TRAFFIC LIGHT CONTROL SYSTEM

Prepared by:

Salma Ihab Abdelmawgoud Ahmad Hamed (19P8794)

Noorhan Hatem Ibrahim Mohamed (19P5821)

Madonna Magdy Moussa (19P2671)

Ibrahim Ahmed Hassan (16P3062)

Noureldien Khaled Ahmed (18P5722)

Faculty of Engineering, Ain Shams University

CSE 312: Electronic Design Automation

Prof. Dr. Hasan A. Youness Alansary

January 14, 2022

Table of Contents

BACKGROUND DETAILS

- The traffic light system was modelled after the system designed in this video: https://youtu.be/DP62ogEZgkI?t=276
- There are 4 lanes: north, south, east, west.
- Outputs: 2 traffic lights for through passing and right turning cars, 2 traffic lights for left turning cars, 2 pedestrian lights for each lane.
- North and south, in both main and left-turning traffic lights, must display the same color. The same goes for the east and west lights and the pedestrian lights. Therefore, we only use 1 traffic light to display for north and south, and 1 light for east and west.
- The pedestrian light in a lane can only be green if the car light in that lane is red.
- There is an emergency button and a reset button. The emergency button precedes the reset button in priority; if both are on, traffic lights go into stx, an emergency state. The reset button is best used to start up the controller cycle and rescue the controller from stx
- Inputs: counter, emergency button, reset button, clk.

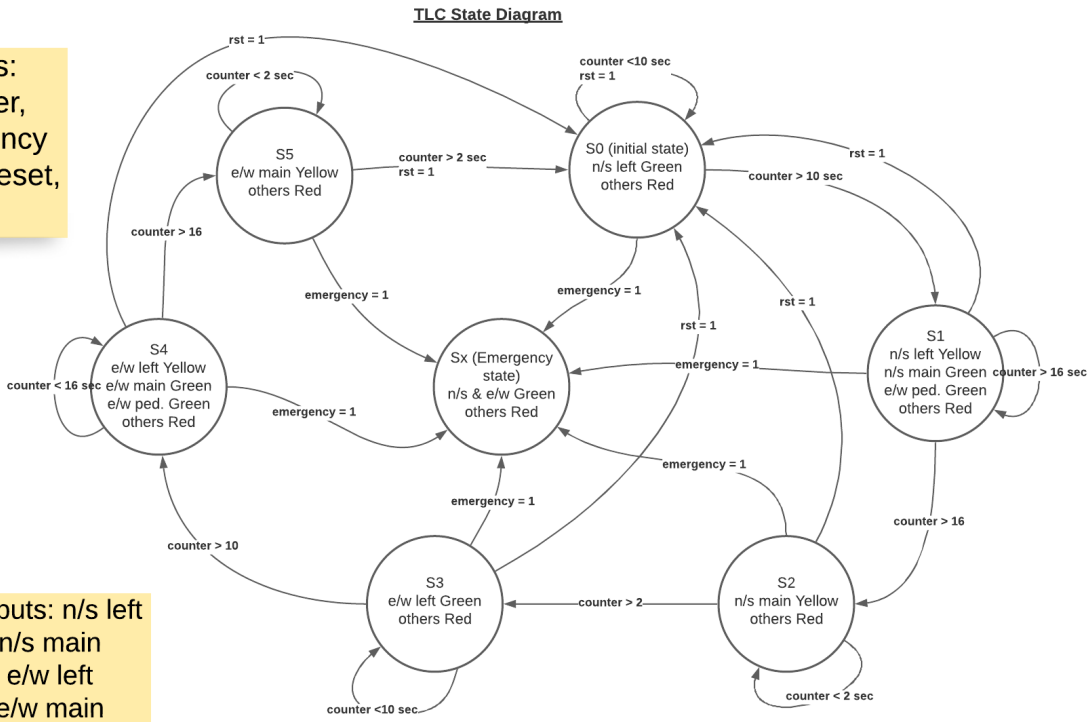
Summary of Changes Made to the Original 1-Street Traffic Control System:

1. Added two traffic lights for each street (left lane & main lane).
2. Added two intersecting streets (north/south intersects with east/west).
3. Emergency button and emergency state added.

