OBJECTIVE:

(i) To increase the coding efficiency in Verilog by giving some complex program.

TOPIC: 🚦 Traffic Light Controller using Verilog

This Verilog code is a Finite State Machine (FSM)-based Traffic Light Controller designed for four-way intersections (North, East, South, and West). It operates in a cyclic manner, controlling traffic lights using a clock signal

Steps:

1. inputs and Outputs

- clk → Clock signal (controls transitions between states).
- reset → Asynchronous reset (resets the system to the initial state).
- north_light, east_light, south_light, west_light → 3-bit outputs representing the color of the traffic light (Red, Yellow, or Green).

2. Traffic Light Encoding (3-bit values)

Light	Binary Code
Red	100
Yellow	010
Green	001

Each direction (North, East, South, West) has an associated 3-bit output that determines which light is ON.

3. State Machine (FSM)

The system operates using 8 states, where each direction gets a Green light in sequence.

State Name	Binary Code	Active Light
NORTH_GREEN	0000	North → Green
NORTH_YELLOW	0001	North → Yellow
EAST_GREEN	0010	East → Green
EAST_YELLOW	0011	East → Yellow
SOUTH_GREEN	0100	South → Green
SOUTH_YELLOW	0101	South → Yellow
WEST_GREEN	0110	West → Green
WEST_YELLOW	0111	West → Yellow

The cycle repeats continuously.

4. State Transition Logic

- The always @(posedge clk or posedge reset) block is responsible for transitioning from one state to another.
- The **timer** ensures a delay (~1 second) before moving to the next state.

Each traffic light remains in a state for 1 second before transitioning.

5. Next State Logic

This is a combinational block that determines the next state of the FSM.

Ensures the correct sequence of traffic light changes.

6. Output Logic (Traffic Light Control)

- **Default all lights to RED.**
- Based on the current state, one direction gets GREEN or YELLOW.

