INTELLIGENT TRAFFIC LIGHT CONTROLLER (ITLC)

V.Purna Chandra Reddy ,Asst. professor
Dept of ECE,VVIT Nambur,522508
A.P ,India

V.Ravindra Reddy,Asst.Professor
Dept of ECE,VVIT Nambur,522508
A.P ,India

***Abstract*— Because of the increasing density of the traffic flow in urban areas there is a need for optimal performance of traffic lights. Traffic lights are generally installed to ensure safety, decrease the average time of proceeding through the intersection, increase the capacity of multileg intersections, and improve quality of service. An Intelligent traffic light controller (ITLC) has been designed, which controls the traffic on a busy intersection as per density of traffic, sensed by sensors (video image processors-VIPs or vehicle counters etc) placed on the roads on each approach. This sensed signal is used to determine the duty cycle for the green signal allowing larger period for the roads with larger number of vehicles. The behavioral code will be written in Verilog hardware description language (HDL). The behavioral code will be pre-simulated for functional verification and then gate level net list extracted after synthesis of the circuit using design compiler. During emergency the controller can be switched off for free flow of traffic.**

I. INTRODUCTION

In our current world traffic has become an increasingly substantial part of society. With people becoming even more mobile, traffic jams are becoming a more and more common sight, especially in large urban areas. This, of course, leads to all kinds of unwanted side effects like people arriving too late at their destination, economical damage and environmental pollution. Traffic in urban areas is mainly regularized by means of traffic lights, which can make for unnecessary long waiting times for cars when not efficiently configured. This inefficient configuration is unfortunately still the case in a lot of urban areas; most of the traffic lights are based on a ‘fixed cycle' protocol, which basically means that the lights will be turned on green for a fixed amount of time and consecutively on red for a fixed amount of time.
Traffic Light Timing Control Methods: Traffic light timing using the incoming traffic conditions can be accomplished in different ways. In the pre-timed mode, each phase period and cycle duration is determined based on some predetermined values by some statistics. In traffic prediction (Actuated Signals), the Future mode is estimated and decided by sensors based on the measured situation. In the pattern matching method, the information obtained by the sensors is adapted by a set of mathematical operations with the existing information, the closest pattern to the current conditions is then selected and appropriate time values are applied to the traffic lights accordingly.

One may go through an intersection on green and must stop on red. Yellow is displayed right before the light turns red. In addition, this controller gives drivers waiting on a red light a warning just before it turns green by displaying both red and yellow lights. Sensors play an important role in guiding the controller to switch onto different states depending on the traffic. Pedestrians are also considered to make the system more adaptable.

II. DESCRIPTION

A. *FUNCTIONAL SPECIFICATIONS:*

The schematic is shown below. Here we are considering 2-lane roads meeting at a junction namely s1, s2, s3, s4 and their corresponding sensors m1, m2, m3, and m4 respectively.
The different types of scenarios are as follows. Each road has 2 traffic lights, one for straight traffic and other for side traffic and they are assigned as S1, SLR1, S2, SLR2, S3, SLR3, S4, and SLR4 for each of the roads S1, S2, S3, and S4 respectively. P1,P2 traffic lights meant for pedestrians.

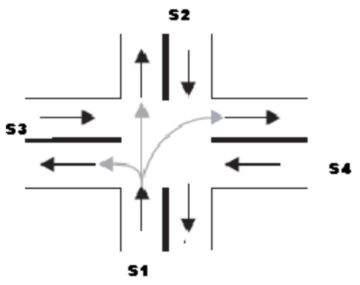
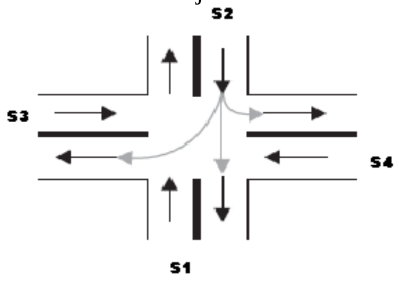
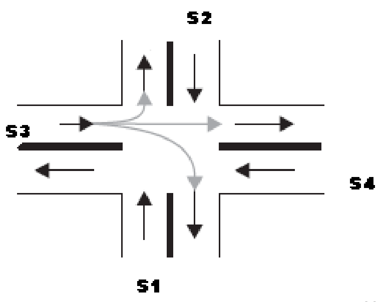


Figure shows a possible scenario whereby the traffic lights of set ***S1***and ***SLR1*** are ***GREEN*** while all others are ***RED***.
Here the shaded arrows indicate the allowed traffic that is the traffic given green; here the case is for S1.

