ASSIGNMENT -3

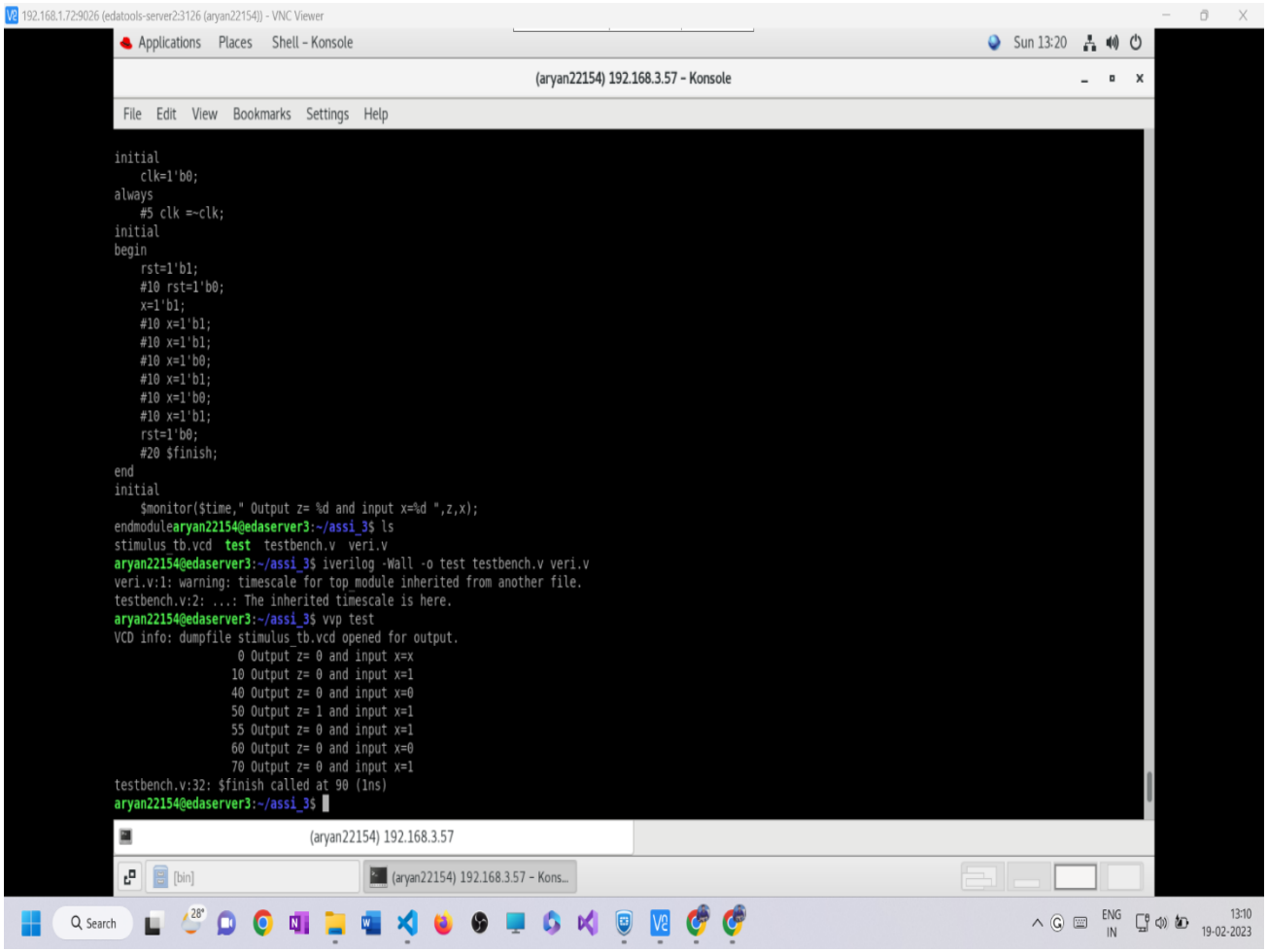
Name: - Aryan Gupta

Roll No- MT22154

VLSI Design Flow (Submitted to Prof. Sneh Saurabh)

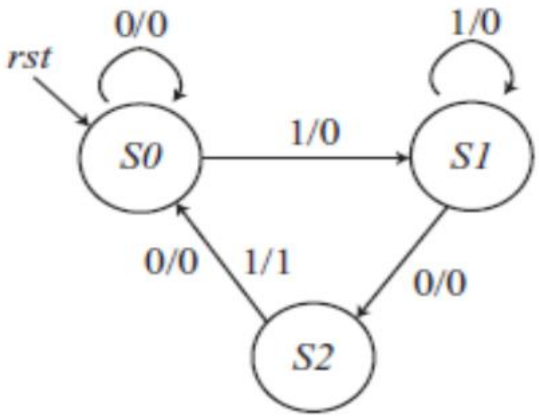
1. Show a screenshot or snippet of the log file to prove that you have indeed run the tool

[If this is not given then your assignment will not be evaluated].



