TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

SMART TRAFFIC LIGHTS

SUBMITTED BY:

AASHUTOSH POUDEL 072BCT502

DINESH BHATTARAI 072BCT512

JEEVAN THAPA 072BCT514

RUPESH SHRESTHA 072BCT530

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING,
IOE, PULCHOWK CAMPUS

9th August, 2018

Introduction

With increasing urbanization and increase in traffic flow, traffic jams and congestion are a serious problem to be solved. Therefore, the traffic control of city's intersections should be adjusted in time based on the structure of the intersections and changes of traffic flow. It is particularly important to choose traffic light phase and duration scientifically and rationally in the process to dredge the traffic jams.

Various techniques can be used for such implementation of a traffic light system. Artificial Intelligence and Machine Learning based implementations can provide better control by learning through data. Similarly, a naive solution will be just a time based solution. However, we can improve upon this by adding some sensors on such naive solution to make it more intelligent and useful.

Design

We have chosen a place where there are four paths from the center, so there are four units of traffic light system that work by coordinating with each other. Each unit consists of RGY lights which are controlled by input pins to the light system given by a 8051 microcontroller.

The basic design of the system is a Finite State Machine (FSM) which changes its state based on inputs, mainly sensor inputs and clock signals. Generally the lights work in circular order. Generally speaking, the total durations of green, left-hand turn and yellow light equal to the duration of red light. For the convenience of programming and debugging, the transition control can be achieved by finite state machine.

Analysis of Running State

The number of actual running state is determined by the input pins of phase selection. The controller receives new phase setting when the reset signal is valid (state S0). When number 2 is input by the pins of phase selection, the order of states changing is S1, S3, S4, S6, and then turn back to S1. So 2-phase control can be realized. When number 3 is input, the order of states changing is S1, S2, S3, S4, S6, and then turn back to S1. 3-phase control can be realized by increasing the left-hand turn phase on master branches on the basis of 2-phase. When number 4 is input, the order of states changing is S1, S2, S3, S4, S5, S6, and then turn back to S1. 4-phase control can be realized by increasing the left-hand turn phase on slave branches on the basis of 3-phase. The translation chart of states is shown in Figure 2.

