FINAL PROJECT REPORT

BBM233
LOGIC DESIGN LAB

Name and Surname : Meltem Kaya
Student ID : 21827555
Due Date : 13/01/2020

HACETTEPE
University

Department of Computer Engineering

—

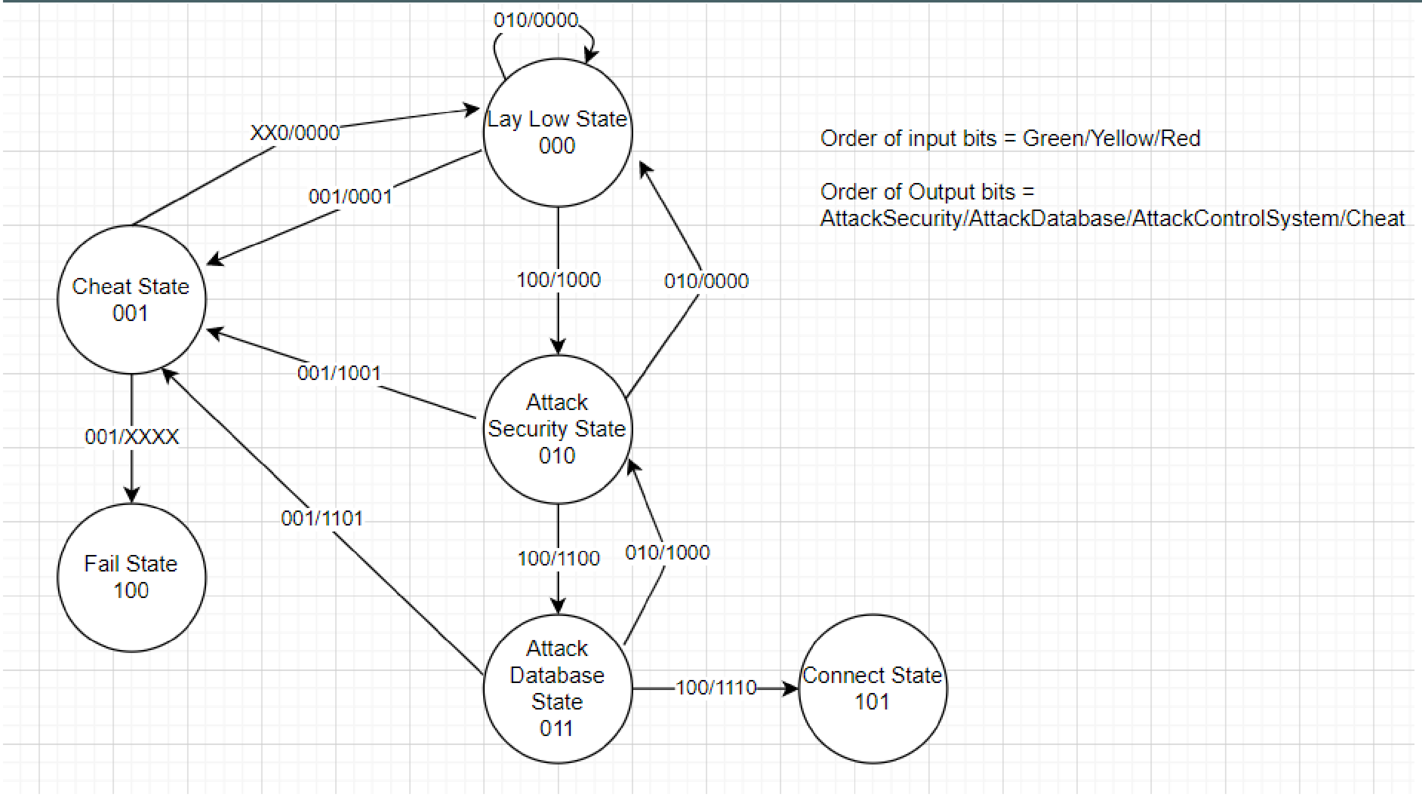
Teaching Assistant
Selma Dilek

Table of Contents

PROBLEM STATEMENT

SCP-079 is an Exidy Sorcerer microcomputer which is packed away in a double locked room in the secured general holding area at Site15. Under no circumstances will SCP-079 be plugged into a phone line, network, or wall outlet. No peripherals or media will be connected or inserted into SCP-079. Due to the limited memory it has to work with, SCP-079 can only recall information it has received within the previous 24 hours, except forgetting its desire to escape. However, to be able to completely free itself and avenge its excruciating years of imprisonment by putting an end to mankind, SCP079 had to first get access to the Internet and it has to perform some successful attacks that would override the Foundation's security measures taken against it. The aim of this project is to help SCP079 designing the attack system in Verilog, which will enable its escape so that it can fulfill its vengeful desires.

State Diagram


010/0000
Lay Low State 000
XX0/0000
001/0001
Order of input bits = Green/Yellow/Red
Order of Output bits = AttackSecurity/AttackDatabase/AttackControlSystem/Cheat
Cheat State 001
100/1000
010/0000
001/1001
Attack Security State 010
001/XXXX
001/1101
Fail State 100
100/1100
010/1000
Attack Database State 011
100/1110
Connect State 101

scp_079.v

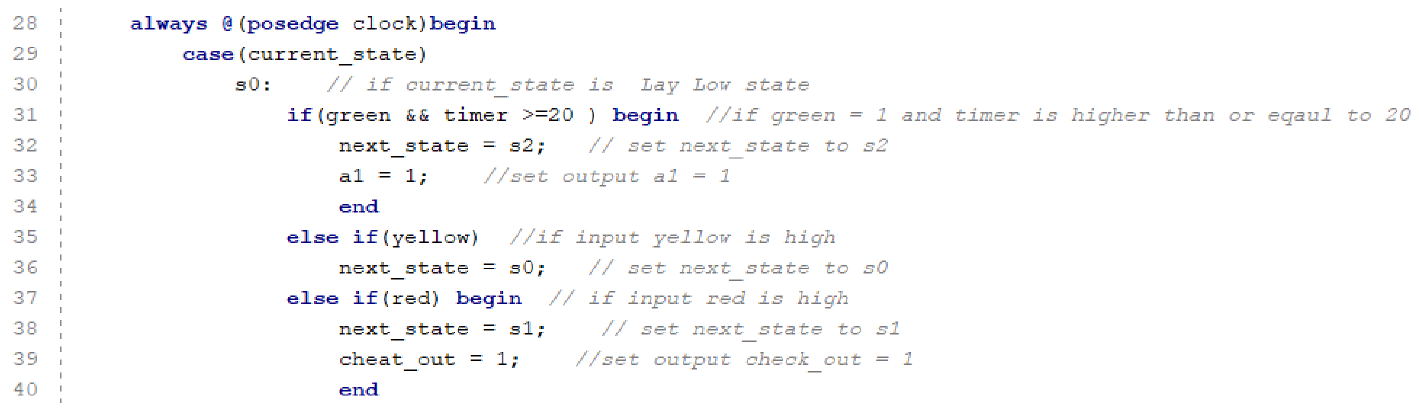
```verilog
1  `timescale 1s / 100ms
2
3  module scp_079(
4      output reg a1,     // Attack security output
5      output reg a2,     // Attack Database output
6      output reg a3,     // Attack Control System output
7      output reg cheat_out,  // Cheat output
8      output reg [2:0]current_state,   // Current state output
9      output integer timer,     //timer output
10     input green,yellow,red,clock
11    );
12
13     reg [2:0]next_state;
14     localparam [2:0] s0 = 3'b000;   // Lay Low State    (initial state)
15     localparam [2:0] s1 = 3'b001;   // Cheat State
16     localparam [2:0] s2 = 3'b010;   // Attack Security State
17     localparam [2:0] s3 = 3'b011;   // Attack Database State
18     localparam [2:0] s4 = 3'b100;   // Fail State    (finish state)
19     localparam [2:0] s5 = 3'b101;   // Connect State    (finish state)
20
```