Model of the Intersection:

STATE DIAGRAM

inputs: counter, emergency button, reset, clk



outputs: n/s left
n/s main
e/w left
e/w main
n/s pedestrian
e/w pedestrian

CODE

```
--CODE

LIBRARY IEEE;

USE  IEEE.STD_LOGIC_1164.all;

USE  IEEE.STD_LOGIC_ARITH.all;      -- needed for arithmetic increment

USE  IEEE.STD_LOGIC_UNSIGNED.all;

use ieee.numeric_std.all;


entity traffic is

Port (

north_tl:out STD_LOGIC_Vector (2 downto 0); -- right turn and through
passing traffic lights

east_tl:out STD_LOGIC_Vector (2 downto 0);


north_left:out STD_LOGIC_Vector (2 downto 0); -- left turn traffic
lights

east_left:out STD_LOGIC_Vector (2 downto 0);


north_ped:out STD_LOGIC_Vector (2 downto 0); -- pedestrian light

east_ped:out STD_LOGIC_Vector (2 downto 0);


emergency: in STD_LOGIC;

--button_n: in STD_LOGIC;

--button_e: in STD_LOGIC;

clk : in STD_LOGIC;

reset : in STD_LOGIC

);


end traffic;


architecture Behavioral of traffic is
```

```
type state_type is (st0, st1, st2, st3, st4, st5, stx); -- stx is when
natural disasters occur so all lights are red except pedestrians

signal state: state_type:= st0;

signal count : std_logic_vector (3 downto 0);

constant sec10 : std_logic_vector ( 3 downto 0) := "1010";

constant sec2 : std_logic_vector (3 downto 0 ) := "0010";

constant sec16: std_logic_vector (3 downto 0 ) := "1111";


begin

process (clk, reset, emergency)

    begin

      if emergency = '1' then --emergency sensor

      state <= stx;

      count <= X"0";

      elsif reset='1' then

      state <= st0; --reset to initial state

      count <= X"0"; -- reset counter


  elsif rising_edge(clk) then


      case (state) is

      when st0 =>

        --if button_n = '1' then count <= count +2; end if;

        if count < sec10 then

          count <= count + 1;

          state <= st0;

        else

          state <= st1;

          count <= X"0";

        end if;

      when st1 =>
```

```
        --if button_n = '1' then count <= count - 2; end if;

if count < sec16 then
          count <= count + 1;
          state <= st1;
        else
          state <= st2;
          count <= X"0";
      end if;
      when st2 =>
        if count < sec2 then
         state <= st2;
         count <= count + 1;
        else
          state <= st3;
          count <= X"0";
        end if;
      when st3 =>
        --if button_e = '1' then count <= count + 2; end if;
        if count < sec10 then
          count <= count + 1;

state <= st3;
        else
          state <= st4;
          count <= X"0";
        end if;
      when st4 =>
        --if button_n = '1' then count <= count +2; end if;
        if (count < sec16) then
```

```
          state <= st4;
          count <= count + 1;
        else
          state <= st5;
          count <= X"0";
        end if;
      when st5 =>
        if count < sec2 then
         state <= st5;
         count <= count + 1;
        else
          state <= st0;
          count <= X"0";
        end if;
      when others =>
          state <= stx;
      end case;

end if;
end process;


OUTPUT_DECODE: process (state)
begin
case (state) is

-- R->G->Y
 --RYG
when st0 => north_tl <= "100";
north_left <= "001";
```

```
east_tl <= "100";
east_left <= "100";
north_ped <= "100";
east_ped <= "100";

when st1 => north_tl <= "001";
north_left <= "010";
east_tl <= "100";
east_left <= "100";
north_ped <= "100";
east_ped <= "001";

when st2 =>
north_tl <= "010";
north_left <= "100";
east_tl <= "100";
east_left <= "100";
north_ped <= "100";
east_ped <= "100";

when st3 =>
north_tl <= "100";
north_left <= "100";
east_tl <= "100";
east_left <= "001";
north_ped <= "100";
east_ped <= "100";

when st4 => north_tl <= "100";
north_left <= "100";
```

```
east_tl <= "001";
east_left <= "010";
north_ped <= "001";
east_ped <= "100";

when st5 => north_tl <= "100";
north_left <= "100";
east_tl <= "010";
east_left<= "100";
north_ped <= "100";
east_ped <= "100";

when stx => north_tl <= "100";
north_left <= "100";
east_tl <= "100";
east_left <= "100";
north_ped <= "001";
east_ped <= "001";

when others => north_tl <= "100";
north_left <= "100";
east_tl <= "100";
east_left <= "100";
north_ped <= "100";
east_ped<= "100";


end case;
end process;
end Behavioral;
```