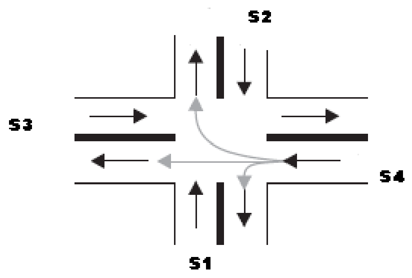
And the other scenarios at the junction are shown below.



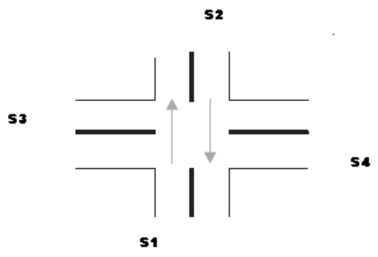
Here S2 and SLR2 is given green that is the traffic of S2 is allowed.



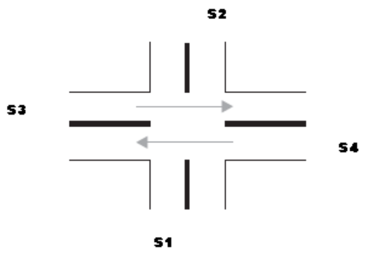
Here S3 and SLR3 is given green that is the traffic of S3 is allowed



Here S4 and SLR4 is given green that is the traffic of S4 is allowed



Here the straight traffics of S1 and S2 are allowed that is S1, S2 are given green and pedestrians(P1) along these directions are allowed.



Here the straight traffics of S3and S4 are allowed that is S3, S4 are given green and pedestrians (P2)along these directions are allowed.

***APPROACH**:*

The above scenarios are described by the state machine approach where the controller takes actions based on the state transitions in the diagram as explained below. a state diagram is drawn to reflect all the possible conditions or situations that may happen. The traffic light state machine controller must be able to handle all of the possible conditions.

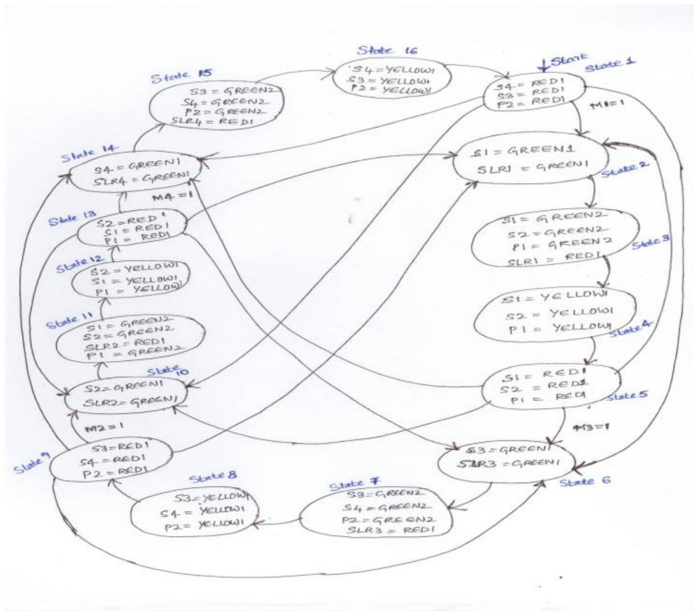


FIG . State diagram for traffic light state machine controller

We are taking four timers corresponding to different possibilities of the output.

g1timer and g2timers are for GREEN traffic light.

r1timer is for RED traffic light

y1timer,y2timer is for YELLOW traffic light

Here the need for two green timers arises due to the thing that the straight and side traffic is allowed for different periods that is for example in the case of road S1 initially both straight and side traffic are given green but after a period ,S1 side traffic is given red where as straight traffic is still green and this applies for other sides also.