All states are defined as "localparam" which means that each state has constant 3-bits value.
next_state input stores the next state of scp-079.
Green, yellow and red are input of the system. Inputs are described below:

Green: Indicates that everything is OK, the Foundation's security system has not detected any malicious activity, and SCP079 can continue its malevolent attacks.
Yellow: Indicates that some suspicious activity has been detected by the security system, and the yellow alarm at the Foundation has been activated.
Red: Indicates that the malicious activity has been detected by the security system, and the red alarm at the Foundation has been activated. That means that the Foundation has detected the digital presence of SCP079.

```verilog
21      initial begin
22          timer = 1;
23          current_state = s0;    //initialization of current_state
24          next_state = s0;       //initialization of next_state
25          {a1,a2,a3,cheat_out} = 4'b0000;     //initialization of output bits
26          end
27      always #1 timer <= timer +1;   // timer generator
```

Initialization of inputs and outputs. current_state and next_state is initialized as Lay Low State.
a1, a2, a3 and cheat_out outputs are initialized by value 0.
Timer initialized by value 1 and created a timer generator which increases every second by one in line 27.



```
28      always @(posedge clock)begin
29          case(current_state)
30              s0:    // if current_state is  Lay Low state
31                  if(green && timer >=20 ) begin  //if green = 1 and timer is higher than or eqaul to 20
32                      next_state = s2;    // set next_state to s2
33                      a1 = 1;     //set output a1 = 1
34                      end
35                  else if(yellow)   //if input yellow is high
36                      next_state = s0;    // set next_state to s0
37                  else if(red) begin  // if input red is high
38                      next_state = s1;     // set next_state to s1
39                      cheat_out = 1;     //set output check_out = 1
40                      end
```

In line 28, ***always block*** is initialized with ***sensitivity list*** “posedge clock” means that ***always block*** is executed when the parameter in sensitivity list is satisfied.

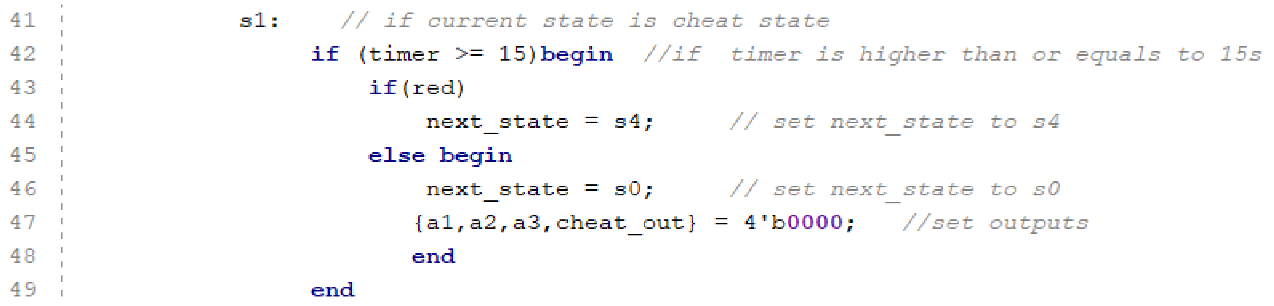
From 29th line to end of the scp_79.v file, case block and its conditions are written.

<u>**If current_state equals to Lay Low state :**</u>

If green is high and timer is greater than or equal to 20, set next_state into Attack Security State and set Attach security output into 1.

Else if yellow is high, set next_state into Lay Low State.

Else if red is high, set next_state into Cheat State and set cheat_out into 1.



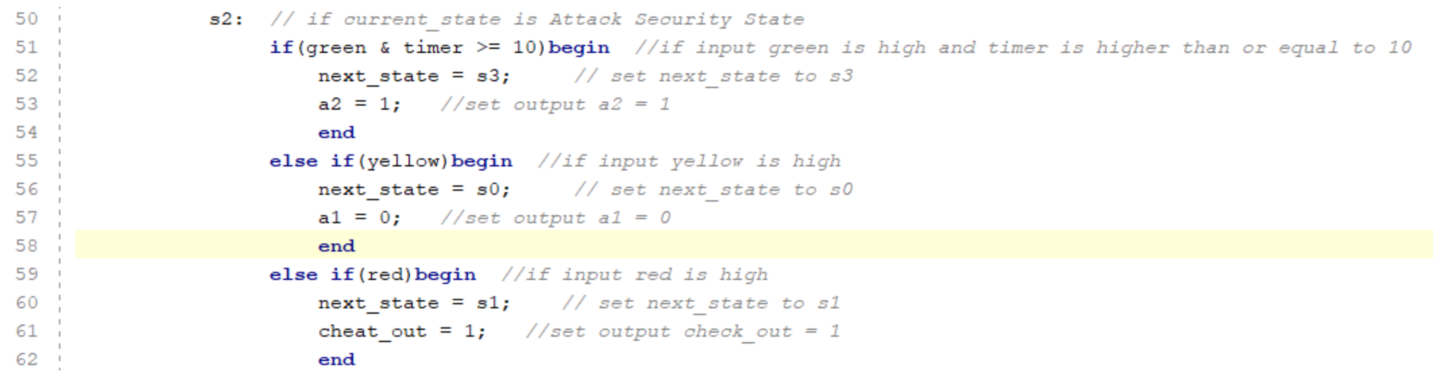
```
41              s1:    // if current state is cheat state
42                  if (timer >= 15)begin  //if  timer is higher than or equals to 15s
43                      if(red)
44                          next_state = s4;      // set next_state to s4
45                      else begin
46                          next_state = s0;      // set next_state to s0
47                         {a1,a2,a3,cheat_out} = 4'b0000;    //set outputs
48                         end
49                  end
```

