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. Procedure

Actions to be taken before completing the tasks:

- Create a new project in ISE 14.7
- Note: Double-check the project's details to ensure they apply to the FPGA you'll be using.

a. Task 1: State table for TLC (Moore)

1. Filling the state table for Traffic light controller by following the below rules:
2. This TLC is independent of traffic sensors, it is purely time-based.
3. TLC always starts from state 1.
4. It then moves from state 1 to 2-3-4-1... according to the remaining time of each signal.

b. Task 2: Verilog module for pulse generator (One hertz)

1. Create a Verilog source Onehz.
2. Write code using Clk, Reset and enable as an input, clk1hz as an output. A 27-bit reg for counter.
3. Assign the counter to count to 99999999 to generate 1 hz clk.
4. Create a block that changes with pose edge clk and and asynchronous reset.
5. In always block if statement, reset or clk1hz take the priority make the counter 0 then if enable make counter increase by 1.
6. End module.

c. Task 3: Verilog module for Taffic Light Controller

1. Create a Verilog source for Traffic Light Controller.
2. Write a code for tlc_time making clk, reset, and enable as an input, s4, s3, s2, and s1 as output reg, and a 4-bits output reg for t4, t3, t2, t1.
3. Initially, T4 will be 15, t3 will be 10, t2 and t1 will be 5
4. Create a 2-bits reg for state and 2-bits reg for next-state.
5. Create an always block change with pose edge clk and pose edge reset(Asynchronous reset).
6. If reset will begin with state =state1 and T4 = 15, t3 = 10, t2 and t1 = 5.
7. When the circuit is enabled, the circuit will begin counting and accordingly the states will be changed.
8. t4, t3, t2, t1 will decrease one by one until t1 and t2 reaches 0 then t1 will be 15 and t2 will be 5 the time will remain decrease by one (t3= 5, t4 = 10)
9. An inner always block is designed to model the moore finite state machine with 4 states.
10. Begin with state1 and then state2 -3 -4 -1. In each state one road receives the green signal.
11. Simulate the module.
12. Taking screenshots of simulation by force clocking and force constant appropriate signals.

d. Task 4: Verilog Module for TLC test

1. Create a Verilog source for tlc Test.
2. Clk, reset and enable as input, and t4, t3,t2,t1 s4 s3, s2 and s1 as output.
3. Add onehz as a source.
4. Add Traffic Light Controller a source.
5. Use onehz as En in Traffic Light Controller.
6. Adding the seven segment display module and giving the timing signals (T4, T3,T2,T1) to it.
7. Implementing the design by pressing “Configure target device”.
8. Programming it to the FPGA.
9. Observing the traffic light signals and verifying it by performing a simple experimental design.

3. Problems Faced

At first, we tried using exclusive modulo counters to get the timing. But, we got clock gating errors.
Hence, we updated the timings in the always block and output the timing signals from the tlc module.

4. Tasks and Snapshots

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 clklhz);
reg[26:0] counter;
// To generate a signal with a frequency of 1hz from 100Mhz, the counter should count 100 000 000 cycles.
assign clklhz = (counter == 99999999);
always@(posedge clk,posedge reset)
if (reset || clklhz )
   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.