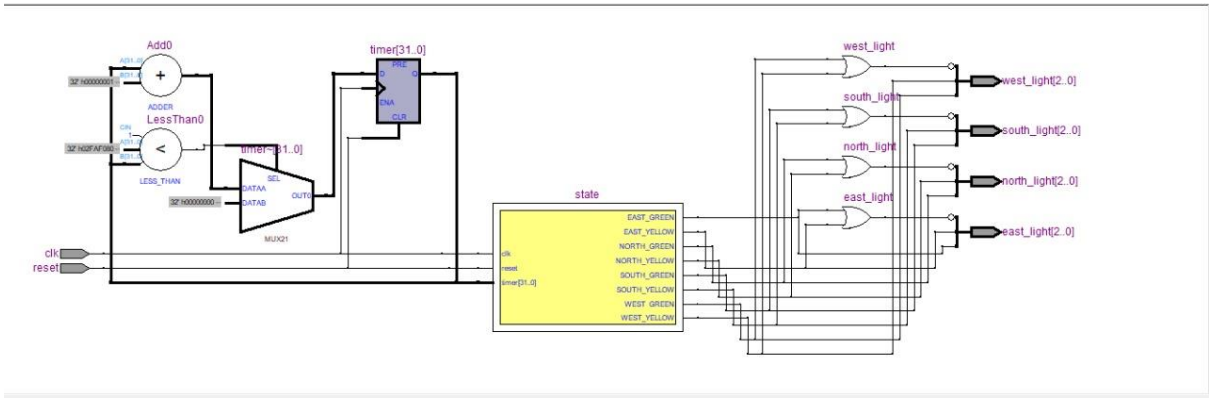
Only one direction gets GREEN at a time

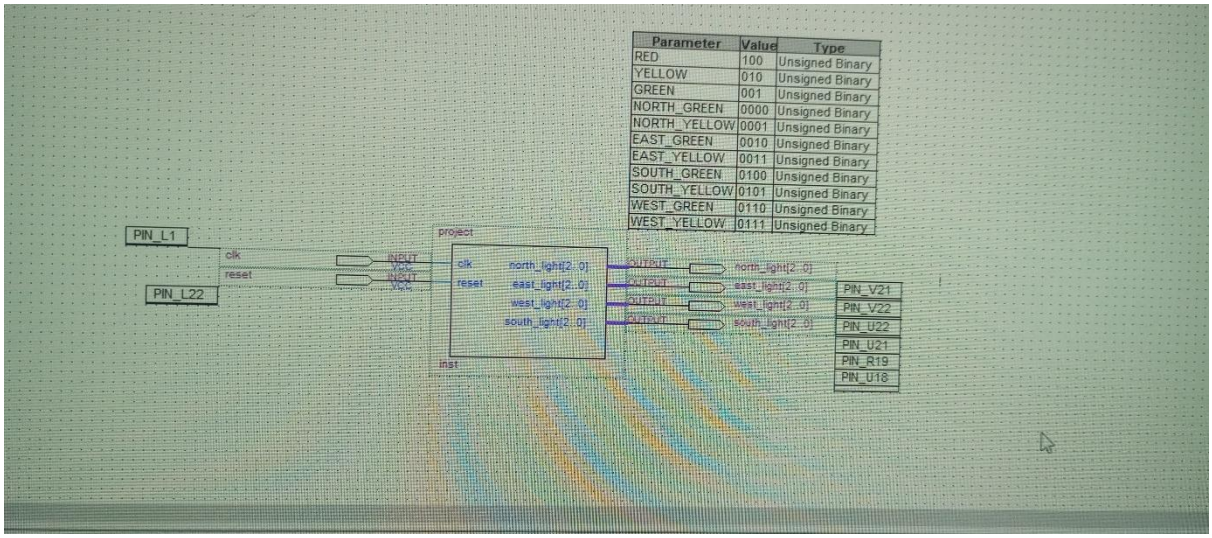
Code:

