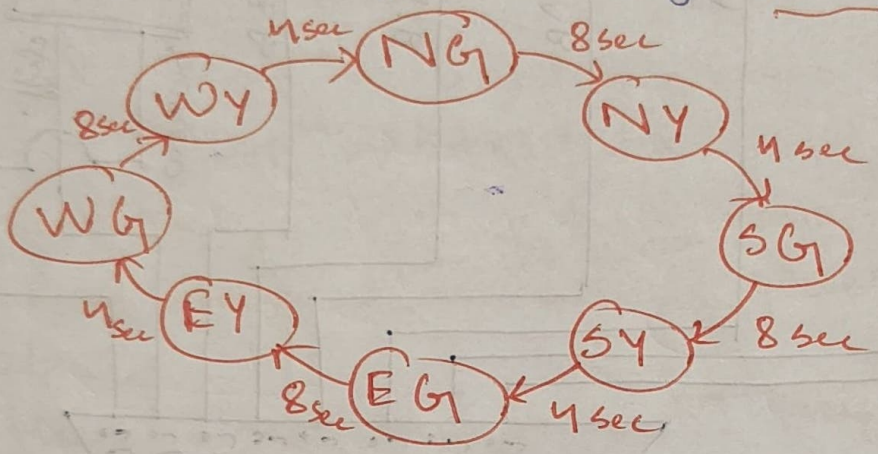
Traffic Signal Controller :-

- Consider a Traffic signal consisting of 4 direct^{n} (North, south, East and west) $\Rightarrow$ [N, S, E, W]
- Each dir^{n} has 3 bit Light [R, G, Y]
  i.e., Red = 000, Green = 001, Yellow = 010
- The controller steps through 8 states



- Remember logic

(1) when NG, then remaining direct^{n} are red
i.e :- North $\Rightarrow$ 001 $\rightarrow$ S, E, W $\Rightarrow$ 000
A direct^{n} Green remains for 8 sec then turn Yellow.

(2) when NG $\rightarrow$ NY, remaining direct^{n} are red
i.e :- N(001) $\xrightarrow{8}$ N(010) $\Rightarrow$ S, E, W $\Rightarrow$ 000
A dir^{n} Yellow remains for 4 sec then switches to next state ($\rightarrow$E $\rightarrow$S $\rightarrow$W)

(3) After state change NY $\rightarrow$ EG, North becomes Red by default i.e., NY(010) $\rightarrow$ NR(000)

so after state change

NY (010) → EG (001)

↓

NR (000)

So, when East ⇒ Green then remaining is Red i.e (N, S, W ⇒ 000)

(4) And this cycle continues.

Structure of circuit :-

(1) FSM based state model
- (1) dirⁿ change
- (2) Light change

(2) Counter controlled Timer
- (1) 8sec (G)
- (2) 4sec (Y)
- (3) Default (0)

(3) Reset (Asynch)
- (1) counter rst to 0 at every state change
- (2) Default Rst to North

inputs :- (1) clk (2) rst

output :- (1) n_light (2) s_light (3) e_light (4) w_light

internal reg. and param :-

(1) State ⇒ reg [2:0] ⇒ holds current state (dirⁿ & colour)

(2) count ⇒ reg [2:0] ⇒ counts clock cycles to define signal durⁿ

(3) north, north_y, east ... ⇒ parameter ⇒ encodes state for FSM opⁿ

Verilog code declaration

```
module traffic_control (
  input clk, rst_a;
  output reg [2:0] n_light, s_light, e_light,
                                          w_light
  );

  reg [2:0] state;   // defines the state (current)
  reg [2:0] count;   // counts till 7(111) & 3(011)
                        and rst when state change

  parameter [2:0] north = 3'b000;
  parameter [2:0] north_y = 3'b001;
       "                "        "
       :                :        :
  parameter [2:0] west = 3'b110;
  parameter [2:0] west_y = 3'b111;
```

* declared states assignment.

initialisatn

```
  always @ (posedge clk, posedge rst_a)
  begin
    if (rst_a)          // if rst_a is true or 1
    begin
      state = north;    // initialise state
      count = 3'b000;   // initialise count
    end
    else begin          // if rst_a = 0
      Case (state)
```

State-transitions :- (will be continued for other 3 dirⁿ)

```
case (state)
  north:
    begin
      n_light = 3'b001;   // Green
      s_light = 3'b000;   // red
      e_light = 3'b000;   // red
      w_light = 3'b000;   // red
    if (count == 3'b111) ⟹ // t = 8sec
      begin
        count = 3'b000;   // count rst
        state = north_y;  // next state
      end
      else begin
        count = count + 3'b001;   // count increment
      end
    end
  north_y:
    begin
      n_light = 3'b010;   // yellow
    if (count == 3'b011) ⟹ // t = 4sec
      begin
        count = 3'b000;   // count rst
        state = East;     // next state = East
      end
      else begin
        count = count + 3'b001;   // increment
      end
    end
```

Light states G = 001
y = 010
R = 000