Structure of the System

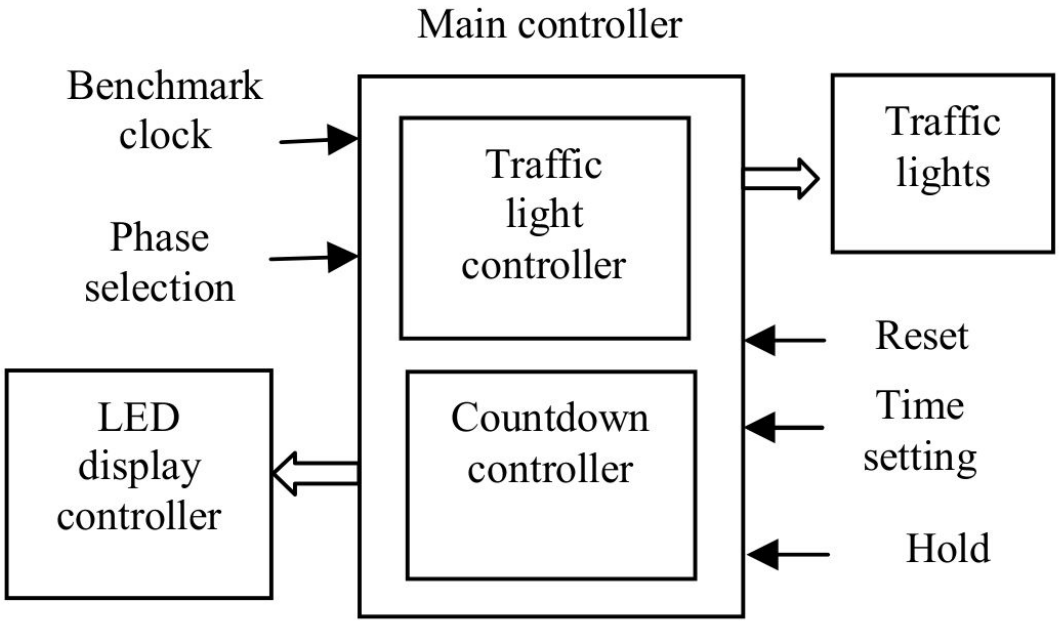


Figure: Structure of the system

The main functions components of the system are traffic lights controller, countdown controller and LED display controller. The chart of system structure is shown in Figure 3. Main controller is the core of the system, all the signals of traffic lights both on master and slave branches are generated by the main controller. According to different value of phase selection, all traffic lights run as the modes shown in Figure 1. Meanwhile, the countdown numbers are provided and sent to LED display controller. Benchmark clock of the system is provided by the external circuit. All the lights both on the master and slave branches are timed for the S unit. Both the LEDs on master and slave branches display the remaining running time in countdown form. Usually, the duration of red light is the longest, the sum time of green, left-hand turn and yellow lights equals to the red light'.

Program Design and Simulation

VHDL Program

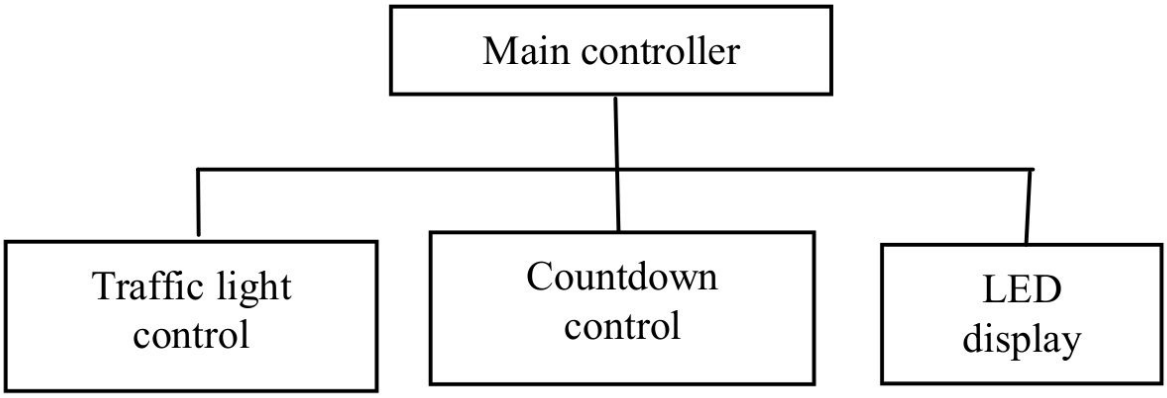


Fig: Structure of Hierarchical Design

Simulation Results

The VHDL program simulate after being compiled and debugged. The aim of simulation is to fully verify whether the designing goal can be achieved or not. Two points need to be considered when to create the simulation file. One is to verify the validity of phase selection signal, time setting signal, hold signal and reset signal, the other is to verify whether realize variable multi-phase control thought when different phase value is set. Wave file is created on the basis of the two points mentioned above. Simulation wave charts are shown as figure 5, 6, and 7. In the simulation file, the time of red light is set 20s, the time of green light is set 10s or 15s when the phase is different, the time of yellow and left-hand turn light are set 5s. As shown in these charts, all lights extinguish and all the LEDs show 0 both in master and slave branches when the reset signal is valid. All the LEDs remain unchanged and all the red lights work both in master and slave branches when the hold signal is effective. The running order of 2-phase, 3-phase and 4- phase can be realized as Figure 2 according to the value of phase selection signals.

Conclusion

The variable multi-phase (2-phase, 3-phase and 4-phase) intelligent traffic light controller can be realized by establishing system model, programming with VHDL language, and simulating with Quartus . This design thought can reduce the design cycle, improve the reliability and flexibility.

The controller can be made into embedded circuit board to satisfy the need to update the valve of
control phase according to the actual traffic flow in city's traffic intersections.