2. Consider the state machine shown below.

(a) Write a Verilog model for the following state machine.



The states are: S0, S1, and S2.

The input values are {0, 1}.

Output values are: {0,1}.

```
module top_module(input clk,input rst,input x,output reg z);
parameter s0 = 3'b000;
parameter s1 = 3'b001;
parameter s2 = 3'b010;
reg [2:0] cur_state,next_state;
always@(posedge clk or posedge rst)
begin
    if(rst)
        cur_state<=s0;
    else
        cur_state<=next_state;
end
always@(*)
begin
    case(cur_state)
```

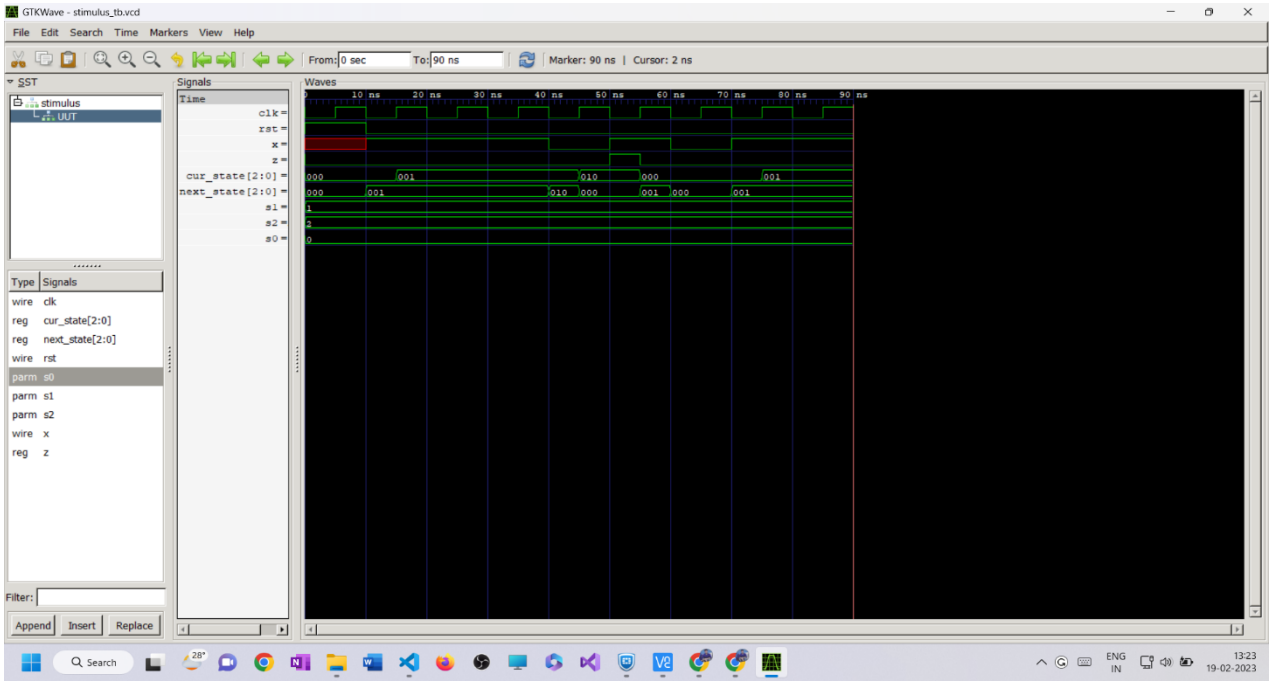
```
    s0 : if(x)
         next_state<=s1;
         else
         next_state<=s0;
    s1 : if(x)
         next_state<=s1;
         else
         next_state<=s2;
    s2 : if(x)
         next_state<=s0;
         else
         next_state<=s0;
    default : next_state<=s0;
    endcase
end
always@(*)
begin
    case(cur_state)
    s0 : z=1'b0;
    s1 : z=1'b0;
    s2 : if(x)
          z=1'b1;
          else
          z=1'b0;
    default : z=1'b0;
    endcase
end
endmodule
```