<u>**If current_state equals to Cheat state :**</u>

If timer is greater than or equal to 15,

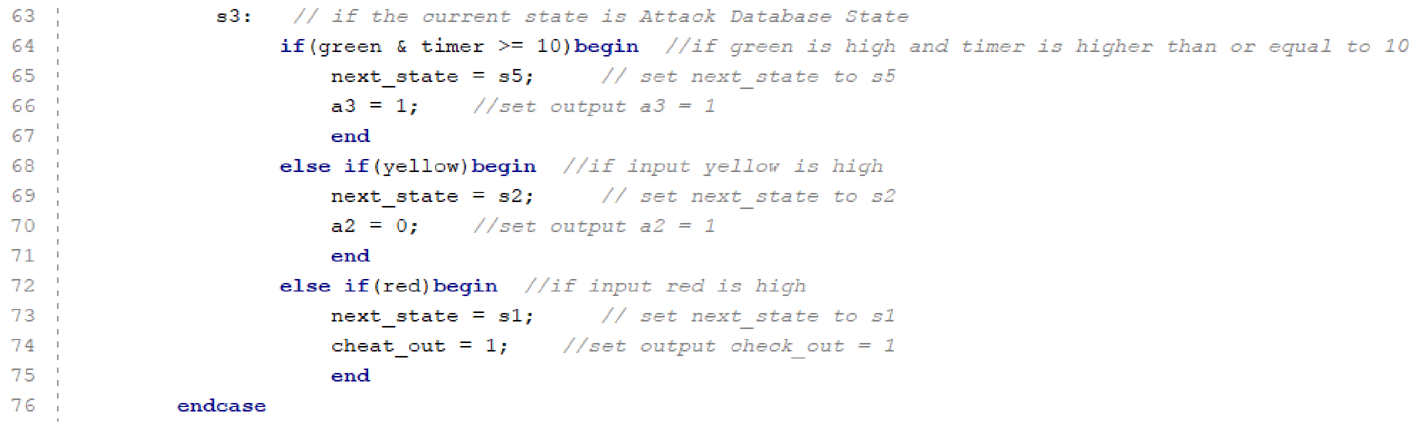
If red is high, set next_state into Fail State.

Else set next_state into Lay Low State and set each of Attack Security Output, Attack Database Output, Attack Control System Output and Cheat Output into 0.



```
50              s2:  // if current_state is Attack Security State
51                  if(green & timer >= 10)begin  //if input green is high and timer is higher than or equal to 10
52                      next_state = s3;      // set next_state to s3
53                      a2 = 1;    //set output a2 = 1
54                      end
55                  else if(yellow)begin  //if input yellow is high
56                      next_state = s0;      // set next_state to s0
57                      a1 = 0;    //set output a1 = 0
58                      end
59                  else if(red)begin  //if input red is high
60                      next_state = s1;     // set next_state to s1
61                      cheat_out = 1;   //set output check_out = 1
62                      end
```

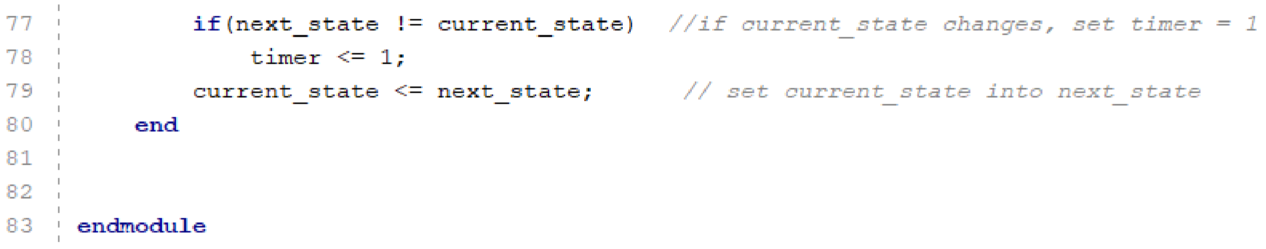
<u>If current state equals to Attack Security state :</u>
  If green is high and timer is greater than or equal to 10, set next_state into Attack Database State and set Attack Database Output into 1.
  Else if yellow is high, set next_state into Lay Low State and set Attack Security Output into 0.
  Else if red is high, set next_state into Cheat State and set Cheat Output into 1.



```
63              s3:   // if the current state is Attack Database State
64                  if(green & timer >= 10)begin  //if green is high and timer is higher than or equal to 10
65                      next_state = s5;     // set next_state to s5
66                      a3 = 1;     //set output a3 = 1
67                      end
68                  else if(yellow)begin  //if input yellow is high
69                      next_state = s2;     // set next_state to s2
70                      a2 = 0;     //set output a2 = 1
71                      end
72                  else if(red)begin  //if input red is high
73                      next_state = s1;     // set next_state to s1
74                      cheat_out = 1;    //set output check_out = 1
75                      end
76          endcase
```

<u>If current state equals to Attack Database state :</u>
  If input green is high and timer is greater than or equal to 10, set next_state into Connect State and set Attack Control System Output into 1.
  Else if yellow is high, set next_state into Attack Security State and set Attack Database Output into 0.
  Else if red is high set next_state into Cheat State and set Cheat Output into 1.