Appendix

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TLC is
    Port (
           Norvic : out  STD_LOGIC_VECTOR (2 downto 0);
           Maitighar : out  STD_LOGIC_VECTOR (5 downto 0);
           Kupandole : out  STD_LOGIC_VECTOR (5 downto 0);
           Tripureshwor : out  STD_LOGIC_VECTOR (5 downto 0));
end TLC;

architecture Behavioral of TLC is
     component  trafficLight
          Port ( TRIGGER : in STD_LOGIC_VECTOR (1 downto 0);
               RYG: out  STD_LOGIC_VECTOR(2 downto 0));
     end component;
     signal state: std_logic_vector(13 downto 0):="01010101010101";

     signal prevstate: std_logic_vector(13 downto 0);
begin
     KunpandoleTLa: trafficLight Port map (
               TRIGGER => state(13 downto 12),
               RYG => Kupandole(2 downto 0)
     );
     KunpandoleTLb: trafficLight Port map (
               TRIGGER => state(11 downto 10),
               RYG => Kupandole(5 downto 3)
     );
     MaitigharTLa: trafficLight Port map (
               TRIGGER => state(9 downto 8),
               RYG => Maitighar(2 downto 0)
     );
     MaitigharTLb: trafficLight Port map (
               TRIGGER => state(7 downto 6),
               RYG => Maitighar(5 downto 3)
```

```
	);
	TripureshworTLa: trafficLight Port map (
			TRIGGER => state(5 downto 4),
			RYG => Tripureshwor(2 downto 0)
	);
	TripureshworTLb: trafficLight Port map (
			TRIGGER => state(3 downto 2),
			RYG => Tripureshwor(5 downto 3)
	);
	NorvicTLa: trafficLight Port map (
			TRIGGER => state(1 downto 0),
			RYG => Norvic(2 downto 0)
	);

	t : process
	begin

		prevstate <= state;
		wait for 0 ns;
		--- Kupandole Go
		if state = "01011000100101" or state = "01010101010101"
then
		state <= "01011000100101";
		wait for 200ns;

		--- Kupandole Stop
		elsif state = "
		state <= "10101001011010";
		wait for 200ns;

		--- Tripureshwor Go
		state <= "01011001100101";
		wait for 200ns;

		--- Tripureshwor Stop
		state <= "10101001011010";
		wait for 200ns;

		--- Maitighar Go
		state <= "01011001100101";
		wait for 200ns;

		--- Maitighar Stop
```

```
            state <= "10101001011010";
            wait for 200ns;

            --- Norvic
            state <= "10101001011010";
            wait for 200ns;
      end process;

      process
      signal finalstate: std_logic_vector(13 downto 0);
      begin
            finalstate <= state;
            wait for 0 ns;
            for i in 0 to 13
            loop
                  if state (i) /= prevstate (i) then
                        state(i) <= '1';
                  end if;
            end loop;
            wait for 20 ns;
            state <= finalstate;
            wait on state;
      end process;
end Behavioral;
```

Simulator Code for 8051 :

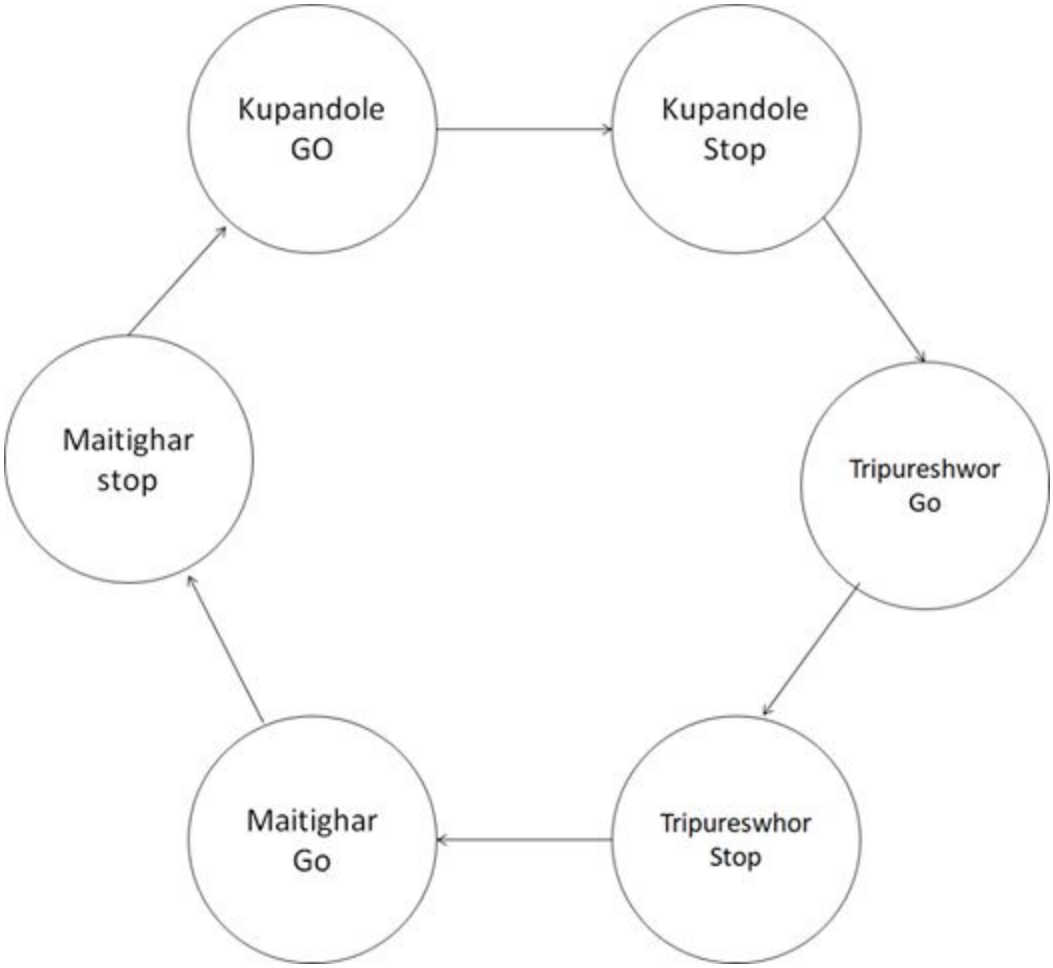
```
#include <reg51.h>
int frames[6][4] = {{0x24, 0x21, 0x21, 0x04},
                {0x09, 0x21, 0x0C, 0x01},
                  {0x21, 0x24, 0x09, 0x04},
                  {0x0C, 0x09, 0x21, 0x04},
                  {0x21, 0x21, 0x24, 0x04},
                  {0x21, 0x0C, 0x09, 0x04}};

