# Time-Based 4 Traffic Lights Controller Circuit

## 1. Objectives:

A. To build and understand finite state machines.
B. To understand Moore finite state machine.
C. Implementing a time-based traffic light controller using finite state machine.
D. Integrating smaller modules.
E. Working in teams.
F. Develop problem solving skills.

## 2. Breakdown of the Project

### a. State Table for TLC (Moore)

| Current State | Input (Time) | | | | Next State | Output | | | |
|---|---|---|---|---|---|---|---|---|---|
| | ***T4*** | ***T3*** | ***T2*** | ***T1*** | | ***S4*** | ***S3*** | ***S2*** | ***S1*** |
| STATE1 | 15 | 10 | 5 | 5 | STATE1 | 0 | 0 | 0 | 1 |
| STATE1 | 14 | 9 | 4 | 4 | STATE1 | 0 | 0 | 0 | 1 |
| STATE1 | 13 | 8 | 3 | 3 | STATE1 | 0 | 0 | 0 | 1 |
| STATE1 | 12 | 7 | 2 | 2 | STATE1 | 0 | 0 | 0 | 1 |
| STATE1 | 11 | 6 | 1 | 1 | STATE1 | 0 | 0 | 0 | 1 |
| STATE1 | 10 | 5 | 0 | 0 | STATE2 | 0 | 0 | 0 | 1 |
| STATE2 | 10 | 5 | 5 | 15 | STATE2 | 0 | 0 | 1 | 0 |
| STATE2 | 9 | 4 | 4 | 14 | STATE2 | 0 | 0 | 1 | 0 |
| STATE2 | 8 | 3 | 3 | 13 | STATE2 | 0 | 0 | 1 | 0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| STATE2 | 7 | 2 | 2 | 12 | STATE2 | 0 | 0 | 1 | 0 |
| STATE2 | 6 | 1 | 1 | 11 | STATE2 | 0 | 0 | 1 | 0 |
| STATE2 | 5 | 0 | 0 | 10 | STATE3 | 0 | 0 | 1 | 0 |
| STATE3 | 5 | 5 | 15 | 10 | STATE3 | 0 | 1 | 0 | 0 |
| STATE3 | 4 | 4 | 14 | 9 | STATE3 | 0 | 1 | 0 | 0 |
| STATE3 | 3 | 3 | 13 | 8 | STATE3 | 0 | 1 | 0 | 0 |
| STATE3 | 2 | 2 | 12 | 7 | STATE3 | 0 | 1 | 0 | 0 |
| STATE3 | 1 | 1 | 11 | 6 | STATE3 | 0 | 1 | 0 | 0 |
| STATE3 | 0 | 0 | 10 | 5 | STATE4 | 0 | 1 | 0 | 0 |
| STATE4 | 5 | 15 | 10 | 5 | STATE4 | 1 | 0 | 0 | 0 |
| STATE4 | 4 | 14 | 9 | 4 | STATE4 | 1 | 0 | 0 | 0 |
| STATE4 | 3 | 13 | 8 | 3 | STATE4 | 1 | 0 | 0 | 0 |
| STATE4 | 2 | 12 | 7 | 2 | STATE4 | 1 | 0 | 0 | 0 |
| STATE4 | 1 | 11 | 6 | 1 | STATE4 | 1 | 0 | 0 | 0 |
| STATE4 | 0 | 10 | 5 | 0 | STATE1 | 1 | 0 | 0 | 0 |

