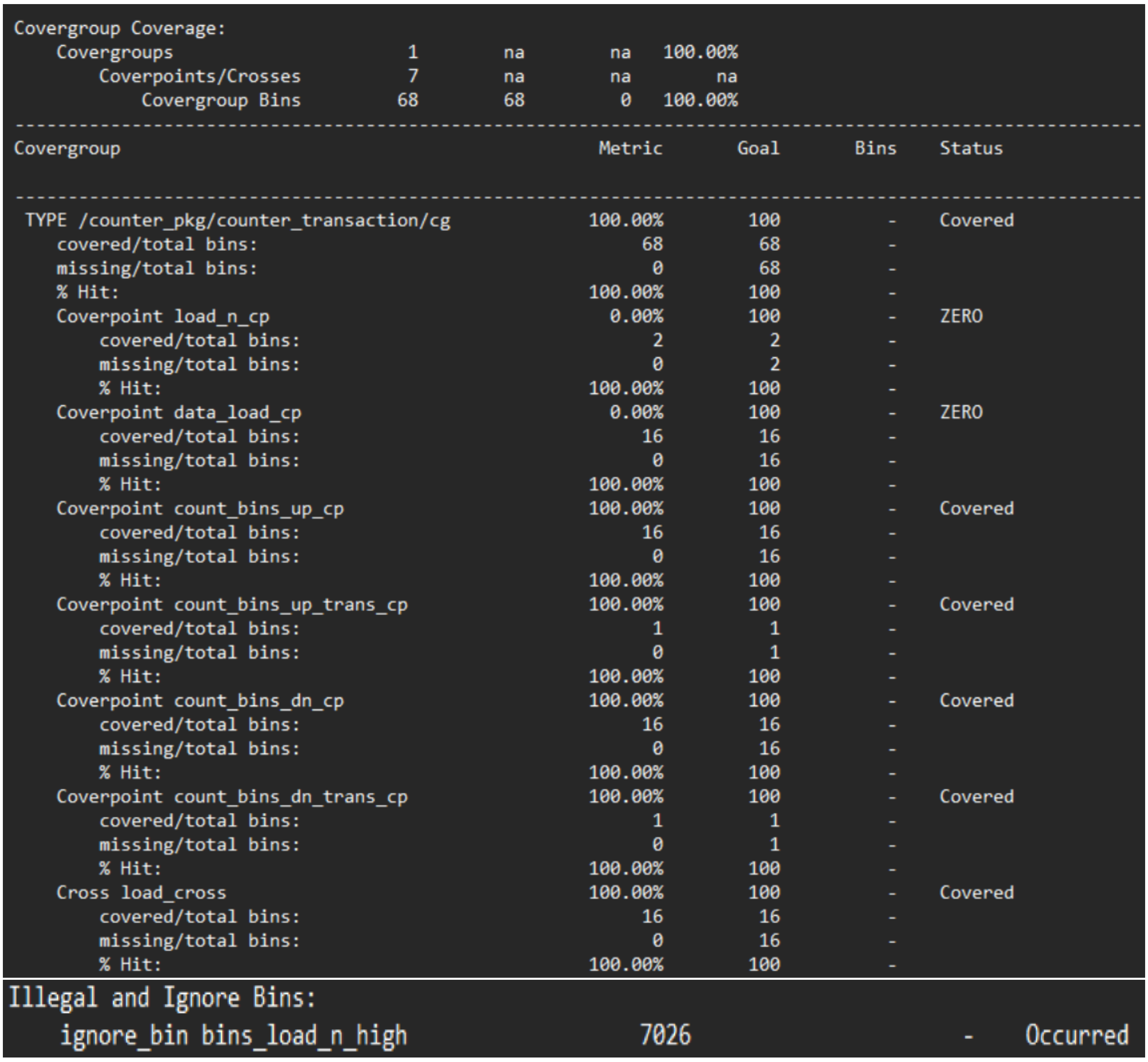
```
==========================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        30         0   100.00%

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

Toggle Coverage for instance /top/c_if --

                                              Node      1H->0L      0L->1H  "Coverage"
                                              ---------------------------------------
                                                ce           1           1      100.00
                                               clk           1           1      100.00
                                    count_out[3-0]           1           1      100.00
                                    data_load[3-0]           1           1      100.00
                                            load_n           1           1      100.00
                                         max_count           1           1      100.00
                                             rst_n           1           1      100.00
                                           up_down           1           1      100.00
                                              zero           1           1      100.00

Total Node Count      =         15
Toggled Node Count    =         15
Untoggled Node Count  =          0

Toggle Coverage       =     100.00% (30 of 30 bins)
```

