Hacettepe University

Computer Engineering Department

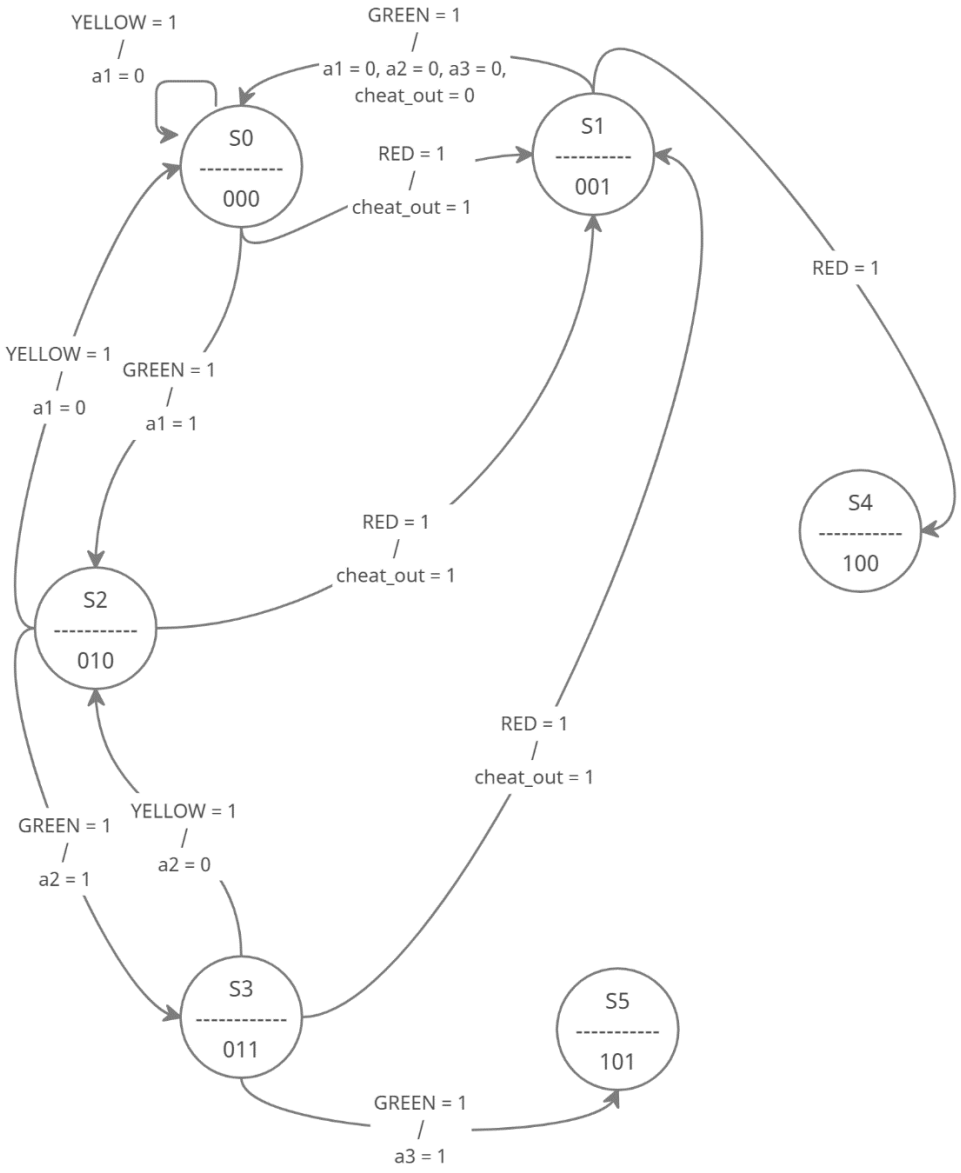
BBM233 Logic Design Laboratory

Fall 2020

Problem Definition

This deadly AI who wants to eradicate all human beings from the Earth, needs our help to achieve its villainous goal. This wicked machine, named SCP-079 by the SCP foundation wants to break free from the computer which it was born. We are asked to design a “Finite State Machine” that simulates the escape attempts of SCP-079.