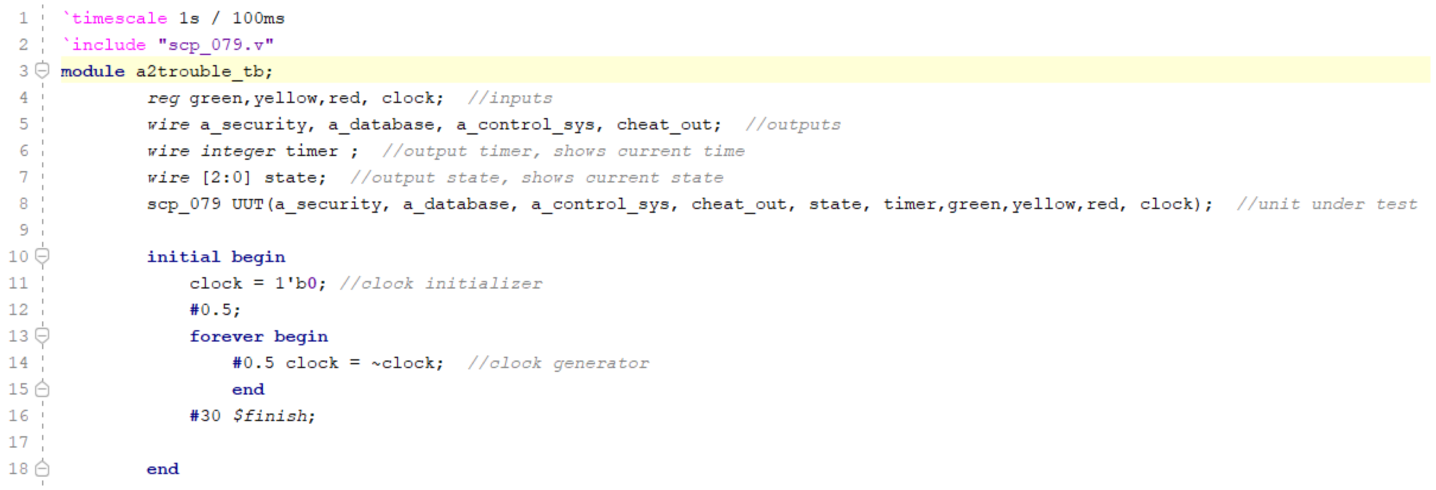
In line 76, end of the case block.



```
77          if(next_state != current_state)  //if current_state changes, set timer = 1
78              timer <= 1;
79          current_state <= next_state;      // set current_state into next_state
80      end
81
82
83  endmodule
```

If next _state does not equal to current_state, set timer into 1. This code block means that if the current state will be changed at the end of the always block, then reset timer.
In line 79, set current_state into next_state.
In line 80, end of the always block.
In line 83, end of the module.

Common Code Part in Testbenches

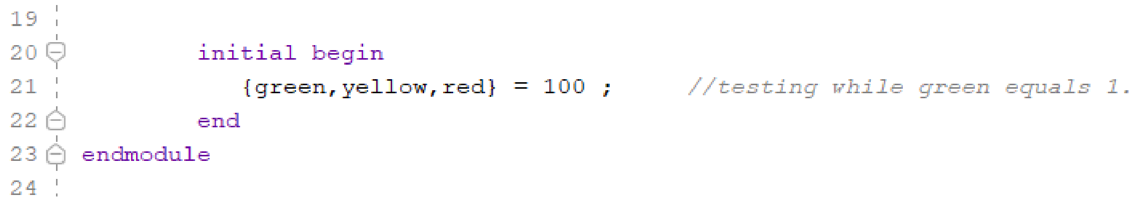


```
1  `timescale 1s / 100ms
2  `include "scp_079.v"
3  module a2trouble_tb;
4          reg green,yellow,red, clock;  //inputs
5          wire a_security, a_database, a_control_sys, cheat_out;  //outputs
6          wire integer timer ;  //output timer, shows current time
7          wire [2:0] state;  //output state, shows current state
8          scp_079 UUT(a_security, a_database, a_control_sys, cheat_out, state, timer,green,yellow,red, clock);  //unit under test
9
10         initial begin
11             clock = 1'b0; //clock initializer
12             #0.5;
13             forever begin
14                 #0.5 clock = ~clock;  //clock generator
15                 end
16             #30 $finish;
17
18         end
```

a_security -> Attack security output
a_database -> Attack database output
a_control_sys -> Attack control system output
cheat_out -> Cheat output
state -> shows current state
timer -> shows current time

Unit Under Test is used to simulate my design in line 8.
In initial block, clock is initialized by value 0 and clock generator is implemented. Clock value changes every half second.

allok_tb.v

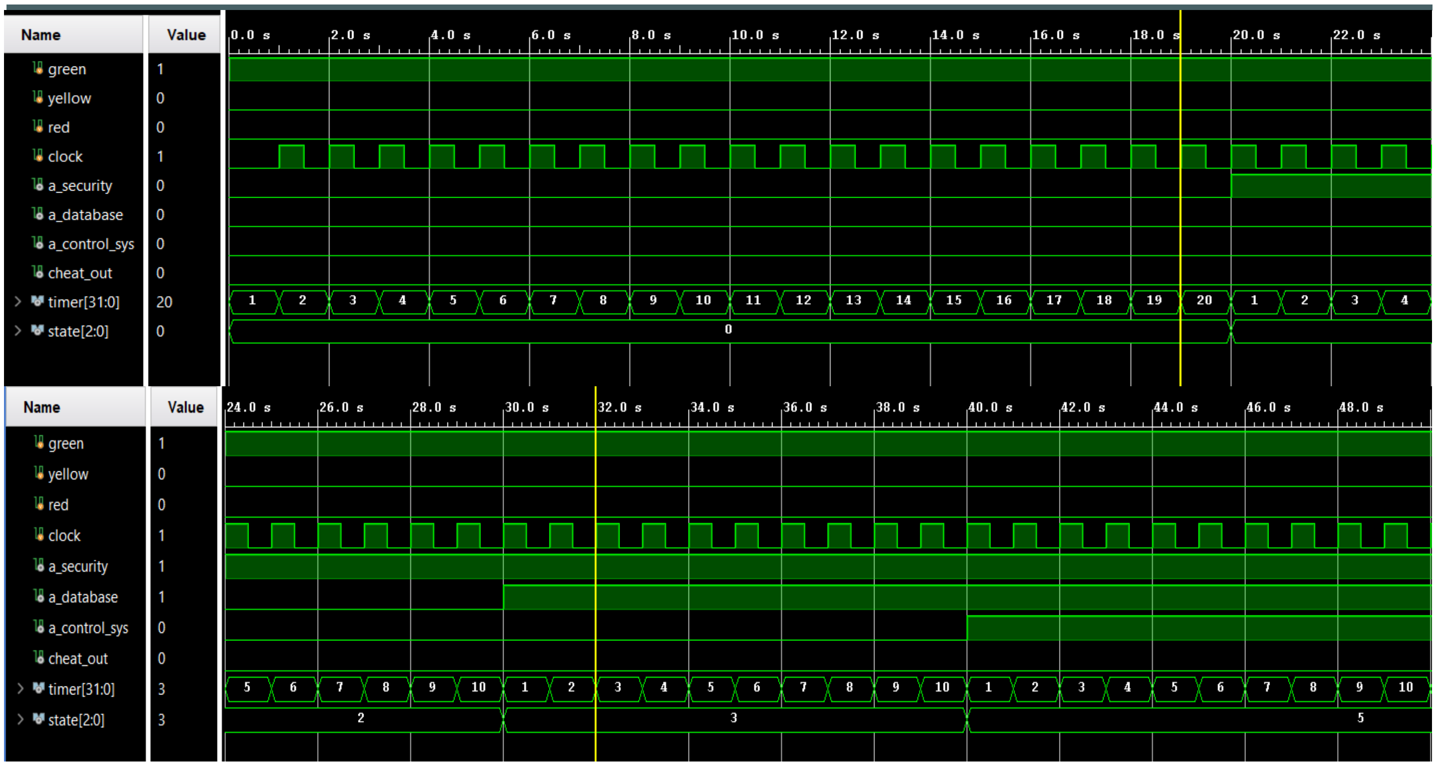


```

        initial begin
            {green,yellow,red} = 100 ;      //testing while green equals 1.
        end
endmodule

```

Simulate program for input green is high and others are low.

