Project 2 – Traffic Light Controller

1. Design

```
`timescale 1ns / 1ps
module traffic_light_ctrl_eng(
//port declaration
//inputs
input wire [4:0] i_NS_count       ,
input wire [3:0] i_EW_count       ,
input wire [1:0] yellow_count     ,
input wire       NS_vehicle_detect ,
input wire       EW_vehicle_detect ,

//outputs
output reg NS_red      ,
output reg NS_yellow   ,
output reg NS_green    ,
output reg EW_red      ,
output reg EW_yellow   ,
output reg EW_green
);

//initialization
initial begin
   NS_red     <=1'b0;
   NS_yellow  <=1'b0;
   NS_green   <=1'b1;
   EW_red     <=1'b1;
   EW_yellow  <=1'b0;
   EW_green   <=1'b0;
end
```

```
//NS_controller
always @ (i_NS_count) begin

   if (i_NS_count==31 & EW_vehicle_detect & NS_green) begin
      NS_red     <=1'b0;
      NS_yellow   <=1'b1;
      NS_green    <=1'b0;
      EW_red     <=1'b1;
      EW_yellow   <=1'b0;
      EW_green   <=1'b0;
   end
end

//NS_controller
always @ (i_EW_count) begin

    if (i_EW_count==15 & NS_vehicle_detect & EW_green) begin
      NS_red     <=1'b1;
      NS_yellow   <=1'b0;
      NS_green    <=1'b0;
      EW_red     <=1'b0;
      EW_yellow   <=1'b1;
      EW_green   <=1'b0;
   end
end

//yellow controller
always @ (yellow_count) begin

   if (yellow_count ==3 & NS_yellow) begin
      NS_red     <=1'b1;
```

```
      NS_yellow   <=1'b0;
      NS_green    <=1'b0;
      EW_red      <=1'b0;
      EW_yellow   <=1'b0;
      EW_green   <=1'b1;
   end
   if (yellow_count ==3 & EW_yellow) begin
      NS_red      <=1'b0;
      NS_yellow   <=1'b0;
      NS_green    <=1'b1;
      EW_red      <=1'b1;
      EW_yellow   <=1'b0;
      EW_green   <=1'b0;
   end
end
endmodule

//NS counter
module NS_count (
//port declaration
input   wire    i_clk        , //input clock
output  reg     [4:0] o_count  //output clock
);

//initialization
initial
   o_count = 0;
always @ (negedge i_clk)
   o_count[0] <= ~o_count[0];
always @ (negedge o_count [0])
   o_count[1] <= ~o_count[1];
```

```
always @ (negedge o_count [1])
   o_count[2] <= ~o_count[2];
always @ (negedge o_count [2])
   o_count[3] <= ~o_count[3];
always @ (negedge o_count [3])
   o_count[4] <= ~o_count[4];
endmodule

//EW counter
module EW_count (
//port declaration
input   wire    i_clk       , //input clock
output  reg     [3:0] o_count  //output clock
);

//initialization
initial
   o_count =0;
always @ (negedge i_clk)
   o_count[0] <= ~o_count[0];
always @ (negedge o_count [0])
   o_count[1] <= ~o_count[1];
always @ (negedge o_count [1])
   o_count[2] <= ~o_count[2];
always @ (negedge o_count [2])
   o_count[3] <= ~o_count[3];
endmodule
```

```
//yellow counter
module yellow_count (
//port declaration
input   wire    i_clk        , //input clock
output  reg     [1:0] o_count  //output clock
);

//initialization
initial
   o_count = 0;
always @ (negedge i_clk)
   o_count[0] <= ~o_count[0];
always @ (negedge o_count [0])
   o_count[1] <= ~o_count[1];
endmodule
```

