# Airport Control System

Airport system includes a total of 4 inputs with 1 clock signal and 2 outputs. The first of these inputs is "weather". This parameter tells us whether the weather conditions are suitable for the landing and take-off of aircraft. When we examine the code part soon, we will observe that this input works as a control input. A second input is "passengers". This input has 6 indexes. (0 to 5) . This input tells us if there are enough passengers in the waiting area. When the number of passengers in the waiting area exceeds 60, the yellow light turns on in the 1st or 2nd LED, depending on the situation. This yellow light tells the pilot to take off but be careful. Our last input is the "airfield". According to this input value, it tells whether the airport is available or not. ( Is there a plane at the airport ). In addition, if there is a waiting aircraft, it is decided what the aircraft will do according to this value. And as I just said, we finally have 2 outputs , these outputs are "led1" and "led2". These outputs can take 3 different colors as seen in Figure1 . (Green , red and yellow)

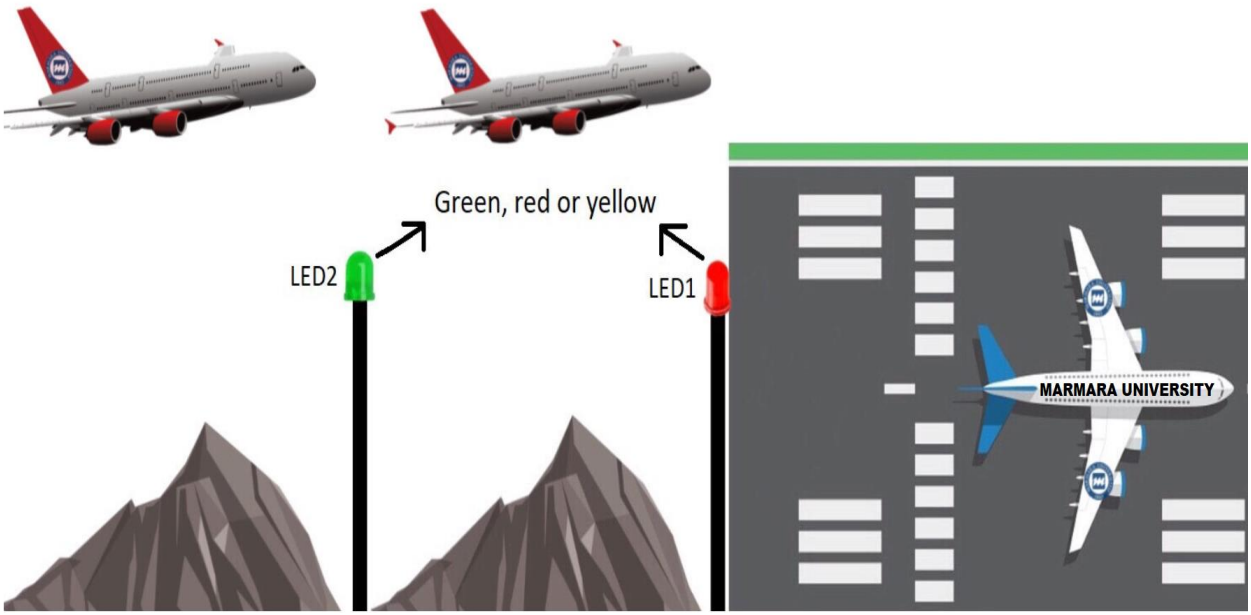


*Figure 1: Project visualization*


*Figure 2: LED light commands*

## AIRPORT RULES

1 ) The waiting area can hold a maximum of 60 people.

2) If there is an emergency, that is, if there are more than 60 people in the waiting area, the another aircraft can land.

3) If there is no emergency, there can only be 1 plane at the airport.

4) If there is an emergency, a maximum of 2 aircraft can be on the airfield at the same time.

This project is not a traffic lights project. Here, the green and red light of the led is related to the emergency situation at the airport and how many planes are in the airspace. The yellow led is a more detailed situation. If suitable conditions are provided and an emergency is required, the yellow LED is activated and 2 planes are allowed to land at the airport. In this case, the pilot who sees the yellow led realizes that there was another plane at the airport road before and behaves more carefully ,and If LED1 is green, LED2 can be positioned as yellow according to the emergency situation and the number of aircraft.

```
`timescale 1ns / 1ps

module airfield_control_system(clk,passengers,weather,airfield,led1,led2);

input clk;
input weather; // weather conditions are favorable or unsuitable. 0 means bad weather
// The weather condition will act as a control input.
input [5:0] passengers; // capacity could be max 60
input airfield; // airfield availability , "is there a plane or not" 0 means available

output reg[2:0] led1; // led1 is the right led in the photo.
output reg[2:0] led2; //led2 is the left led in the photo.

//Led1 has 2 color , led2 has 3 color.
// led[0] --> green (keep moving)
// led[1] --> red (wait)
// led[2] --> yellow ( keep moving but be careful)

initial begin

     led1[0] = 0; // In the first case, the airfield is not available and the led turns red.
     led1[1] = 1;
     led1[2] = 0;
     led2[0] = 0; // If led1 is red, led2 should light up red.
     led2[1] = 1;
     led2[2] = 0;

end
```