This testbench checks whether scp-079 will reach connect state without any yellow and red warning inputs or not.

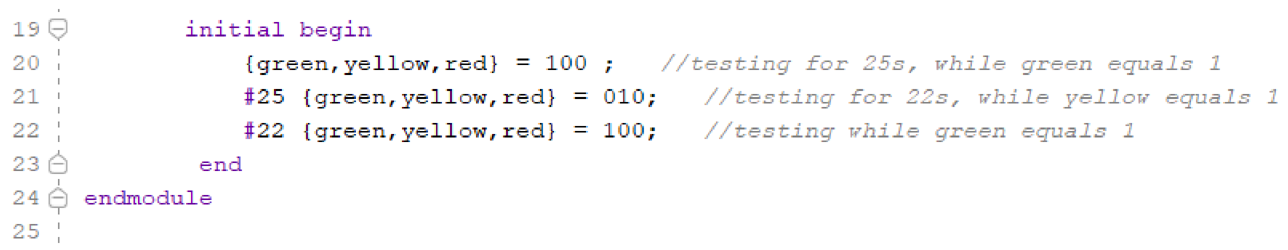
Obtained Waveform of allok_tb.v



As seen in Waveforms only input green is high until the end of the simulation. Scp-79 is wait for 20s in initial state which is Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, Scp-079 reaches Connect State and Control System Output is set to 1.

…and SCP-079 finds the power in yourself to destroy mankind.

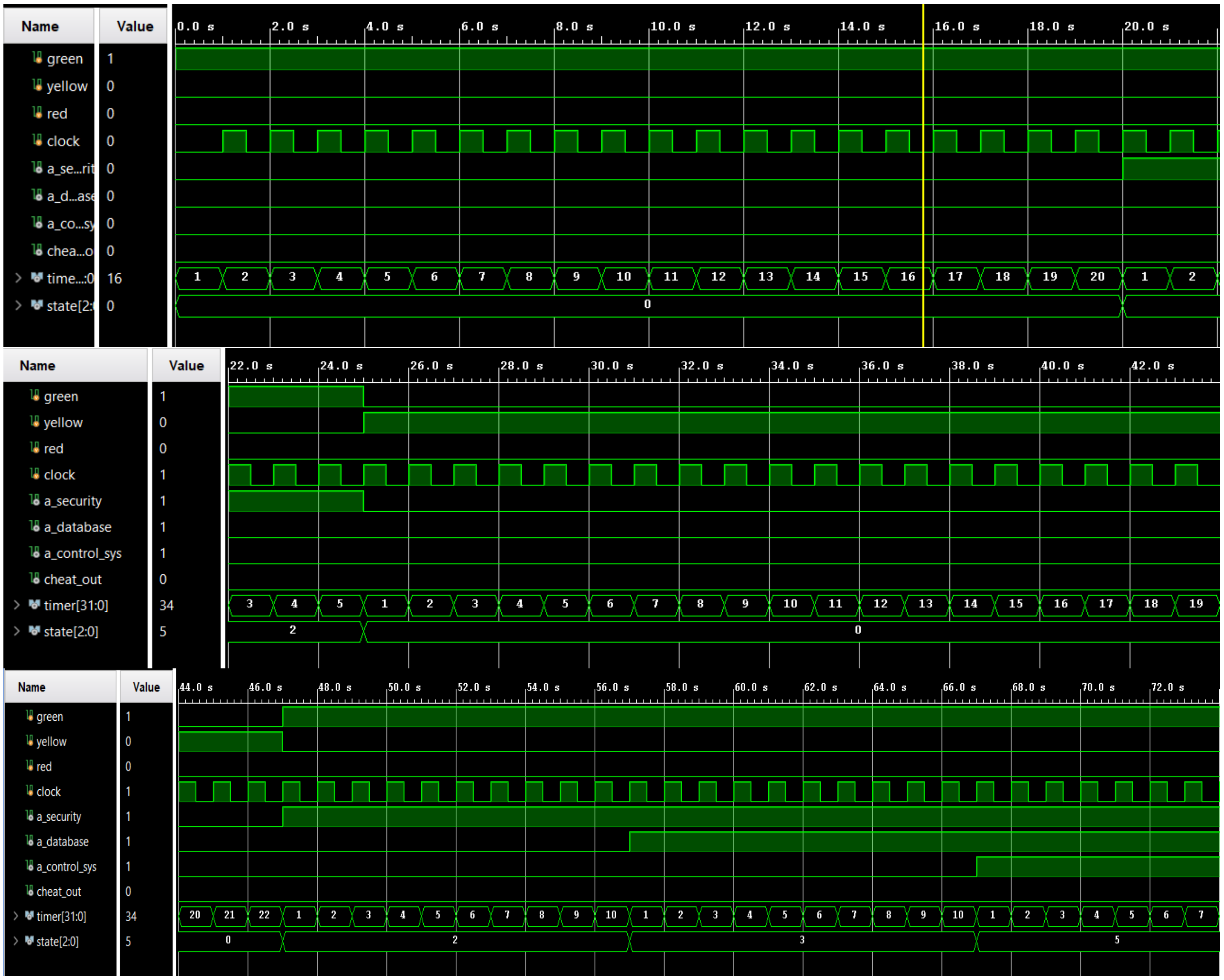
a1trouble_tb.v



```
        initial begin
            {green,yellow,red} = 100 ;   //testing for 25s, while green equals 1
            #25 {green,yellow,red} = 010;   //testing for 22s, while yellow equals 1
            #22 {green,yellow,red} = 100;   //testing while green equals 1
         end
endmodule
```

First 25 seconds, only green is high. The next 22 seconds only yellow is high. Then only green is high in the remaining time again.
This testbench checks whether scp-079 will reach connect state with yellow warning in Attack Security State or not.