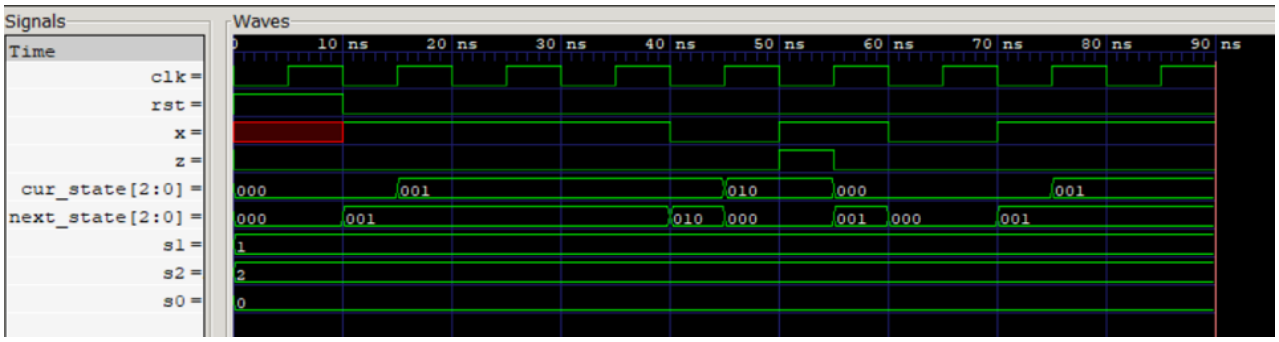
(b) Write the testbench to test the Verilog model.

```
`timescale 1ns/1ns
`include "veri.v"
module stimulus;
reg clk;
reg rst;
reg x;
wire z;
top_module UUT(clk,rst,x,z);
initial
  begin
    $dumpfile("stimulus_tb.vcd");
    $dumpvars;
  end

initial
    clk=1'b0;
always
    #5 clk =~clk;
initial
begin
    rst=1'b1;
    #10 rst=1'b0;
    x=1'b1;
    #10 x=1'b1;
    #10 x=1'b1;
    #10 x=1'b0;
    #10 x=1'b1;
    #10 x=1'b0;
    #10 x=1'b1;
    rst=1'b0;
    #20 $finish;
end
initial
    $monitor($time," Output z= %d and input x=%d ",z,x);
endmodule
```

(c) Show the simulation output waveform and explain the correctness of the model.





So, at the time, t=0 clk starts with the time period of 10ns.

Firstly, the rst is 1 till t=10, then the rst is 0. So, the state can be changed from S0 to other states.

Now from the X=1 till 40ns. With the state diagram, the state change from 000(S0) to 001(S1) in the first posedge of clk, the next_state change to 001, and then curr_state change to 001.

Because for S1(001) to S2(010), the X needs to be 0. So, no state change from 20ns to 40ns.

From t=40ns to t=50ns, X=0, so the state change their according to the state diagram.

001(S1) to 010(S2) in 40ns to 50ns time frame, first next_state change to 010 then curr_state change to 010.

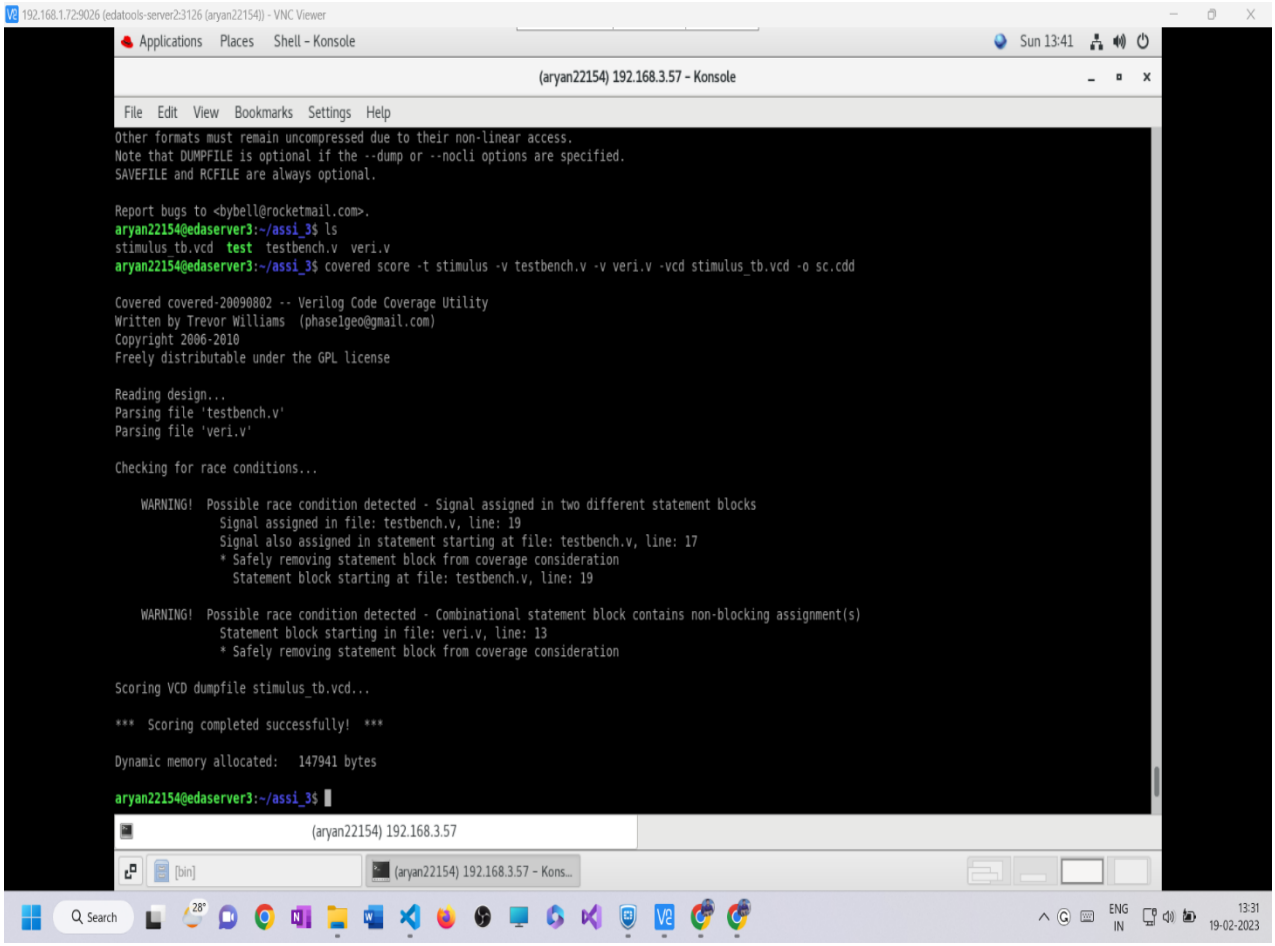
And after that whether X=0 or X=1, the state need to be changed(as per the state machine) so At t=55ns the state change from 010(S2) to 000(S0).