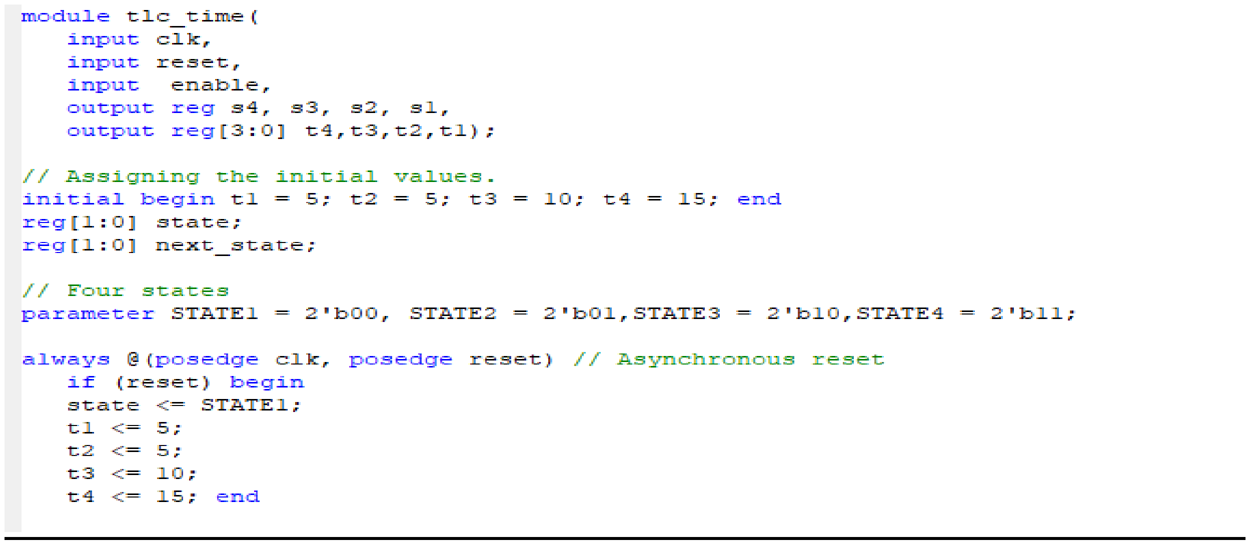
*Table 1. State Table for Traffic Light Controller*

## b. Verilog Module for Pulse Generator (One Hertz)

```
module onehz( input clk, reset, enable, output clk1hz);
reg[26:0] counter;
// To generate a signal with a frequency of 1hz from 100Mhz, the counter should count 100 000 000 cycles.
assign clk1hz = (counter == 99999999);
always@(posedge clk,posedge reset)
if (reset || clk1hz )
   counter <= 0;
else if (enable)
   counter <= counter + 1;
endmodule
```

*Figure 1. Verilog module for one hertz pulse generator.*

## c. Verilog Module for TLC and Simulation



```
module tlc_time(
   input clk,
   input reset,
   input  enable,
   output reg s4, s3, s2, s1,
   output reg[3:0] t4,t3,t2,t1);

// Assigning the initial values.
initial begin t1 = 5; t2 = 5; t3 = 10; t4 = 15; end
reg[1:0] state;
reg[1:0] next_state;

// Four states
parameter STATE1 = 2'b00, STATE2 = 2'b01,STATE3 = 2'b10,STATE4 = 2'b11;

always @(posedge clk, posedge reset) // Asynchronous reset
   if (reset) begin
   state <= STATE1;
   t1 <= 5;
   t2 <= 5;
   t3 <= 10;
   t4 <= 15; end
   // Each time signal will be counted down and assigned new values accordingly upon reaching 0.
   else if(enable) begin
   if (t1 == 0 && t2 == 0) begin t1 <= 15 ; t2 <= 5; state <= next_state; end
   else if ( t2 == 0 && t3 == 0) begin t2 <= 15 ; t3 <= 5 ;state <= next_state; end
   else if ( t3 == 0 && t4 == 0) begin t3 <= 15 ; t4 <= 5 ;state <= next_state; end
   else if ( t4 == 0 && t1 == 0) begin t4 <= 15 ; t1 <= 5 ;state <= next_state; end
   else begin
      t1 <= t1 - 1;
      t2 <= t2 - 1;
      t3 <= t3 - 1;
      t4 <= t4 - 1;
      state <= next_state;
      end
   end

   // Inner always block to detect changes in the state and times, and accordingly change the state.
   // Moore finite state machine
   always @(state,t1,t2,t3,t4)begin
   s1 = 0; s2 = 0; s3 = 0; s4 = 0;
   case (state)
   STATE1:
   begin s1 = 1 ; if (t1 == 0 && t2 == 0) next_state = STATE2;
            else next_state = STATE1;  end
   STATE2:
   begin s2 = 1 ; if (t2 == 0 && t3 == 0) next_state = STATE3;
            else next_state = STATE2;  end
   STATE3:
   begin s3 = 1 ; if (t3 == 0 && t4 == 0) next_state = STATE4;
            else next_state = STATE3;  end
   STATE4:
   begin s4 = 1 ; if (t4 == 0 && t1 == 0) next_state = STATE1;
            else next_state = STATE4;  end
   endcase
   end
endmodule
```