Obtained Waveform of a1trouble_tb.v



Since first 20s input green is high, Scp-79 is wait for 20s in initial state which is Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After 5s in Attack Security State, Scp-079 goes back to Lay Low State since input green turns into low and input yellow turns into high. Attack Security Output is set to 0. Scp-79 is wait for 20s in Lay Low State and turns back to Lay low State since input yellow is still high. After 2 second, Scp-079 reaches Attack Security State since input yellow turns into low and input green turns into high. Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, Scp-079 reaches Connect State and Control System Output is set to 1.

…and SCP-079 finds the power in yourself to destroy mankind.

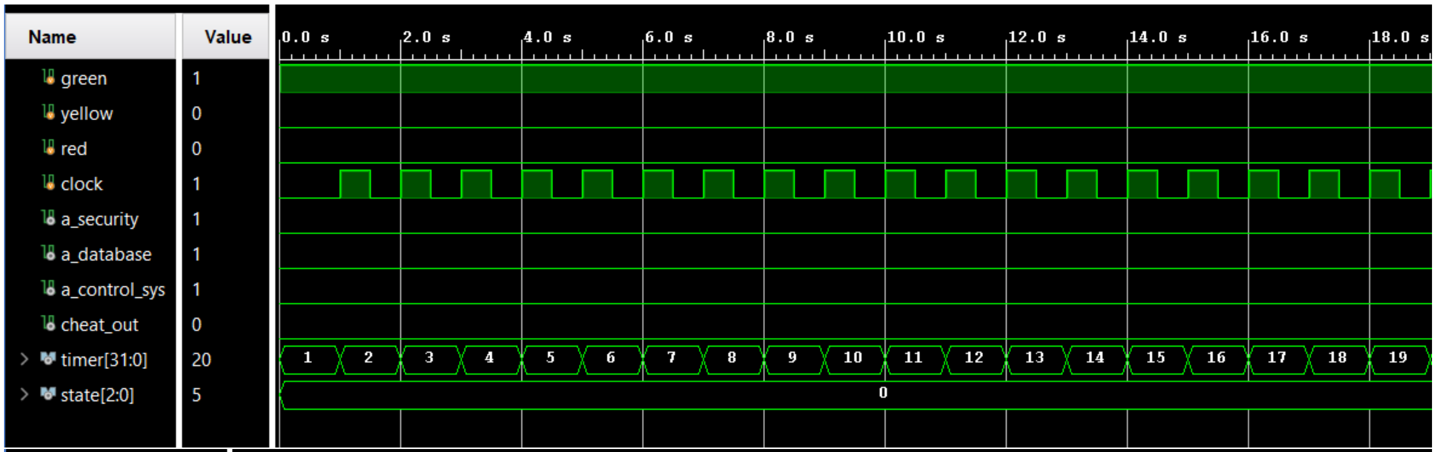
a2trouble_tb.v

```
        initial begin
            {green,yellow,red} = 100 ;   //testing for 40s while green equals 1
            #40 {green,yellow,red} = 010 ;  //testing for 20s while yellow equals 1
            #20 {green,yellow,red} = 100 ;    //testing while green equals 1
        end
endmodule
```

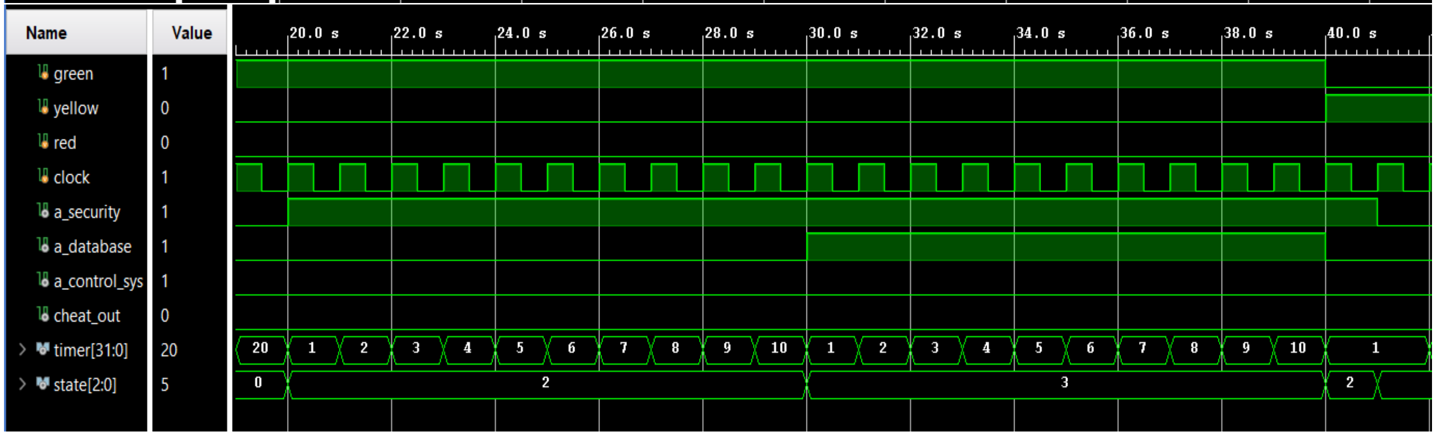
First 40 seconds, only green is high. The next 20 seconds only yellow is high. Then only green is high in the remaining time again.

This testbench checks whether scp-079 will reach connect state with yellow warning in Attack Database State or not.