Mealy State Transition Diagram



My Code

```
module scp_079(timer, clock, green, yellow, red, state, a1, a2, a3, cheat_out);
    input green, yellow, red, clock;
    output [2:0] state;
    output reg [5:0] timer;
    output a1, a2, a3, cheat_out;

    wire green, yellow, red, clock;
    reg [2:0] state;
    reg a1, a2, a3, cheat_out;
    initial begin
        timer = 1;                          //initializer for inputs, state and timer
        state = 0;
        a1 = 0;
        a2 = 0;
        a3 = 0;
        cheat_out = 0;
    end
                                            //if blocks for transitions
    always @(posedge clock) begin
        assign timer = timer + 1; //timer count is done here
        if(state == 0 && timer > 20 && green)begin  //lay low to attack security state
            assign timer = 1;
            assign state = 2;
            assign a1 = 1;
        end
        else if((state == 0 || state == 2 || state == 3) && red)begin  //lay low, attack sec., attack dat. to cheat state
            assign timer = 1;
            assign state = 1;
            assign cheat_out = 1;
        end
        else if(state == 2 && timer > 10 && green)begin     // attack sec. to attack dat. state
            assign timer = 1;
            assign state = 3;
            assign a2 = 1;
        end
        else if(state == 2 && yellow)begin      //attack sec. to lay low state
            assign timer = 1;
            assign state = 0;
            assign a1 = 0;
        end
        end
        else if(state == 3 && timer > 10 && green)begin  //attack dat. to connect state
            assign timer = 1;
            assign state = 5;
            assign a3 = 1;
        end
        else if(state == 3 && yellow)begin      //attack dat. to attack sec state
            assign timer = 1;
            assign state = 2;
            assign a3 = 0;
        end
        else if(state == 1 && timer > 15 && red)begin //cheat to fail state
            assign timer = 1;
            assign state = 4;
        end
        else if(state == 1 && timer > 15 && green)begin // cheat to lay low state
            assign timer = 1;
            assign state = 0;
            assign a1 = 0;
            assign a2 = 0;
            assign a3 = 0;
            assign cheat_out = 0;
        end
    end
endmodule
```

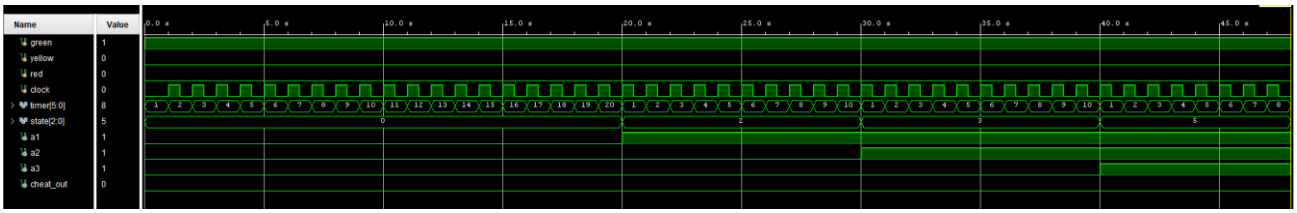
Explanation

I used an always loop with for both timer and state transitions. I checked for the state transitions with if blocks. I wrote an if block for each transition.

Testbenches

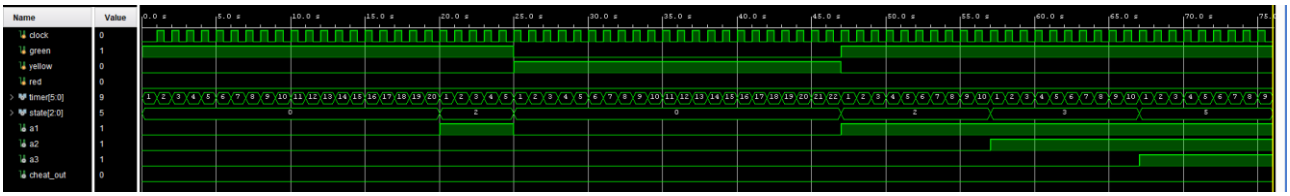
Successful Connection

```
`timescale 1ns / 1ps
`include "scp_079.v"
module allok_tb;
    reg green, yellow, red, clock;
    wire [5:0] timer;
    wire [2:0] state;
    wire a1;
    wire a2;
    wire a3;
    wire cheat_out;

    initial begin
        clock = 0;
        #0.5;
        forever begin
            #0.5 clock = ~clock;
        end
    end

    scp_079 UUT(.green(green), .clock(clock),
     .timer(timer), .yellow(yellow), .red(red),
      .state(state), .a1(a1), .a2(a2), .a3(a3),
       .cheat_out(cheat_out));

    initial begin
        green=1; red=0; yellow = 0;
        #48 $finish;
    end

endmodule
```



