Traffic Lights Controller

Design, Code and Simulation

By -

Abhishek Jaisingh [14114002]

A simple traffic light controller can be implemented by a state machine that has a state diagram such as the one shown in Figure. Its state progresses according the value of the timer used . When the value of timer reaches a specific value, the state of the system changes .

A set of 4 traffic lights have been designed for a 4 – way junction as shown.

The states are defined in terms of the output . Timer goes from 0 to 100 and then it is reset to 0. Each light is programmed to run as follows:

1. It is GREEN for 20 s.
2. Then it remains YELLOW for 5s.

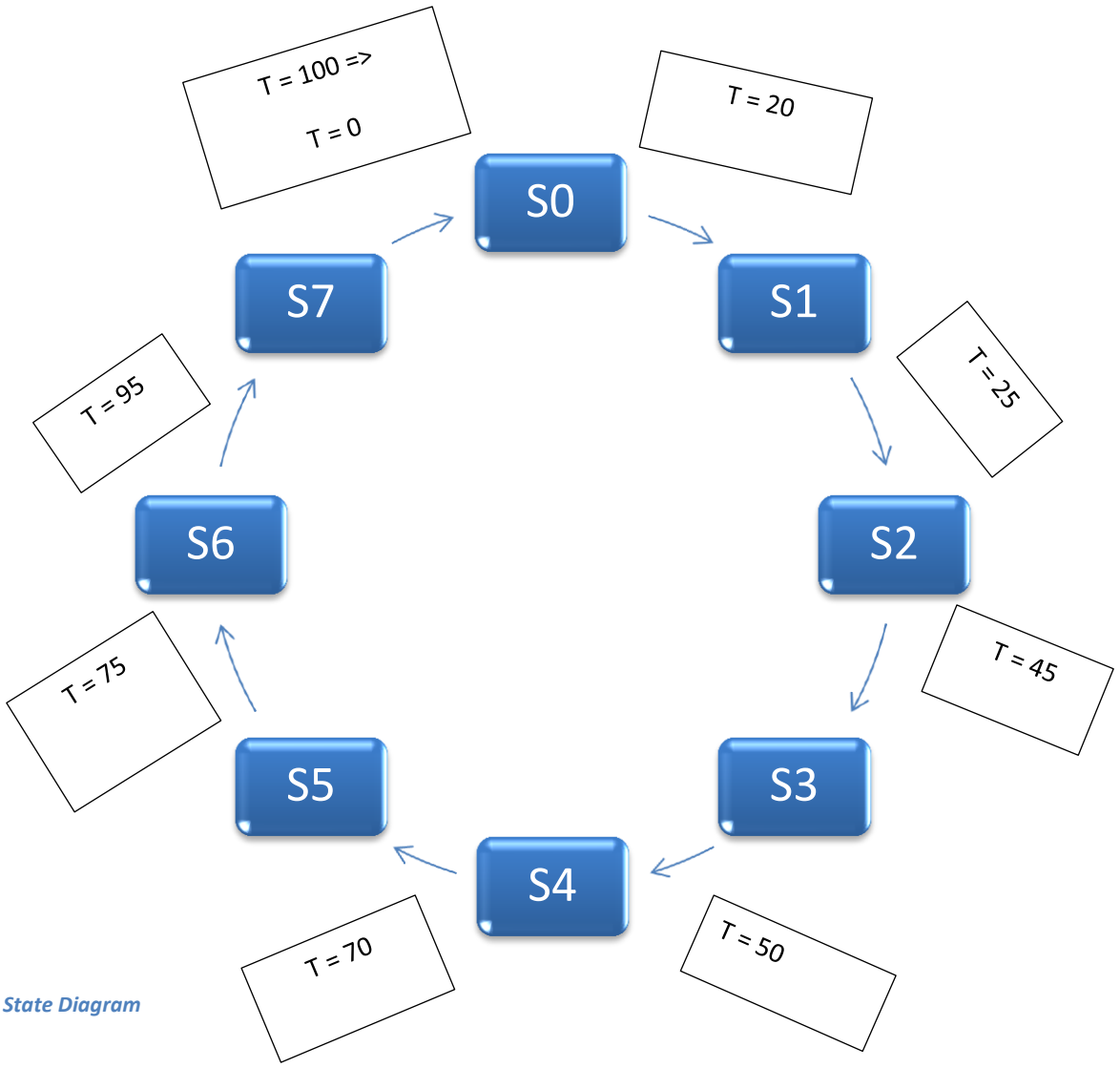
During this time all other lights remain RED.

This process is continued for all the lights successively.

Finite State Machine

(Moore Machine)

State Diagram



State Diagram

State Table

Time (input)	Current State	Next State	Output
0	S0	S0	1234
20	S0	S1	1234
25	S1	S2	1234
45	S2	S3	1234
50	S3	S4	1234
70	S4	S5	1234
75	S5	S6	1234
95	S6	S7	1234
100	S7	S0	1234

The color in output represents output light.

time = 0 after time becomes 100.

Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;

entity traffic is
	port(
		clk : in STD_LOGIC;
		clr : in STD_LOGIC;
		red1 : out STD_LOGIC;
		yellow1 : out STD_LOGIC;
		green1 : out STD_LOGIC;
		red2 : out STD_LOGIC;
		yellow2 : out STD_LOGIC;
		green2 : out STD_LOGIC;
		red3 : out STD_LOGIC;
		yellow3 : out STD_LOGIC;
		green3 : out STD_LOGIC;
		red4 : out STD_LOGIC;
		yellow4 : out STD_LOGIC;
		green4 : out STD_LOGIC
	    );
end entity traffic;

architecture trafficA of traffic is
type state_type is (s0, s1, s2, s3, s4, s5,s6,s7);
signal state : state_type := s0;
signal count : integer := 0;
		signal lights: std_logic_vector(11 downto 0);
begin
	STATEpro : process(state)
		begin
			case state is
				when s0 => lights <= "001100100100";
				when s1 => lights <= "010100100100";
				when s2 => lights <= "100001100100";
				when s3 => lights <= "100010100100";
```

```
                        when s4 => lights <= "100100001100";
                        when s5 => lights <= "100100010100";
                        when s6 => lights <= "100100100001";
                        when s7 => lights <= "100100100010";
                        when others => lights <= lights;
                    end case;
            end process;
    LT : process(clk)

        begin
            case count is
                    when 0 => state <= s0; count <= count + 1;
                    when 20 => state <= s1; count <= count + 1; -- 1st green ends
                    when 25 => state <= s2; count <= count + 1; -- 1st yellow ends
                    when 45 => state <= s3; count <= count + 1; -- 2nd green ends
                    when 50 => state <= s4; count <= count + 1; -- 2nd yellow ends
                    when 70 => state <= s5; count <= count + 1; -- 3rd green ends
                    when 75 => state <= s6; count <= count + 1; -- 3rd yellow ends
                    when 95 => state <= s7; count <= count + 1; -- 4th green ends
                    when 100 => count <= 0;-- 4th yellow ends
                    when others => count <= count + 1;
            end case;
            green4 <= lights(0);
            yellow4 <= lights(1);
            red4 <= lights(2);
            green3 <= lights(3);
            yellow3 <= lights(4);
            red3 <= lights(5);
            green2 <= lights(6);
            yellow2 <= lights(7);
            red2 <= lights(8);
            green1 <= lights(9);
            yellow1 <= lights(10);
            red1 <= lights(11);
        end process;
end architecture trafficA;
```

Input / Output Waveforms