Explanation:

The state diagram as shown above is explained as follows initially assume that all the states are at red.

I)Then the controller examines all the sensor inputs and the next state is achieved depending upon the active sensor that is the sensor which is 1. (M1=1 orM2=1 or M3=1 or M4=1)If M1=1 next state is state 2 where the S1 straight and side traffic is allowed. For the other sensors their corresponding traffics are allowed. As the default case we ask to go for state2. At state 2, S1 and SLR1 are Green for a period of g1timer.

II) After g1timer changes its state from 0 to 1, that is when the timer indicates the controller to change the state, the state comes to state 3 where S1 is still at green but SLR1 comes to red and S2 comes to green. Here pedestrians are also allowed, that is P1 comes to green.

III) Here for a while, when the g2timer is exhausted then the coming state is state4 where S1, S2 and P1 are yellow. After ytimer is exhausted the state changes to state5 where these comes to red.

IV) Here again the sensors are examined and the states are achieved by the active sensor, assigning state 6 as active M3 state and also the default one. Then the straight and side traffic is allowed and the next states are repeated.

V) At every red state the controller looks for an active sensor and the corresponding traffic is allowed. Thus each of the traffic of S1, S2, S3, S4 are allowed by their sensors or as the default cases and when one sensor state is achieved the next state is its side traffic is given red allowing opposite straight traffics with pedestrians. Then these comes to yellow and then to red.

If reset is active then all the lights come to state1 i.e. all RED state .The emergency (emer) switch is in active then all the lights come to yellow state and they will keep on blinking until emergency (emer) goes into inactive.

III.CODE AND SIMULATION RESULTS

INTELLIGENT TRAFFIC LIGHT CONTROLLER PROGRAM

1.Synthesizable Verilog Code for Traffic Light State Machine Controller

```
Module
trlc(g1timer,g2timer,r1timer,y1timer,y2timer,clc,res,M1,M2,
M3,M4,S1,S2, S3,S4,SLR1,SLR2,SLR3,SLR4,P1,P2,emer);
//M1,M2,M3,M4 are sensors
//S1,S2,S3,S4 are traffic lights for signaling opposite roads
//SLR1,SLR2,SLR3,SLR4 are traffic lights for signaling side
roads
//P1,P2 are traffic lights for signaling pedestrians

input g1timer,g2timer,r1timer,y1timer,y2timer;
//g1timer and g2timers are for GREEN traffic light.
//r1timer is for RED traffic light
//y1timer is for RED traffic light

input clc,res,M1,M2,M3,M4,emer;
//clc is the clock signal
//res is the reset signal
//emer is the emergency signal

output [4:0]S1,S2,S3,S4,SLR1,SLR2,SLR3,SLR4,P1,P2;
reg [4:0] pre_state,next_state;
reg i;
parameter [4:0]
GREEN1=1,GREEN2=2,RED1=7,YELLOW1=4,YELLOW2
=5,OFF=0;
parameter [4:0] STATE1=1,
                STATE2=2,
                STATE3=3,
                STATE4=4,
                STATE5=5,
                STATE6=6,
                STATE7=7,
                STATE8=8,
                STATE9=9,
                STATE10=10,
                STATE11=11,
                STATE12=12,
                STATE13=13,
                STATE14=14,
                STATE15=15,
                STATE16=16,
                STATE17=17,
                STATE18=18;

always@(g1timer or g2timer or r1timer or y1timer or M1 or
M2 or M3 or y2timer or M4 or emer )
begin
 case(pre_state)

STATE1:  begin
      //Initial state
     //Traffic lights are in RED state
       if(r1timer  & M1==1)
       next_state = STATE2;//If sensor M1 is ON//
       else if(r1timer==1 & ~M1 & M3)
       next_state = STATE6;//If sensor M3 is ON//
      else if(r1timer==1  & ~M1 & ~M3 & M2)
       next_state = STATE10;//If sensor M2 is ON//
      else if(r1timer==1  & ~M1 & ~M2 & ~M3 & M4)
       next_state = STATE14;//If sensor M4 is ON//
       else
       next_state = STATE1;
    end
```