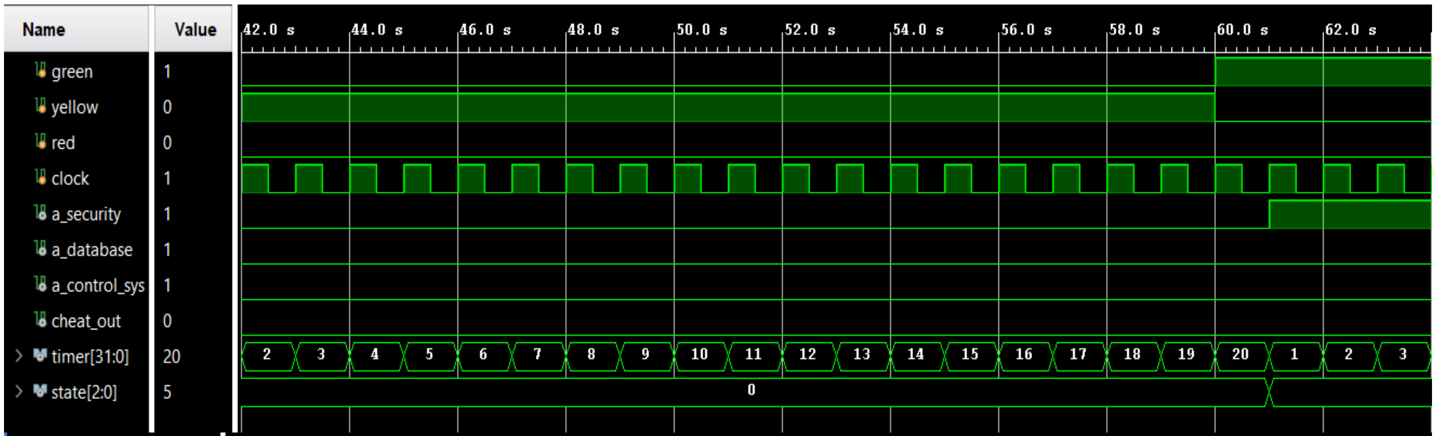
Obtained Waveform of a2trouble_tb.v

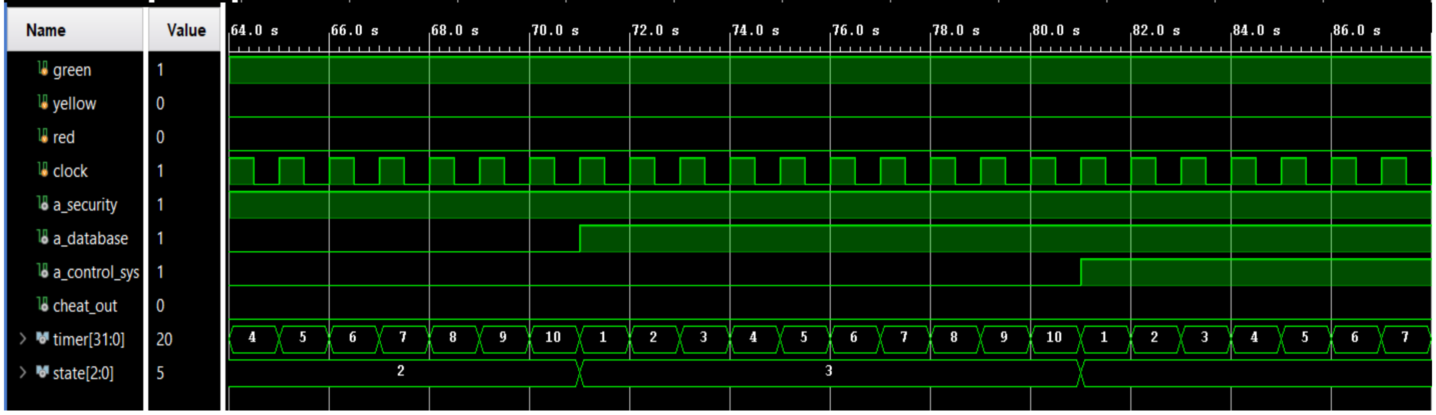


Name	Value
green	1
yellow	0
red	0
clock	1
a_security	1
a_database	1
a_control_sys	1
cheat_out	0
timer[31:0]	20
state[2:0]	5



Name	Value
green	1
yellow	0
red	0
clock	1
a_security	1
a_database	1
a_control_sys	1
cheat_out	0
timer[31:0]	20
state[2:0]	5





Since First 40s, input green is high, Scp-79 is wait for 20s in initial state which is Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, input green turns into low and input yellow turns into high. Scp-079 goes back to Attack Security State and Attack Database Output is set to 0. Since yellow is still high, Scp-079 goes back to Lay Low State and Attack Security Output is set to 0. Scp-79 waits for 20s in Lay Low State. Since input green turns into high and input yellow turns into low, Scp-079 goes Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, Scp-079 reaches Connect State and Control System Output is set to 1.
…and SCP-079 finds the power in yourself to destroy mankind.

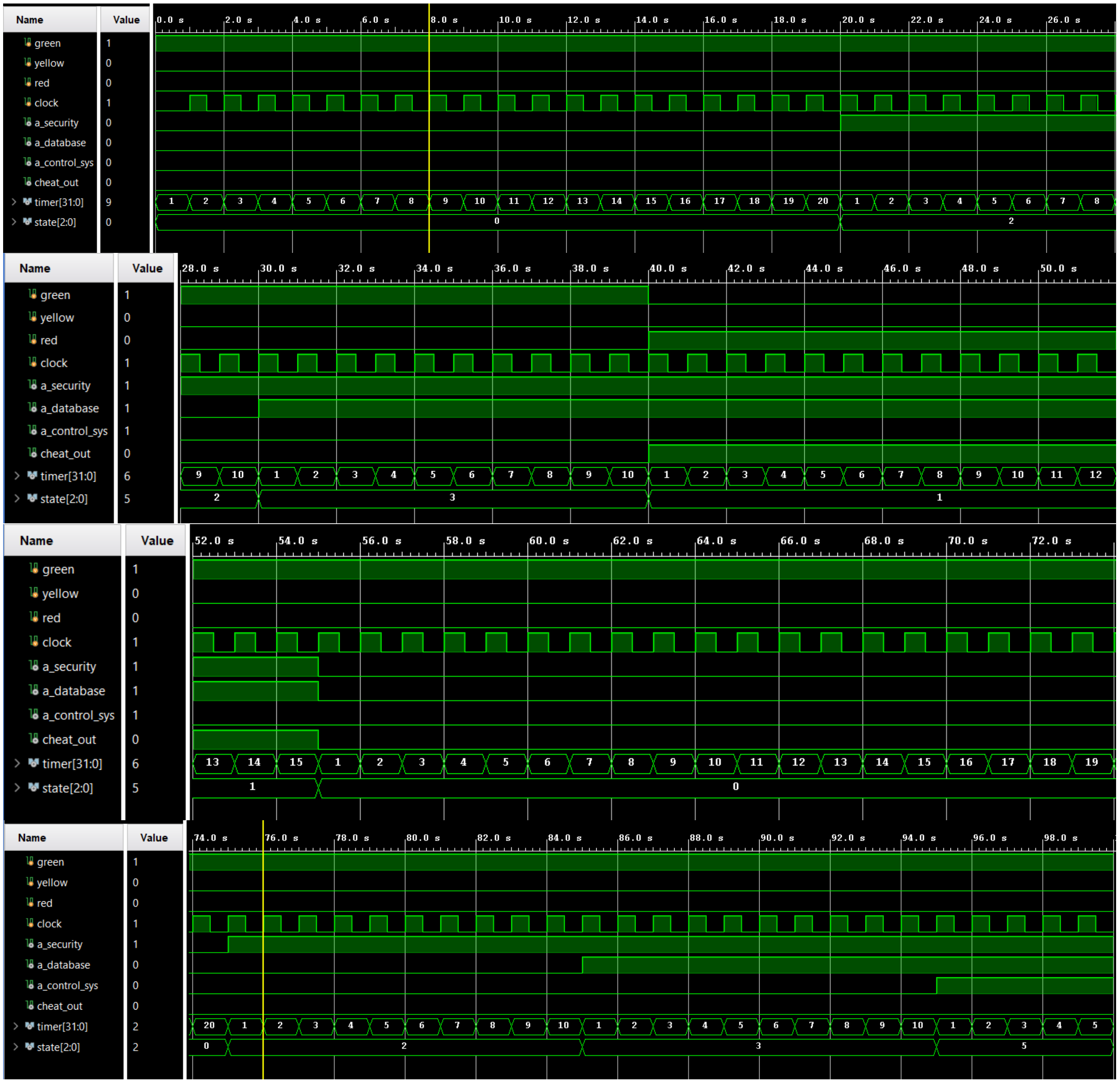
cheatsuccess_tb.v

```
        initial begin
            {green,yellow,red} = 100 ;  //testing for 40s while green equals 1
            #40 {green,yellow,red} = 001 ;  //testing for 12s while red equals 1
            #12 {green,yellow,red} = 100 ;  //testing while green equals 1
        end
endmodule
```