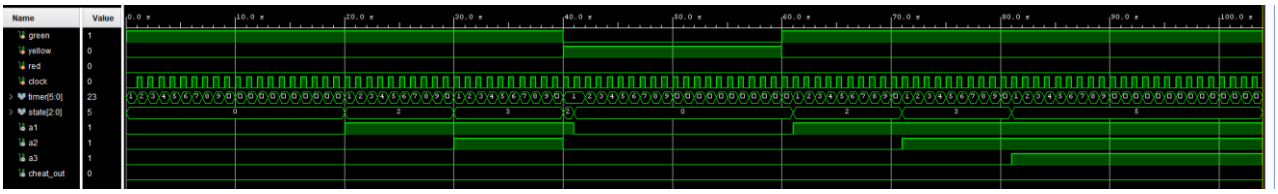
Trouble in Attack Security State

```
initial begin
    green = 1; yellow = 0; red = 0;
    #25 green = 0; yellow = 1;
    #22 green = 1; yellow = 0;
    #29 $finish;
end
```



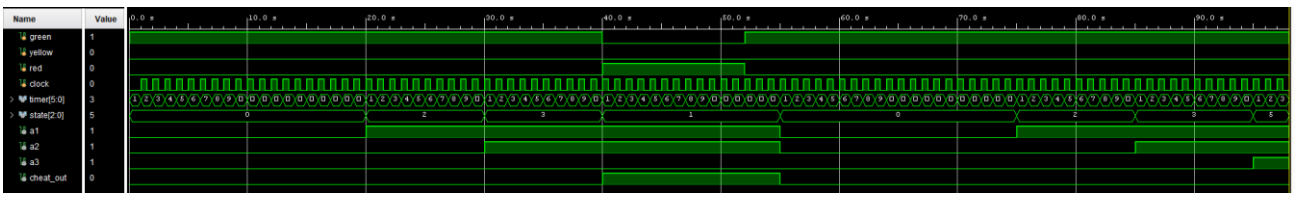
Trouble in Attack Database State

```
initial begin
    green = 1; yellow = 0; red = 0;
    #40 green = 0; yellow = 1;
    #20 green = 1; yellow = 0;
    #44 $finish;
end
```



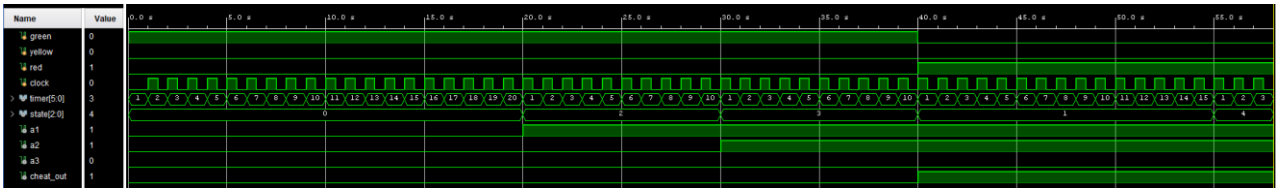
Successful Cheat

```
initial begin
    green = 1; yellow = 0; red = 0;
    #40 green = 0; red = 1;
    #12 green = 1; red = 0;
    #46 $finish;
end
```



Failure

```
initial begin
    green = 1; yellow = 0; red = 0;
    #40 green = 0; red = 1;
    #12 green = 1; red = 0;
    #46 $finish;
end
```



As you can see all my waveforms matches with the ones you have shown to us in your pdf file.

Resources

1. http://www.asic-world.com/verilog/first1.html