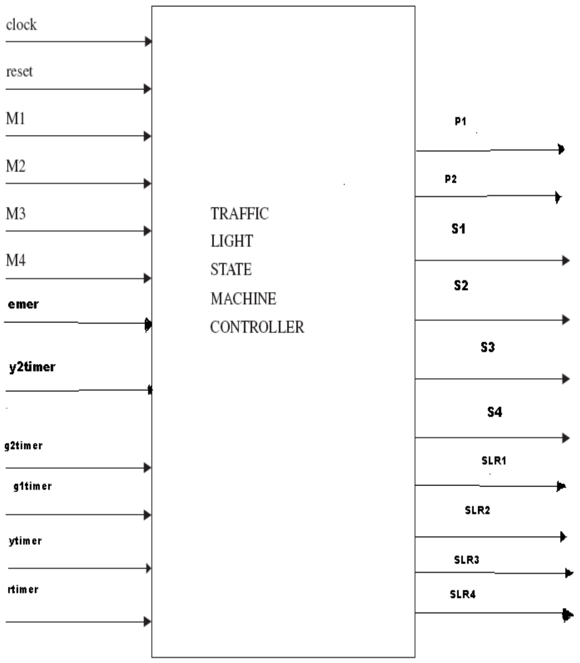


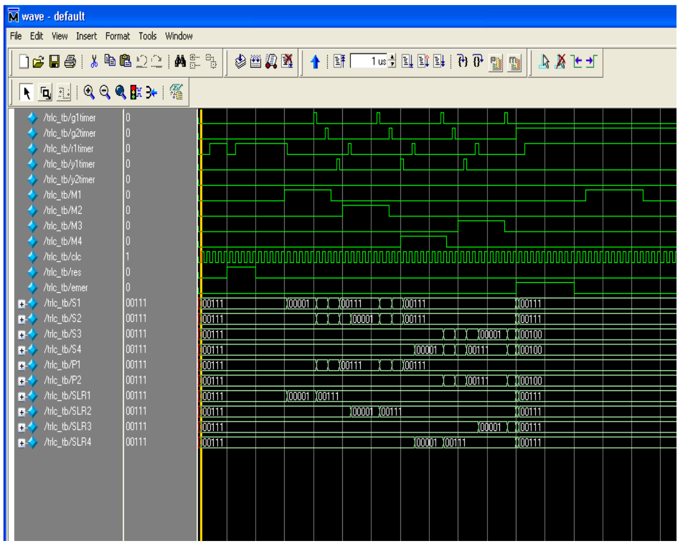
Fig. Diagram shows interface signals for traffic light controller

```
STATE2:  begin
 // Traffic lights S1,SLR1  are in GREEN1 state
       if(g1timer )
       next_state = STATE3;
       else
       next_state=STATE2;
```

```
        end

STATE3:  begin
       //Traffic lights S1,S2,P1 are in GREEN2 state
      //Traffic light SLR1 is in RED1 state
      if(g2timer)
         next_state = STATE4;
         else
         next_state=STATE3;
    end

STATE4:begin
      //Traffic lights S1,S2,P1 are in YELLOW state
       if(y1timer )
         next_state = STATE5;
         else
         next_state=STATE4;
       end

STATE5: begin
       //Traffic lights are in RED state

         if(r1timer==1 & M3==1)
         next_state = STATE6;//If sensor M3 is ON//
         else if(r1timer==1& ~M3 & M2)
         next_state = STATE10;//If sensor M2 is ON//
         else if(r1timer==1 &    ~M2 & ~M3 & M4)
         next_state=STATE14;//If sensor M4 is ON//
         else if(r1timer==1 &   ~M2 & ~M3 & ~M4& M1)
         next_state=STATE2;//If sensor M1 is ON//
         else
         next_state=STATE5;
       end

STATE6:   begin
      // Traffic lights S3,SLR3  are in GREEN1 state
         if(g1timer )
         next_state=STATE7;
         else
         next_state=STATE6;
       end

STATE7:   begin
        //Traffic lights S3,S4,P2 are in GREEN2 state
       //Traffic light SLR3 is in RED1 state

         if(g2timer)
         next_state=STATE8;
         else
         next_state=STATE7;
       end

STATE8:   begin
      //Traffic lights S3,S4,P2 are in YELLOW state

        if(y1timer)
         next_state=STATE9;
         else
         next_state=STATE8;
       end

STATE9: begin
         if(r1timer==1  & M2==1)
          next_state = STATE10;//If sensor M2 is ON//
          else if(r1timer==1&  ~M2 & M4)
          next_state = STATE14;//If sensor M4 is ON//
          else if(r1timer==1 &   ~M2 & ~M4 & M1)
          next_state=STATE2;//If sensor M1 is ON//
          else if(r1timer==1 &  ~M2 & ~M1 & ~M4& M3)
          next_state=STATE6;//If sensor M3 is ON//
          else
          next_state=STATE9;
       end

STATE10:begin
         // Traffic lights S2,SLR2  are in GREEN1 state
         if(g1timer)
          next_state=STATE11;
          else
          next_state=STATE10;
       end



STATE11:  begin
        //Traffic lights S1,S2,P1 are in GREEN2 state
        //Traffic light SLR2 is in RED1 state
         if( g2timer)
          next_state=STATE12;
          else
          next_state=STATE11;
       end

STATE12:   begin
         //Traffic lights S1,S2,P1 are in YELLOW state
       if(y1timer )
          next_state=STATE13;
          else
          next_state=STATE12;
       end

STATE13:begin
 if(r1timer==1 & M4==1)
          next_state = STATE14;//If sensor M4 is ON//
          else if(r1timer==1&  ~M4 & M1)
          next_state = STATE2;//If sensor M1 is ON//
          else if(r1timer==1 &   ~M4 & ~M1 & M3)