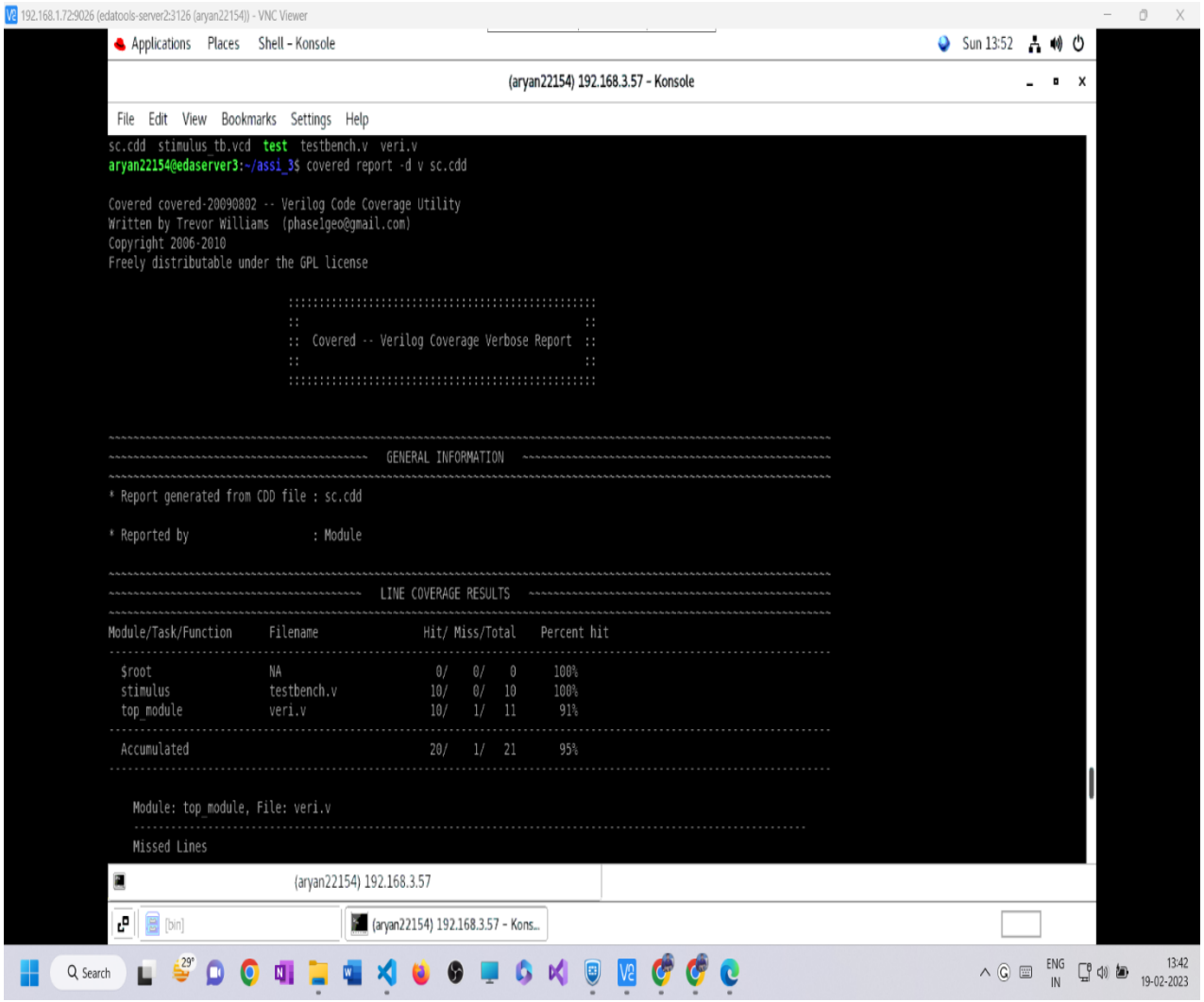
But also, in that time frame the z is also define to change so the Z change from the 0 to 1 in this state.