void delay(int a){
      int i = 0;
      int j = 0;

      for ( i = 0; i< a ; i++){
            for (j = 0 ; j < 1000; j++){
                  };
      }
}
```

```
int transita(int a, int b) {
    int c, d, e, f, g;
    c = 0;
    if (a == b) {
        return a;
    } else {
        d = a % 8;
        e = b % 8;

        if (d == e)
            c = d;
        else
            c = 2;

        f = a / 8;
        g = b / 8;

        if (f == g)
            c = c + f * 8;
        else
            c =  c + 16;
    }
    return c;
}



void main(void)
{
     int i, j, k;
     for (i= 0;i<6;i++)
     {

           P2 = frames[i][0];
           P0 = frames[i][1];
           P1 = frames[i][2];
           P3 = frames[i][3];
           delay(500);
           if (i == 5 )
                k = 0;
           else
                k = i+1;
```

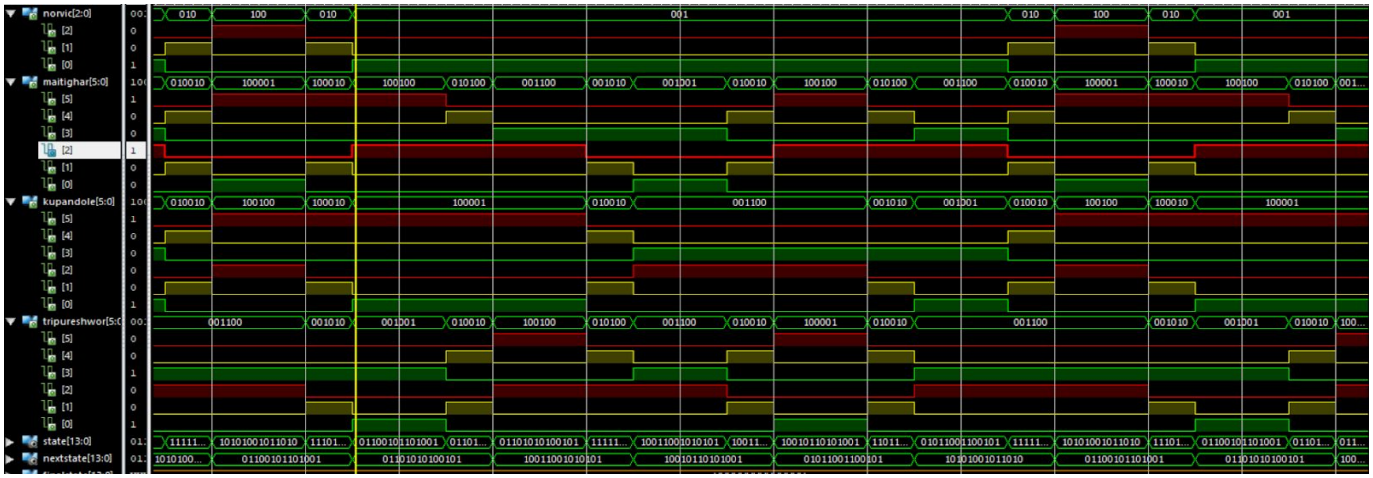
```
            P2 = transita(frames[i][0],frames[k][0]);
            P0 = transita(frames[i][1],frames[k][1]);
            P1 = transita(frames[i][2],frames[k][2]);
            P3 = transita(frames[i][3],frames[k][3]);
            delay (400);
      };
}
```

Map of Thapathali

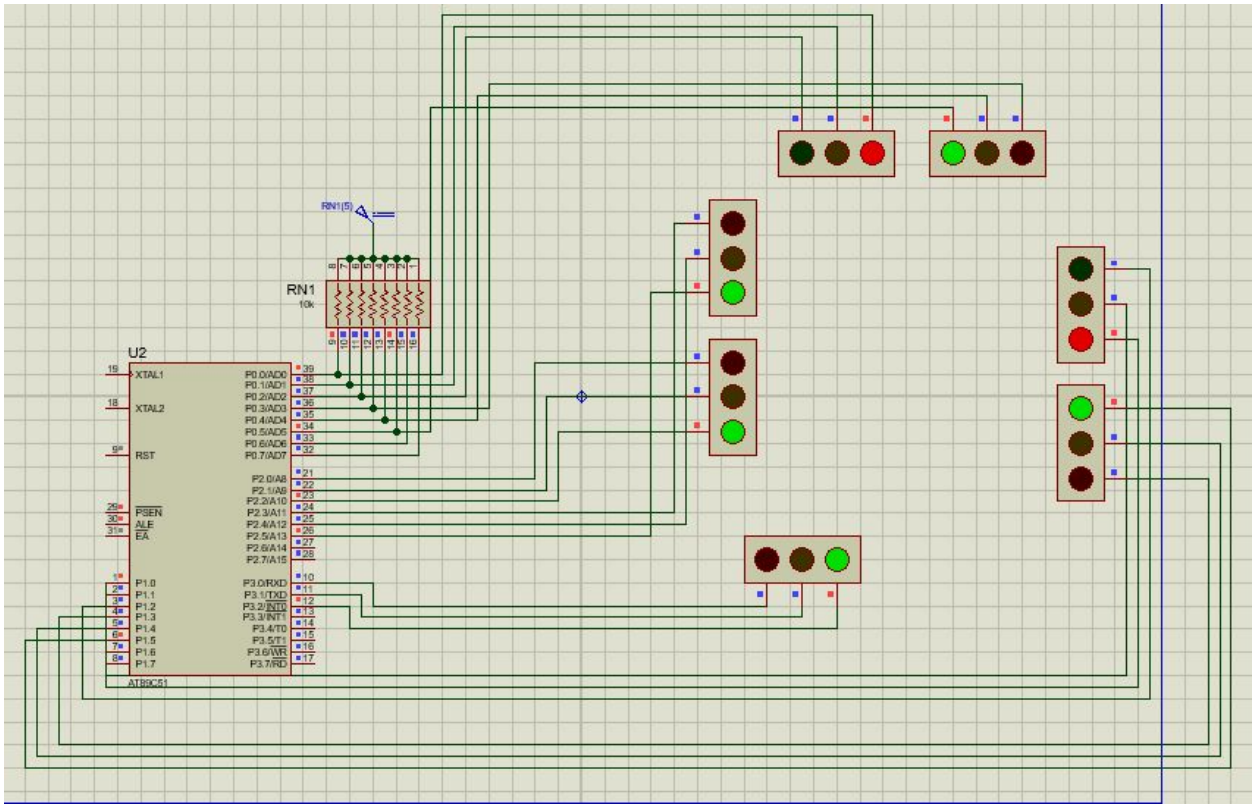
State Diagram



Timing Diagram



Simulation


RN1
10k
U2
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
AT89C51