          next_state=STATE6;//If sensor M3 is ON//
          else if(r1timer==1 & ~M1 & ~M3 & ~M4& M2)
          next_state=STATE10;//If sensor M2 is ON//
          else
          next_state=STATE13;
       end
```

```
STATE14: begin
     // Traffic lights S4,SLR4  are in GREEN1 state
        if(g1timer )
           next_state=STATE15;
           else
           next_state=STATE14;
       end

STATE15:begin
    //Traffic lights S4,S3,P2 are in GREEN2 state
     //Traffic light SLR4 is in RED1 state
    if(g2timer )
           next_state=STATE16;
           else
           next_state=STATE15;
       end

STATE16:begin
         //Traffic lights S3,S4,P2 are in YELLOW state
    if(y1timer)
           next_state=STATE1;
           else
           next_state=STATE16;
       end



STATE17:begin
    //All traffic lights are in blinking Yellow state
      if(y2timer)
      next_state=STATE18;
      else
      next_state=STATE17;
      end

STATE18:begin
    //When emergency case is present
      if(emer)
      next_state=STATE17;
      else
      next_state=STATE1;
      end

default:
       next_state=STATE1;
  endcase
end
always@(posedge res or posedge clc)
begin
   if(res)
   pre_state<=STATE1;
   else
   pre_state<=next_state;
end
always@(posedge emer)
begin
   if(emer)
   pre_state<=STATE17;
   else
   pre_state<= STATE1;
end

assign S1=(pre_state== STATE2 )? GREEN1:
(pre_state== STATE3)? GREEN2:(pre_state== STATE11)?
GREEN2:
(pre_state==STATE4 )? YELLOW1 :(pre_state==STATE12
)? YELLOW1 :(pre_state== STATE17
)?YELLOW2:(pre_state== STATE18 )?OFF:RED1;

assign S2= (pre_state== STATE10)? GREEN1:
(pre_state==STATE3)? GREEN2:(pre_state==STATE11)?
GREEN2:
(pre_state==STATE12)? YELLOW1 :(pre_state==STATE4)?
YELLOW1 :(pre_state== STATE17
)?YELLOW2:(pre_state== STATE18 )?OFF:RED1;

assign S3=(pre_state== STATE6 )? GREEN1:
(pre_state== (STATE15))? GREEN2:(pre_state==
(STATE7))? GREEN2:
(pre_state==(STATE8))?
YELLOW1:(pre_state==(STATE16))?
YELLOW1:(pre_state== STATE17
)?YELLOW2:(pre_state== STATE18 )?OFF:RED1;

assign S4=(pre_state== (STATE14))? GREEN1:
(pre_state== (STATE7))? GREEN2:(pre_state==
(STATE15))? GREEN2:
(pre_state==(STATE8))?
YELLOW1:(pre_state==(STATE16))?
YELLOW1:(pre_state== STATE17
)?YELLOW2:(pre_state== STATE18 )?OFF:RED1;

assign P1=(pre_state== STATE3) ? GREEN2:
(pre_state== STATE11) ? GREEN2: (pre_state==STATE4) ?
YELLOW1:
(pre_state==STATE12) ? YELLOW1:RED1;

assign P2=(pre_state== STATE7) ? GREEN2:
(pre_state== STATE15) ? GREEN2: (pre_state==STATE8) ?
YELLOW1:
(pre_state==STATE16) ? YELLOW1:RED1;
assign SLR1=(pre_state == STATE2)? GREEN1:(pre_state==
STATE17 )?YELLOW2:(pre_state== STATE18
)?OFF:RED1;
assign SLR2=(pre_state == STATE10)?
GREEN1:(pre_state== STATE17 )?YELLOW2:(pre_state==
STATE18 )?OFF:RED1;
assign SLR3=(pre_state == STATE6)? GREEN1:(pre_state==
STATE17 )?YELLOW2:(pre_state== STATE18
)?OFF:RED1;
assign SLR4=(pre_state == STATE14)?
GREEN1:(pre_state== STATE17 )?YELLOW2:(pre_state==
STATE18 )?OFF:RED1;
endmodule
```