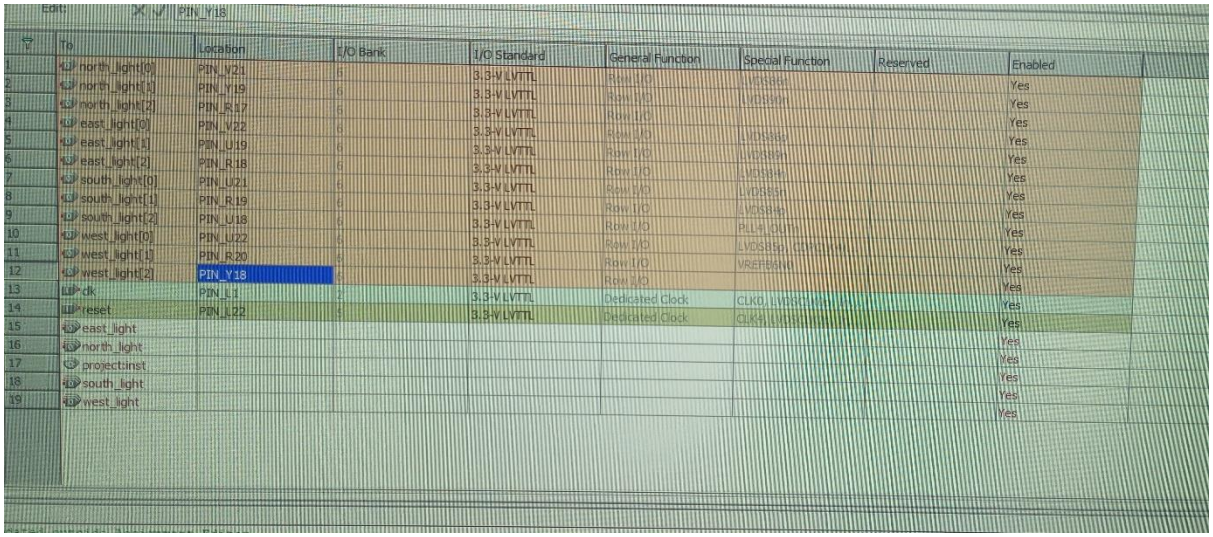
```
module project(
   input clk,          // Clock signal
   input reset,        // Asynchronous reset
   output reg [2:0] north_light,  // North Traffic Light {Red, Yellow, Green}
   output reg [2:0] east_light,   // East Traffic Light {Red, Yellow, Green}
   output reg [2:0] south_light,  // South Traffic Light {Red, Yellow, Green}
   output reg [2:0] west_light    // West Traffic Light {Red, Yellow, Green}
);

// Traffic Light Encoding (3-bit)
parameter RED = 3'b100;
parameter YELLOW = 3'b010;
parameter GREEN = 3'b001;

// State Encoding (4-bit)
parameter NORTH_GREEN  = 4'b0000;
parameter NORTH_YELLOW = 4'b0001;
parameter EAST_GREEN   = 4'b0010;
parameter EAST_YELLOW  = 4'b0011;
parameter SOUTH_GREEN  = 4'b0100;
parameter SOUTH_YELLOW = 4'b0101;
parameter WEST_GREEN   = 4'b0110;
parameter WEST_YELLOW  = 4'b0111;
```

```
// Registers to hold current and next state
reg [3:0] state, next_state;

// Timer Counter for delay
reg [31:0] timer;

// State Transition Logic (Clock-Driven FSM)
always @(posedge clk or posedge reset) begin
   if (reset) begin
      state <= NORTH_GREEN;  // Start with North Green
      timer <= 0;
   end else begin
      if (timer >= 32'd50_000_000) begin  // Assuming 50M cycles (~1 sec delay)
         state <= next_state;
         timer <= 0;
      end else begin
         timer <= timer + 1;
      end
   end
end

// Next State Logic (Combinational Logic)
always @(*) begin
   case (state)
      NORTH_GREEN:  next_state = NORTH_YELLOW;
      NORTH_YELLOW: next_state = EAST_GREEN;
      EAST_GREEN:   next_state = EAST_YELLOW;
      EAST_YELLOW:  next_state = SOUTH_GREEN;
      SOUTH_GREEN:  next_state = SOUTH_YELLOW;
      SOUTH_YELLOW: next_state = WEST_GREEN;
      WEST_GREEN:   next_state = WEST_YELLOW;
      SOUTH_YELLOW: next_state = WEST_GREEN;
      WEST_GREEN:   next_state = WEST_YELLOW;
      WEST_YELLOW:  next_state = NORTH_GREEN;
      default:      next_state = NORTH_GREEN;
   endcase
end

// Traffic Light Output Logic
always @(*) begin
   // Default all lights to RED
   north_light = RED;
   east_light  = RED;
   south_light = RED;
```

```
   west_light  = RED;

   case (state)
      NORTH_GREEN:  north_light = GREEN;
      NORTH_YELLOW: north_light = YELLOW;
      EAST_GREEN:   east_light  = GREEN;
      EAST_YELLOW:  east_light  = YELLOW;
      SOUTH_GREEN:  south_light = GREEN;
      SOUTH_YELLOW: south_light = YELLOW;
      WEST_GREEN:   west_light  = GREEN;
      WEST_YELLOW:  west_light  = YELLOW;
   endcase
end
endmodule
```






Edit:
PIN_Y18
To
Location
I/O Bank
I/O Standard
General Function
Special Function
Reserved
Enabled
north_light[0]
PIN_V21
north_light[1]
PIN_Y19
north_light[2]
PIN_R17
east_light[0]
PIN_V22
east_light[1]
PIN_U19
east_light[2]
PIN_R18
south_light[0]
PIN_U21
south_light[1]
PIN_R19
south_light[2]
PIN_U18
west_light[0]
PIN_U22
west_light[1]
PIN_R20
west_light[2]
PIN_Y18
clk
PIN_L1
reset
PIN_L22
east_light
north_light
project:inst
south_light
west_light
3.3-V LVTTL
Dedicated Clock
Yes