So, Z=1 from 50ns to 55ns.

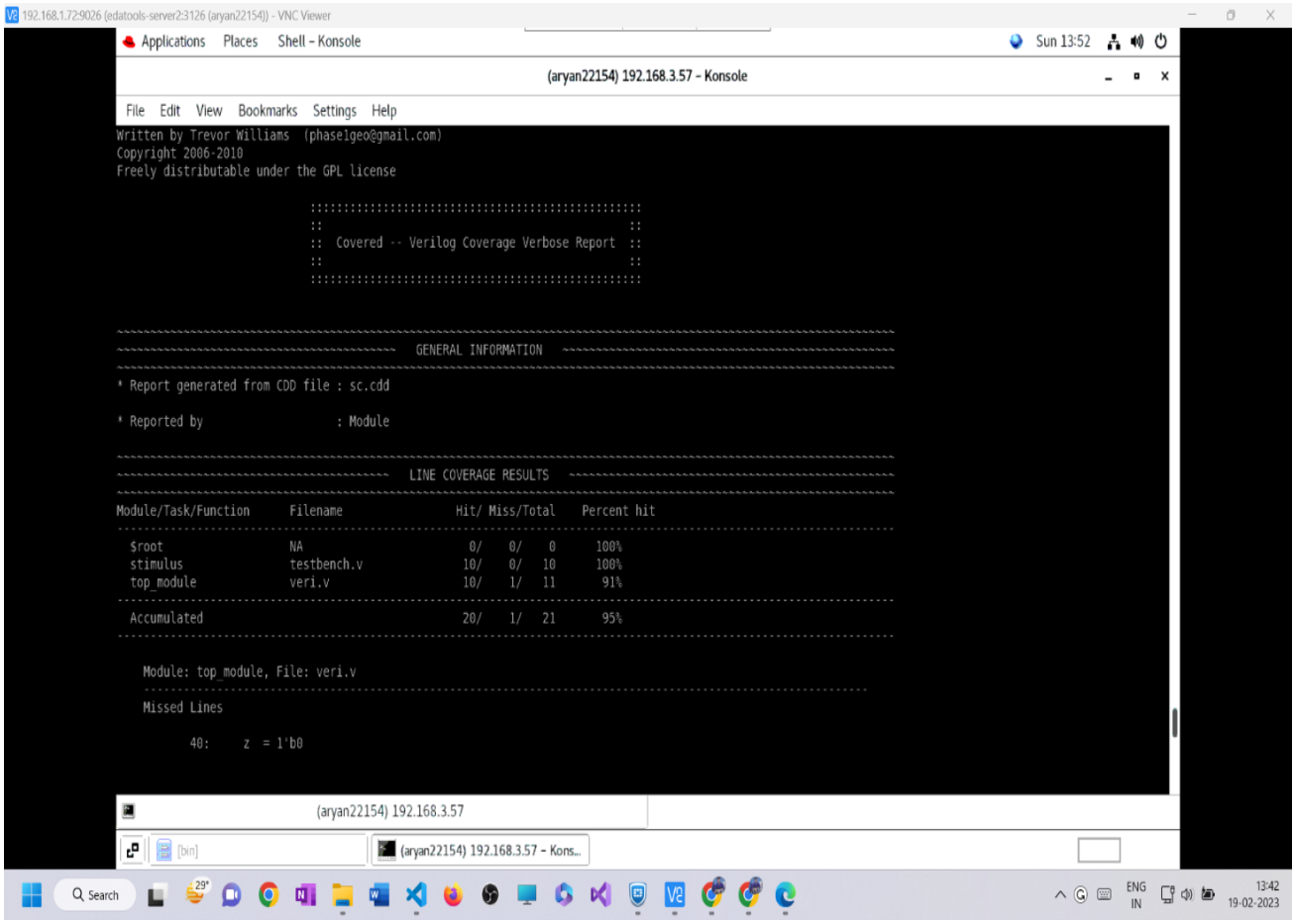
Like that it going to happen. The model is perfect, and it is also shown in the GTKWAVE.