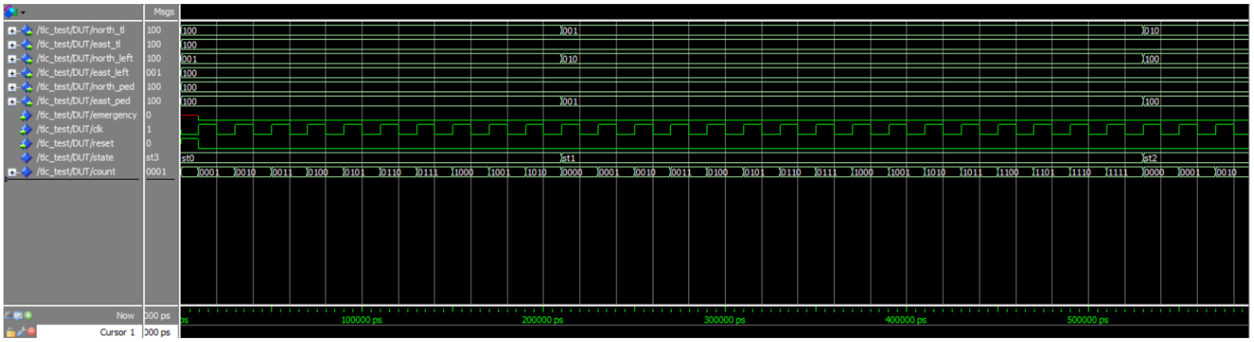
TEST

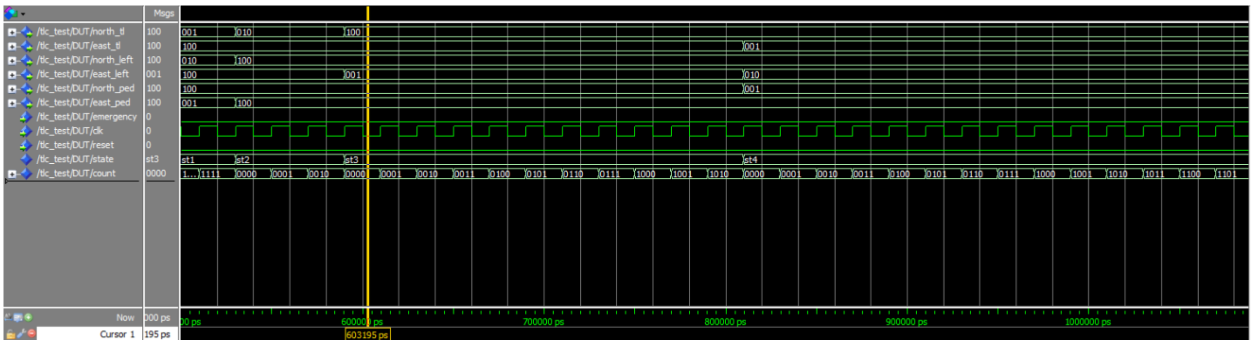
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity TLC_test is
end entity;

architecture tb of TLC_test is
  signal
north_tl,
east_tl,
north_left,
east_left,
north_ped,
east_ped: STD_LOGIC_Vector (2 downto 0);

  signal
--button_n, button_e,
emergency, clk, reset : STD_LOGIC;