Simulation Result:

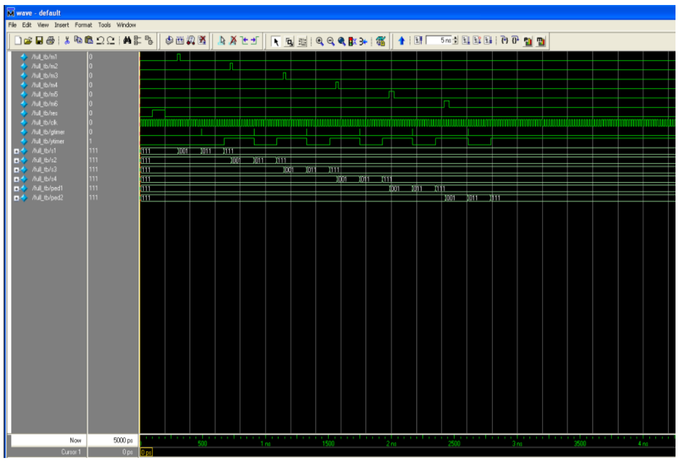


The above projects implemented have some constraints. It cannot support traffic going left and right when straight traffic is ‘on’. The preference for pedestrians is given in above code. Inorder to ease the traffic, new sensors are introduced and alleviate the problem. The code for the project is shown below.