*Figure 3: Inputs/Outputs & Inital begin values for project.*

Since the clock signal we created in our project will work continuously and we use a behavioral modeling, the assignments in the always block we used non-blocking assignment.

```
always @ (posedge clk)
begin
    if (airfield == 0) // runway available
    begin
        if ( weather == 1 )
        begin
            if ( passengers < 60) //its not emergency and
            begin                 //could be only 1 plane
                led1[0] <= 1;
                led1[1] <= 0 ;    // We turned our LED1 on green
                led2[2] <= 0;     // and LED2 on red.
                led2[0] <= 0;
                led2[1] <= 1;
                led2[2] <= 0;
            end
            else // if passengers > 60) // The waiting area is crowded,
            begin                      //so an emergency is declared.
                led1[0] <= 1;
                led1[1] <= 0; //Since there was no aircraft at the airport before,
                led1[2] <= 0; //although LED1 was green, LED2 turns yellow.
                led2[0] <= 0;
                led2[1] <= 0;
                led2[2] <= 1;
            end
       end
    end
    if (airfield == 0)
    begin                 //If the weather conditions are not suitable,
        if (weather == 0 ) //take-off and landing are not allowed.
        begin
            led1[0] <= 0;
            led1[1] <= 1;
            led1[2] <= 0;
            led2[0] <= 0;
            led2[1] <= 1;
            led2[2] <= 0;
        end
    end
```

```
    if (airfield == 1) //there is plane at runway
    begin
        if (weather == 1)
        begin
            if (passengers > 60)
            begin
                led1[0] <= 0;
                led1[1] <= 0;
                led1[2] <= 1;
                led2[0] <= 0;
                led2[1] <= 1;
                led2[2] <= 0;
            end
            else //if passengers < 60)
            begin
                led1[0] <= 0;
                led1[1] <= 1;
                led1[2] <= 0;
                led2[0] <= 0;
                led2[1] <= 1;
                led2[2] <= 0;
            end
       end
    end
  if (airfield == 1)
    begin
        if (weather == 0)
        begin
            led1[0] <= 0;
            led1[1] <= 1;
            led1[2] <= 0;
            led2[0] <= 0;
            led2[1] <= 1;
            led2[2] <= 0;
        end
    end
end
endmodule
```

```
`timescale 1ns / 1ps
module airfield_control_system_tb();
reg clk , weather , airfield;
reg [5:0] passengers;
wire [2:0] led1 , led2;

airfield_control_system UUT(.clk(clk),.weather(weather),
.passengers(passengers),
.airfield(airfield),.led1(led1),.led2(led2)
);
initial begin
clk = 0;
passengers = 6'b0;
#10;

airfield = 0;
weather = 1;
passengers = 6'b001111;
#20;

airfield = 0;
weather = 1;
passengers = 6'b111110;
#20;

airfield = 1;
weather = 1;
passengers = 6'b001111;
#20;

airfield = 1;
weather = 1;
passengers = 6'b111101;
#20;

airfield = 1;
weather = 0;
passengers = 6'b111110;
#20;
end

always
begin
#10;
clk =~ clk;
end
endmodule
```

We set the period of our clock signal to 20ns and we changed our initial values every 20ns in testbench and checked in the simulation to see if our code was working correctly.

It was observed that our code worked correctly because the outputs in the simulation and our own interests overlapped.

For example ;

airfield = 0; weather = 1;passengers = 6'b111110;

For this conditions , we can see that our LED1 is green and our LED 2 is yellow. Because,in case of emergency, the airport could take 2 planes.

*Figure 4: Main code (above page) and Testbench*

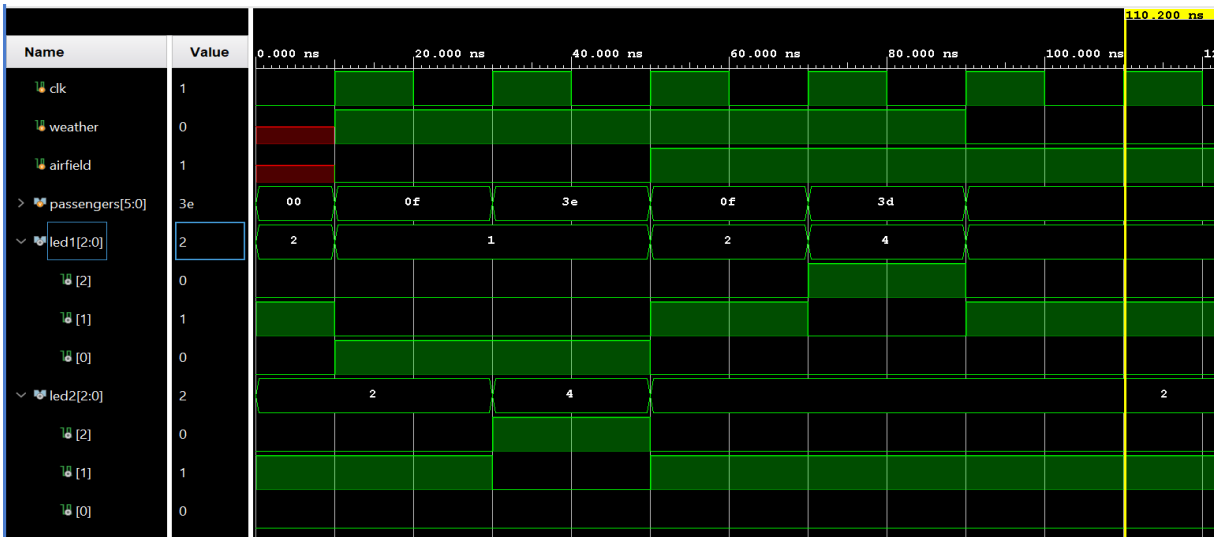


Figure 5: Simulation Result

| Airfield | Passengers (>60 == 1) | Weather | Green LED1 | Red LED1 | Yellow LED1 | Green LED2 | RED LED2 | Yellow LED2 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

*Table 1: Truth table for Airport Control System*