*Figure 2. Verilog Module for Traffic Light Controller.*

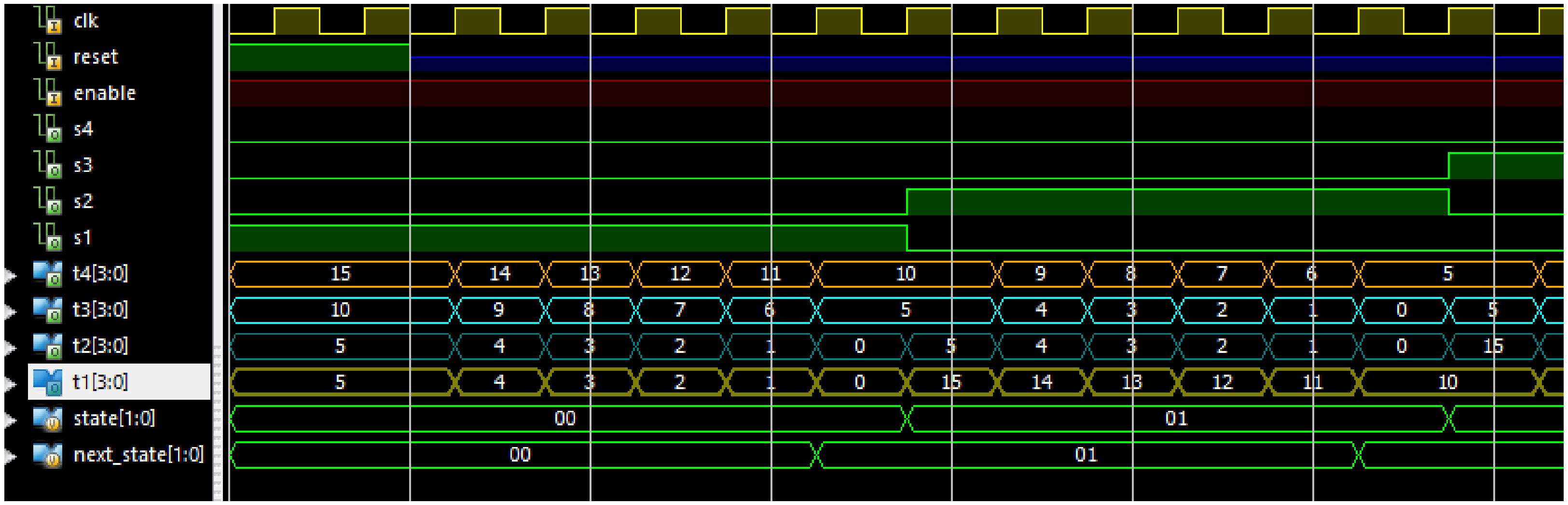


*Figure 3. Simulation of Traffic Light Controller for STATE 1 and STATE 2.*

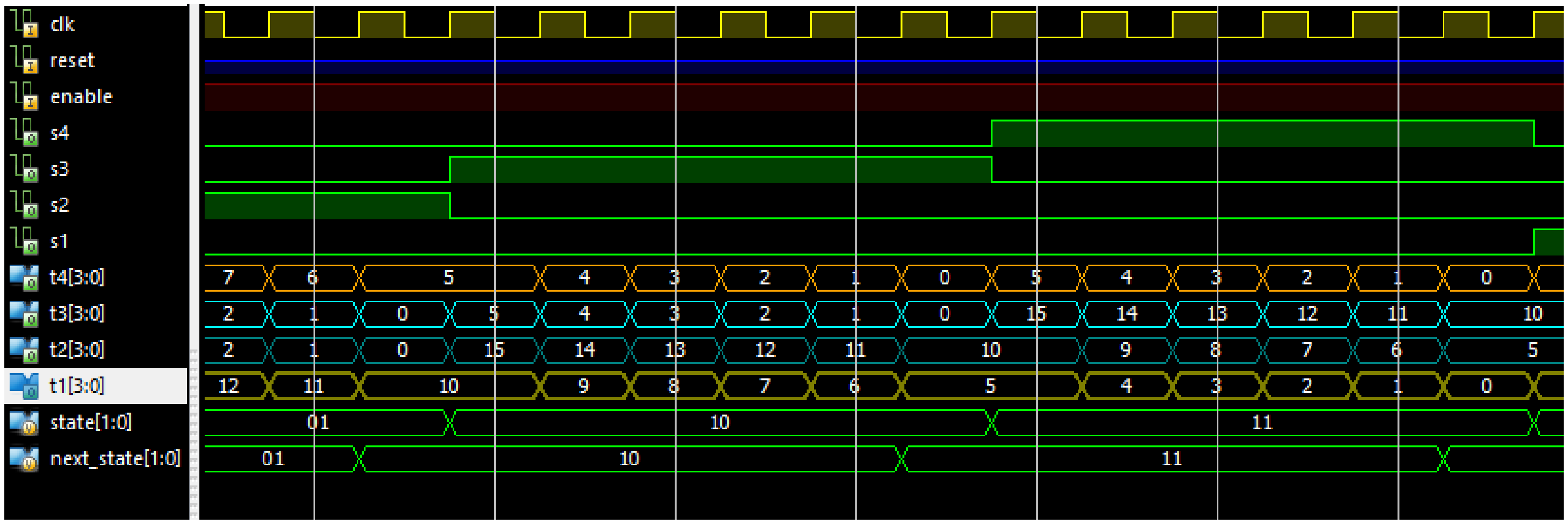
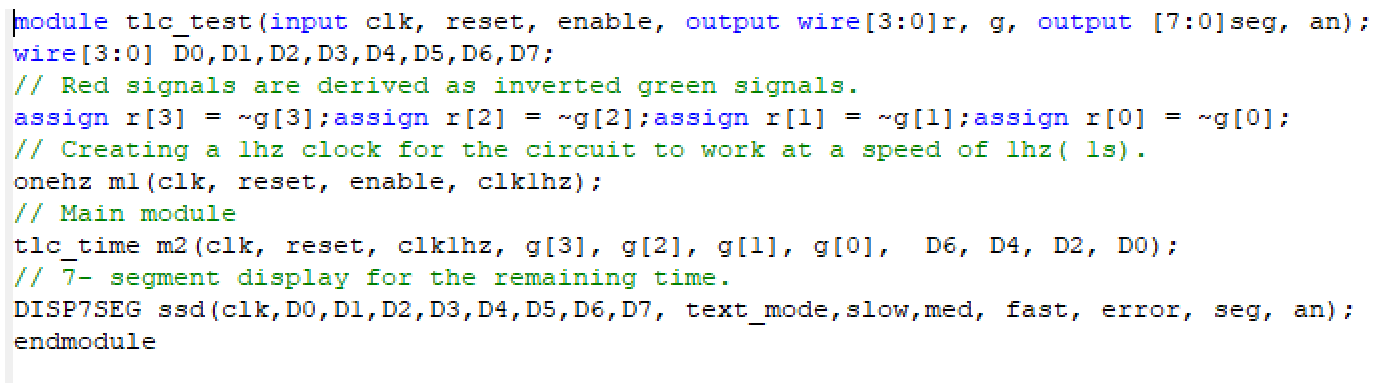


*Figure 4. Simulation of Traffic Light Controller for STATE 3 and STATE 4.*

### d. Verilog Module for TLC test



```
module tlc_test(input clk, reset, enable, output wire[3:0]r, g, output [7:0]seg, an);
wire[3:0] D0,D1,D2,D3,D4,D5,D6,D7;
// Red signals are derived as inverted green signals.
assign r[3] = ~g[3];assign r[2] = ~g[2];assign r[1] = ~g[1];assign r[0] = ~g[0];
// Creating a 1hz clock for the circuit to work at a speed of 1hz( 1s).
onehz m1(clk, reset, enable, clk1hz);
// Main module
tlc_time m2(clk, reset, clk1hz, g[3], g[2], g[1], g[0],  D6, D4, D2, D0);
// 7- segment display for the remaining time.
DISP7SEG ssd(clk,D0,D1,D2,D3,D4,D5,D6,D7, text_mode,slow,med, fast, error, seg, an);
endmodule
```

*Figure 5. Verilog Module for TLC test*

## 3. Challenges Overcome:

Throughout the project, I tackled challenges related to timing synchronization, state transitions, and ensuring the reliability of the simulated traffic flow. Rigorous testing and debugging methods were employed to address these challenges effectively.

## 4. Results and Achievements:

The successful emulation of a traffic signal system on the FPGA showcased the efficient operation of the implemented Moore state machine. The controller accurately managed traffic light sequences, validating its reliability and functionality.

## 5. Learnings and Future Developments:

This project provided invaluable insights into digital circuit design, Verilog programming, and finite state machines. Future developments could include scalability for larger intersections, integration of sensor inputs for adaptive signaling, and optimization for real-time applications.