2. Synthesizable Verilog Code for Traffic Light State Machine Controller

```
module basic(m,res,clk,gtimer,ytimer,s);
input m,res,clk,gtimer,ytimer;
output [2:0] s;
reg [3:0] pre_state,next_state;
parameter [2:0] red=7,idle=5,
                green=1,
                yellow=3;
   always @ (gtimer,ytimer,m)
   begin
      case(pre_state)
         green:
           begin
           if(gtimer)
           next_state=yellow;
          else
           next_state=green;
      end
      yellow:
      begin
         if(ytimer)
         next_state=idle;
         else
         next_state=yellow;
      end
        idle:
           begin
           if(m)
           next_state=green;
           else
           next_state=idle;
         end
         default:
         next_state=idle;
   endcase
end
   always @ (posedge clk or posedge res)
   begin
   if(res)
   pre_state<=idle;
   else
   pre_state<=next_state;
end
assign s=(pre_state== green)?green:(pre_state==
yellow)?yellow:red;
endmodule

module
full(m1,m2,m3,m4,m5,m6,res,clk,gtimer,ytimer,s1,s2,s3,s4,pe
d1,ped2);
input  m1,m2,m3,m4,m5,m6,res,clk,gtimer,ytimer;
output  [2:0]s1,s2,s3,s4,ped1,ped2;
basic
one(.m(m1),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer),.
s(s1));
basic
two(.m(m2),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer),.
s(s2));
basic
three(.m(m3),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer)
,.s(s3));
basic
four(.m(m4),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer),
.s(s4));
basic
five(.m(m5),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer),
.s(ped1));
basic
six(.m(m6),.res(res),.clk(clk),.gtimer(gtimer),.ytimer(ytimer),.
s(ped2));
endmodule
```

Simulation Result:



ADVANTAGES:

- Since the waiting time of vehicles for the lights to change is optimal, the emission of carbon monoxide from the vehicles is reduced. This will give a positive effect to the green house effect towards the environment.
- It also save the motorists time and reduces their frustration while waiting for the lights to change since it helps reducing congestion in the intersections.
- There is no interference between the sensor rays and there is no redundant signal triggering.

IV.CONCLUSION

This project outlines the design, simulation and implementation of the ITLC, which behaves adaptively to the density of the vehicles on a busy intersection, as sensed by the sensors. Emphasis has been laid on the logic synthesis of gate-level circuit for adaptive behavior of the ITLC and simulation for the desired performance. The gate net list can be utilized for layout design and FPGA design. The ITLC behavior has been coded in VERILOG.

The behavioral code analyzed by the VERILOG analyzer and pre-simulated by VERILOG system simulator (Model Sim), to verify its functionality. The design has been simulated with a test bench comprising a set of stimuli. Synopsis design compiler was then used to convert this behavior into the gate level net-list. After getting the optimized gate level netlist, it was post simulated with the same test bench to test for the required behavior. The code has to be extended for two way traffic.

REFERENCES

[1] FPGA Design Tutorial, Copyright © 1-CORE Technologies, 2004-2009. [Online]. Available: http://www.1-core.com/library/digital/fpga-design-tutorial/qui ck-startguide.html

[2] P. P. Chu, "FPGA Prototyping by VHDL Examples-Xilinx Spartan-3 Version," John Wiley and Sons, 2008

[3] ISE 9.1 In-Depth Tutorial, Xilinx Inc. Copyright 1995-2007. [Online].

[4] Design of FPGA-Based Traffic Light Controller System M. F. M. Sabri, M. H. Husin, W. A. W. Z. Abidin, K. M. Tay, H. M. Basri, Dept. of Electronic Engineering, Faculty of Engineering, University Malaysia Sarawak, Sarawak, Malaysia. 978-1-4244-8728-8/11

[5] Design and Implementation of FPGA-Based Adaptive Dynamic Traffic Light Controller Shwetank Singh, Shailendra C. Badwaik ,Dept. of Electronics &Telecommunication ,Sinhgad College of Engineering Pune,India

[6] S Singaria, Non-member,Dr R P Agarwal, Fellow,Dr S S Jain, Fellow,Dr M Parida, Member, "Hardware Implementation of an Adaptive Traffic Light Controller for a Busy Intersection", IE (I) Journal−ET,1-24(2006).

[7] G. "The Verilog Hardware Description Language, THIRD EDITION", Donald E. Thomas, Carnegie Mellon University &Philip R. Moorby, Avid Technology, Inc.