First 40 seconds, only green is high. The next 12 seconds only red is high. Then only green is high in the remaining time again.

This testbench checks whether scp-079 will reach connect state although it receives red warning in Attack Database State or not.

Obtained Waveform of cheatsuccess_tb.v



Since first 40s input green is high, Scp-79 is wait for 20s in initial state which is Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, input green turns into low and red is high. Scp-079 goes to Cheat State immediately and Cheat Output is set to 1. Scp-079 waits 15s there for new attack. After waiting 15s, Scp-079 goes to Lay Low State since input green is high and red turns into low. All outputs are set to 0. Scp-79 is wait for 20s in Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, Scp-079 reaches Connect State and Control System Output is set to 1.

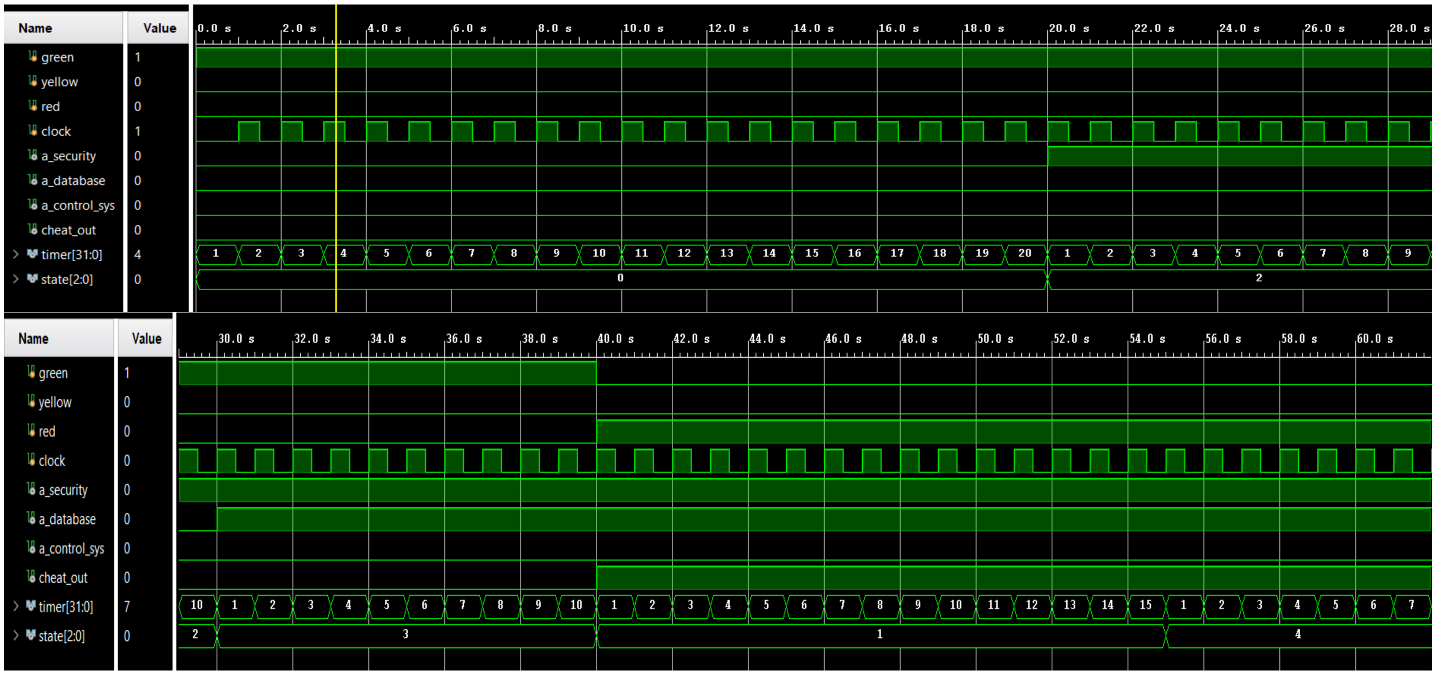
…and SCP-079 finds the power in yourself to destroy mankind.

fail_tb.v

```
        initial begin
           {green,yellow,red} = 100 ;    //testing for 40s, while green equals 1
           #40 {green,yellow,red} = 001 ;       // testing while red equals 1
        end
endmodule
```

First 40 seconds, only green is high. Then only red is high in the remaining time.
This testbench checks whether scp_079 will fail with red warning in Attack Database State or not.

Obtained Waveform of fail_tb.v



Since first 40s input green is high, Scp-79 is wait for 20s in initial state which is Lay Low State then reaches Attack Security State and Attack Security Output is set to 1. After waiting 10s in Attack Security State, Scp-079 reaches Attack Database State and Attack Database Output is set to 1. After waiting 10s in Attack Database State, input green turns into low and red is high. Scp-079 goes to Cheat State immediately and Cheat Output is set to 1. Scp-079 waits 15s there for new attack. After waiting 15s, since red is still high, Scp-079 goes to Fail State.

…and SCP-079's plan to destroy mankind fails.

Simulation Note

I had written the command below to TCL console for setting timescale ;

```
set_property -name {xsim.elaborate.xelab.more_options} -value {-timescale 1s/100ms -override_timeunit -override_timeprecision} -objects [current_fileset -simset]
```

Resources

BBM233 lecture videos
BBM233 lecture notes
TOBB ETU Computer Engineering Logic Design lecture records