(d) Show line coverage report for the above testbench. Explain the report.




192.168.1.72:9026 (edatools-server2:3126 (aryan22154)) - VNC Viewer
Applications Places Shell – Konsole
Sun 13:52
(aryan22154) 192.168.3.57 – Konsole
File Edit View Bookmarks Settings Help
sc.cdd stimulus_tb.vcd test testbench.v veri.v
aryan22154@edaserver3:~/assi_3$ covered report -d v sc.cdd
Covered covered-20090802 -- Verilog Code Coverage Utility
Written by Trevor Williams (phase1geo@gmail.com)
Copyright 2006-2010
Freely distributable under the GPL license
:: Covered -- Verilog Coverage Verbose Report ::
GENERAL INFORMATION
* Report generated from CDD file : sc.cdd
* Reported by : Module
LINE COVERAGE RESULTS
Module/Task/Function Filename Hit/ Miss/Total Percent hit
$root NA 0/ 0/ 0 100%
stimulus testbench.v 10/ 0/ 10 100%
top_module veri.v 10/ 1/ 11 91%
Accumulated 20/ 1/ 21 95%
Module: top_module, File: veri.v
Missed Lines
(aryan22154) 192.168.3.57
[bin]
(aryan22154) 192.168.3.57 - Kons...
Search
ENG IN
13:42 19-02-2023