2. Test bench

```
`timescale 1ns / 1ps
module traffic_tb;
//internal wires/regs
wire [4:0] i_NS_count   ;
wire [3:0] i_EW_count   ;
wire [1:0] yellow_count ;
reg     CLK           ;

//inputs
reg NS_vehicle_detect;
reg EW_vehicle_detect;

//outputs
wire NS_red       ;
wire NS_yellow     ;
wire NS_green      ;
wire EW_red       ;
wire EW_yellow     ;
wire EW_green      ;

//initial block
initial begin
   CLK             =1'b0;
   NS_vehicle_detect =1'b1;
   EW_vehicle_detect =1'b0;
   $display ("   NS  |  EW  ");
   $display (" R   Y   G   R   Y   G ");
   $monitor (" %h  %h  %h  %h  %h
%h",NS_red,NS_yellow,NS_green,EW_red,EW_yellow,EW_green);
```

```
   #1000 $finish;
end

//clock generator
always
   #5 CLK = ~CLK    ;
 //test case 2
 always @ (CLK) begin
   if ($time % 26 ==0) begin
      NS_vehicle_detect=~NS_vehicle_detect;
      EW_vehicle_detect=~EW_vehicle_detect;
   end
 end
 //end of test case 2

//traffic core
traffic_light_ctrl_eng CORE(
//inputs
.i_NS_count        (i_NS_count),
.i_EW_count        (i_EW_count),
.yellow_count      (yellow_count),
.NS_vehicle_detect  (NS_vehicle_detect),
.EW_vehicle_detect  (EW_vehicle_detect),
//outputs
.NS_red     (NS_red)   ,
.NS_yellow   (NS_yellow),
.NS_green    (NS_green) ,
.EW_red     (EW_red)   ,
.EW_yellow   (EW_yellow),
.EW_green    (EW_green)
);
```

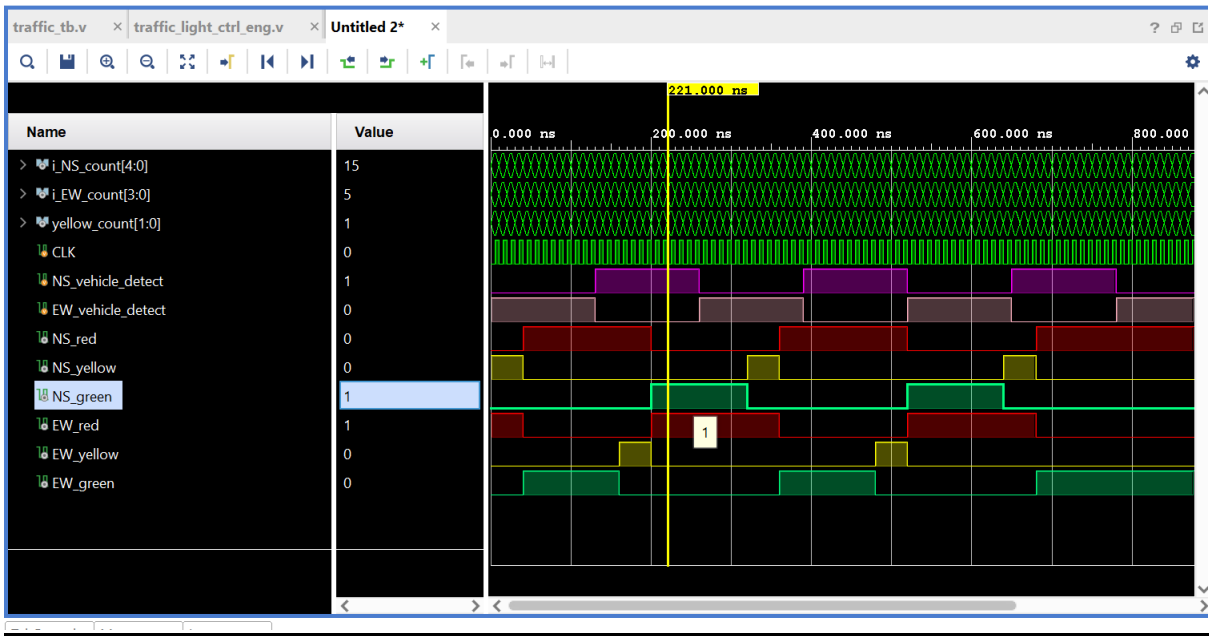
```
//north south counter
NS_count i_NS_count_0(
.i_clk    (CLK),
.o_count  (i_NS_count)
);

//east west counter
EW_count i_EW_count_0(
.i_clk    (CLK),
.o_count  (i_EW_count)
);

//yellow light counter
yellow_count i_yellow_count_0(
.i_clk    (CLK),
.o_count  (yellow_count)
);
endmodule
```

3. Simulation Results

a. when NS_vehicle detected



b. when EW_vehicle detected