Assertions Coverage



```
=================================================================================
=== Instance: /\top#dut /T1
=== Design Unit: work.SVA
=================================================================================

Assertion Coverage:
    Assertions                        7          7          0   100.00%
--------------------------------------------------------------------
Name                     File(Line)                   Failure      Pass
                                                      Count        Count
--------------------------------------------------------------------
/\top#dut /T1/data_load_assertion
                     SVA.sv(7)                              0           1
/\top#dut /T1/not_active_assertion
                     SVA.sv(14)                             0           1
/\top#dut /T1/increment_assertion
                     SVA.sv(21)                             0           1
/\top#dut /T1/decrement_assertion
                     SVA.sv(28)                             0           1
/\top#dut /T1/reset_assertion
                     SVA.sv(34)                             0           1
/\top#dut /T1/max_value_assertion
                     SVA.sv(41)                             0           1
/\top#dut /T1/zero_assertion
                     SVA.sv(48)                             0           1
```

FUNCTIONAL COVERAGE



```
Covergroup Coverage:
    Covergroups                      1         na         na    100.00%
        Coverpoints/Crosses          7         na         na         na
            Covergroup Bins         68         68          0    100.00%
----------------------------------------------------------------------------------------------------------
Covergroup                                             Metric       Goal       Bins     Status

----------------------------------------------------------------------------------------------------------
 TYPE /counter_pkg/counter_transaction/cg             100.00%        100          -     Covered
    covered/total bins:                                    68         68          -
    missing/total bins:                                     0         68          -
    % Hit:                                            100.00%        100          -
    Coverpoint load_n_cp                                0.00%        100          -     ZERO
        covered/total bins:                                 2          2          -
        missing/total bins:                                 0          2          -
        % Hit:                                        100.00%        100          -
    Coverpoint data_load_cp                             0.00%        100          -     ZERO
        covered/total bins:                                16         16          -
        missing/total bins:                                 0         16          -
        % Hit:                                        100.00%        100          -
    Coverpoint count_bins_up_cp                       100.00%        100          -     Covered
        covered/total bins:                                16         16          -
        missing/total bins:                                 0         16          -
        % Hit:                                        100.00%        100          -
    Coverpoint count_bins_up_trans_cp                 100.00%        100          -     Covered
        covered/total bins:                                 1          1          -
        missing/total bins:                                 0          1          -
        % Hit:                                        100.00%        100          -
    Coverpoint count_bins_dn_cp                       100.00%        100          -     Covered
        covered/total bins:                                16         16          -
        missing/total bins:                                 0         16          -
        % Hit:                                        100.00%        100          -
    Coverpoint count_bins_dn_trans_cp                 100.00%        100          -     Covered
        covered/total bins:                                 1          1          -
        missing/total bins:                                 0          1          -
        % Hit:                                        100.00%        100          -
    Cross load_cross                                  100.00%        100          -     Covered
        covered/total bins:                                16         16          -
        missing/total bins:                                 0         16          -
        % Hit:                                        100.00%        100          -
Illegal and Ignore Bins:
    ignore_bin bins_load_n_high                    7026                    -     Occurred
```

Achieved 100 %

Functional & Code & Assertions Coverage