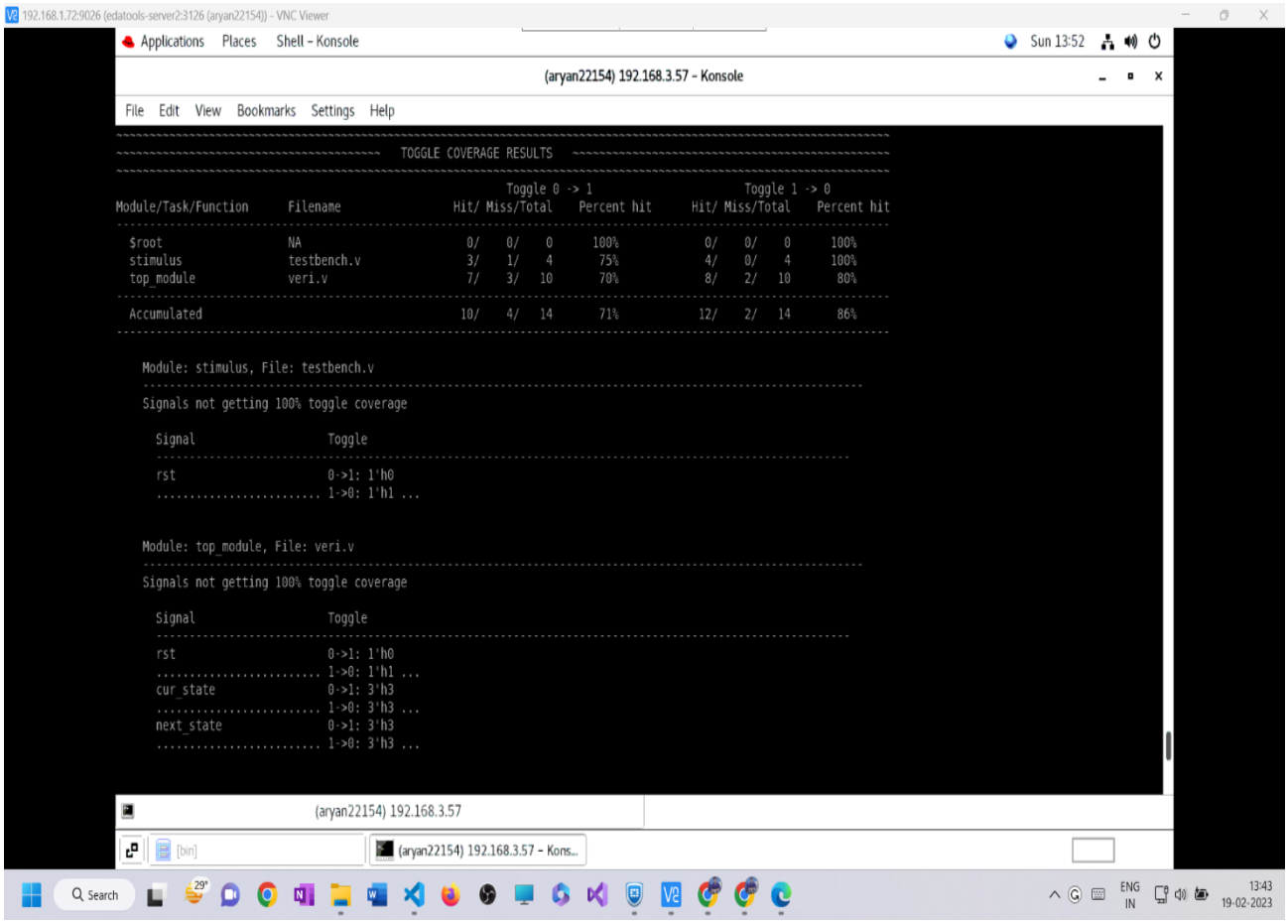
Line Coverage Result



Line coverage refers to how many lines in my code were executed. The lines that are not executed are the else and the default line, so, in my case, there is one line at 40: z=1’b0;(which is the default case)

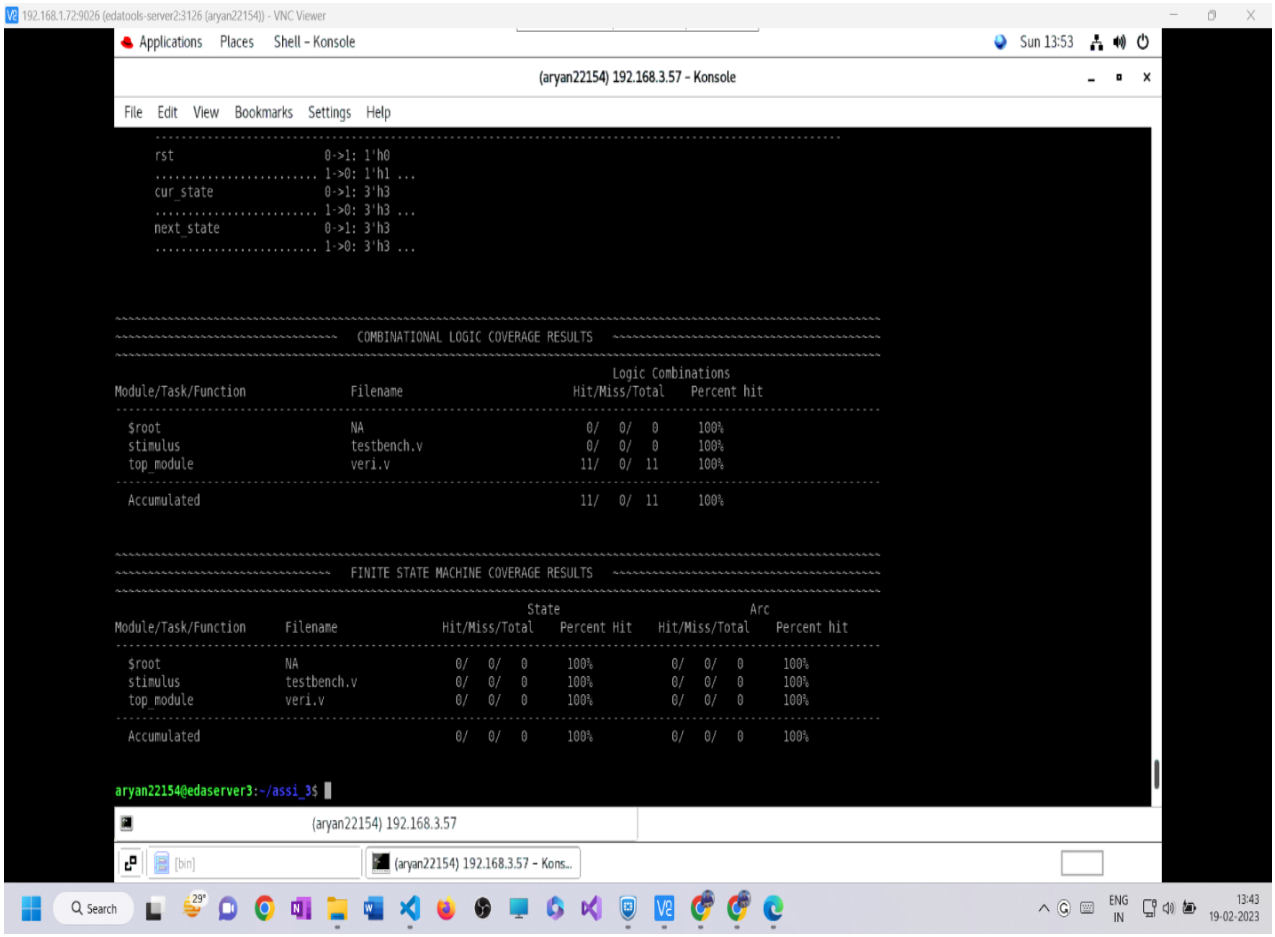
Overall, the Line Coverage result is 95%.