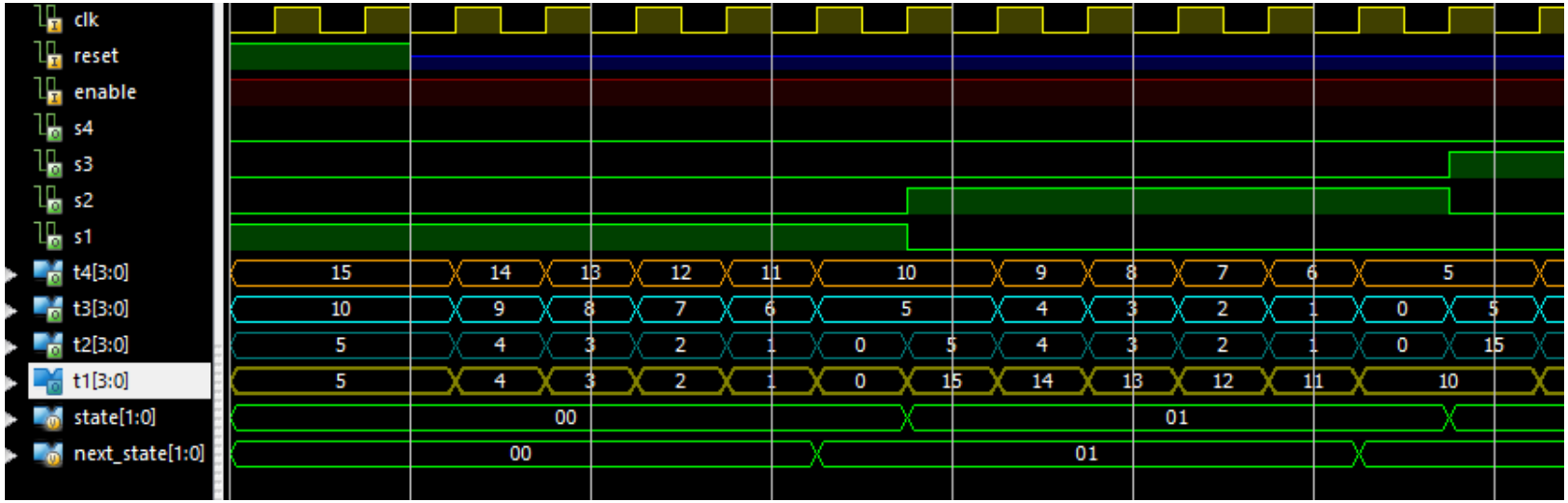

Figure 3. Simulation of Traffic Light Controller for STATE 1 and STATE 2.

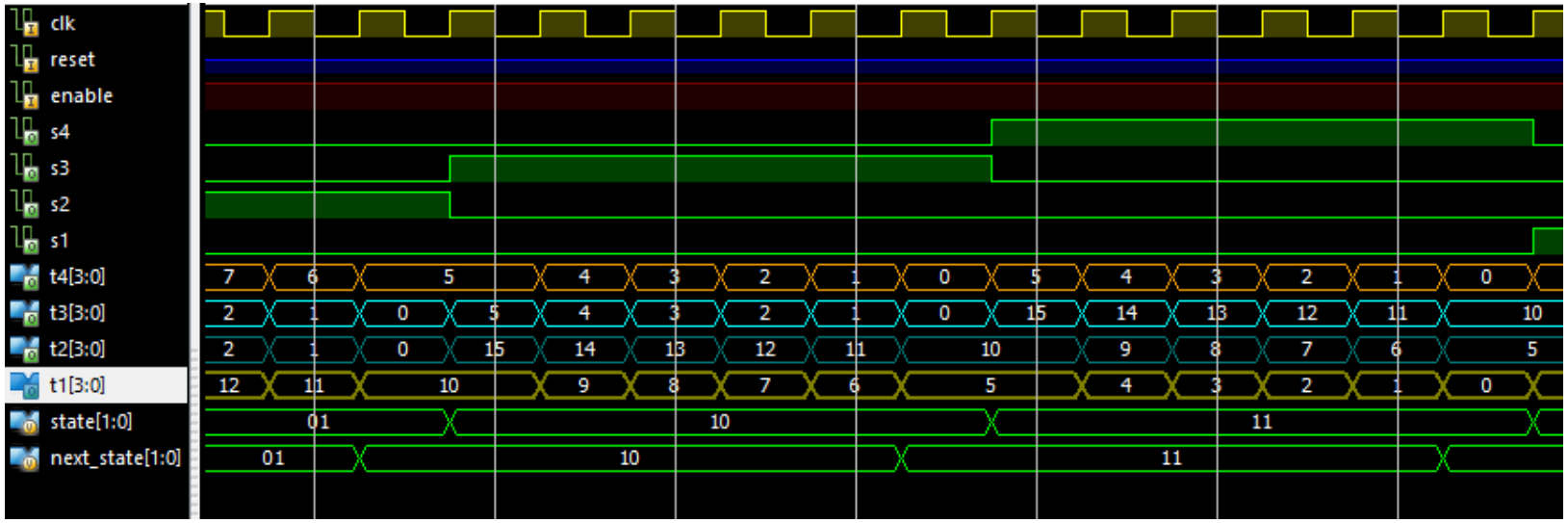

Figure 4. Simulation of Traffic Light Controller for STATE 3 and STATE 4.

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

5. Conclusion

In this Mini Project, we further learned about designing circuits using Verilog, mainly behavioral modelling of circuits. Additionally, we further developed our understanding on how to design and model finite state machines. We also learned about how to effectively work as a team and accordingly divide a bigger problem into smaller tasks. Finally, we learned to design a timed 4-road traffic light controller. In order to achieve the desired aim of this project, it project was divided into four main tasks:

1. State table for TLC
2. Verilog module for a one hertz pulse generator
3. Verilog module for TLC and simulation
4. Verilog module for TLC test

The results of all the tasks were consistent with the expected outcomes.