Toggle Coverage Result



Toggle coverage measures the unique values our expressions evaluated. So, basically, we don’t need the toggle the rst. Due to this, my toggle coverage gets decreased but is highly good, which, is Overall around 85-90 %.

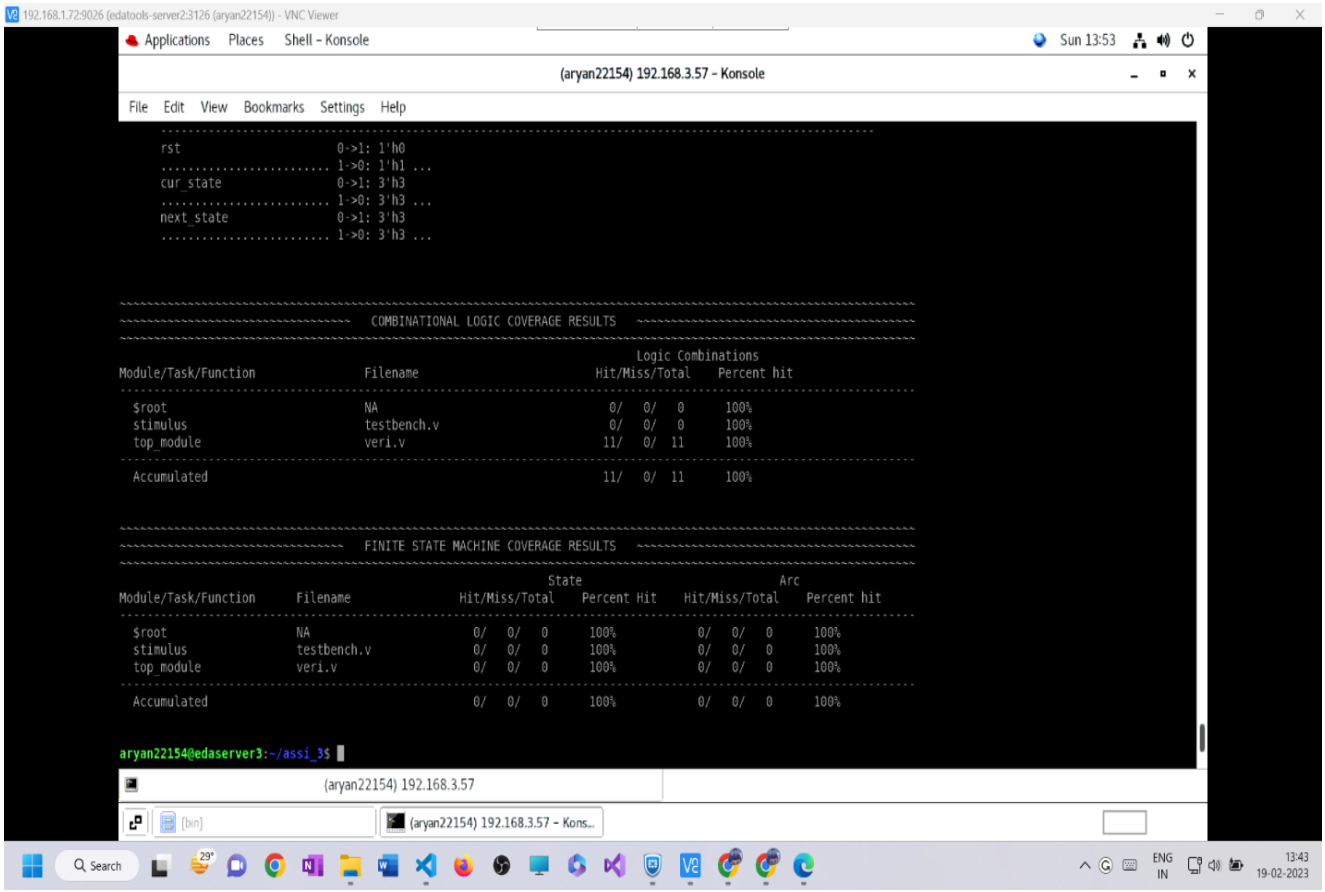
Combinational Logic Coverage Result



It measures the unique values our expressions evaluated. So in the test bench, all the unique values are evaluated whether x=1,0 or state 001,000,010, etc, or output z=0,1.

So, I got 100% Combinational Logic Coverage.

Finite State Machine Coverage Result



This describes the robustness of my test bench. It answers if I have reached all possible states and taken all possible paths.

Since I my code all the states are reached so my Finite state machine coverage result is 100%.