begin
DUT : ENTITY work.traffic
PORT MAP(
north_tl=>north_tl,
east_tl=>east_tl,
north_left=>north_left,
east_left=>east_left,
north_ped=>north_ped,
east_ped=> east_ped,
```

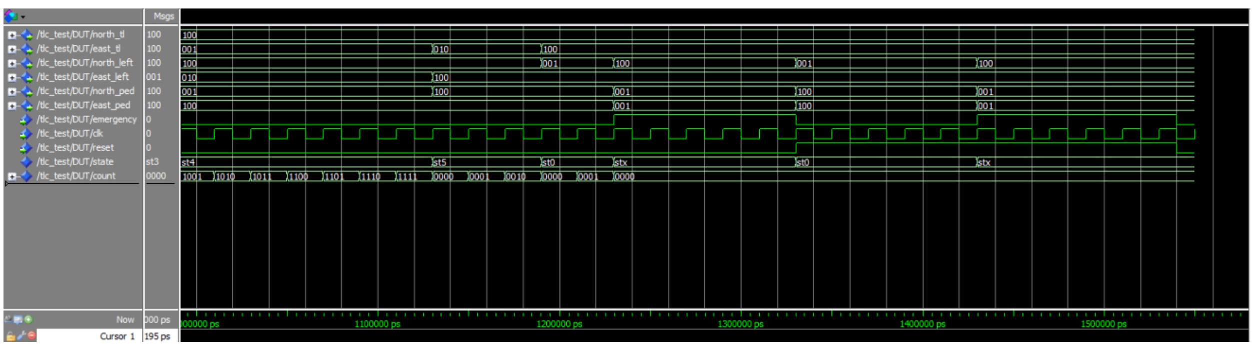
```
--button_n=> button_n,
--button_e=> button_e,
emergency=> emergency,
clk=>clk, reset=>reset
  );


Clock : process
begin
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end process;


stimulis : process
begin
report("Starting simulation");


reset <= '1';wait for 10 ns; --reset to st0
-- one normal cycle = (100ns+160ns+20ns)*2*2 = 1120 ns (not accurate,
only an estimate)
emergency<= '0'; reset<='0';wait for 1220 ns; -- a little more than
one normal cycle
emergency<= '1'; reset<='0';wait for 100 ns;
emergency<= '0'; reset<='1';wait for 100 ns;
emergency<= '1'; reset<='1';wait for 100 ns;
-- make sure to simulate for at least 1520 ns!!
```

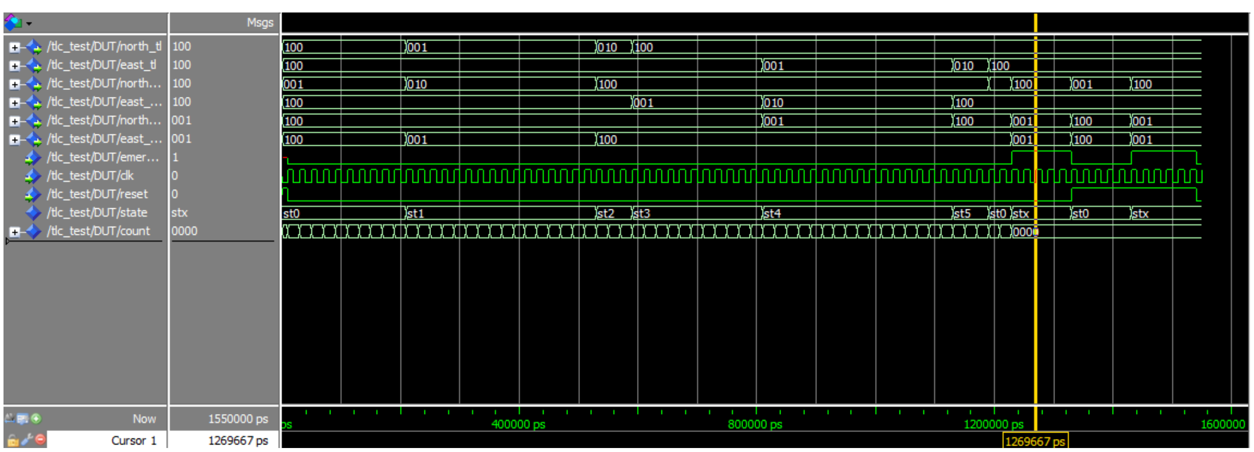
```
--emergency <= '0'; reset <= '0'; button_n <= '1'; wait for 200 ns; --
st0 shorter or st1 longer


report("End simulation");

end process;

end architecture;
```

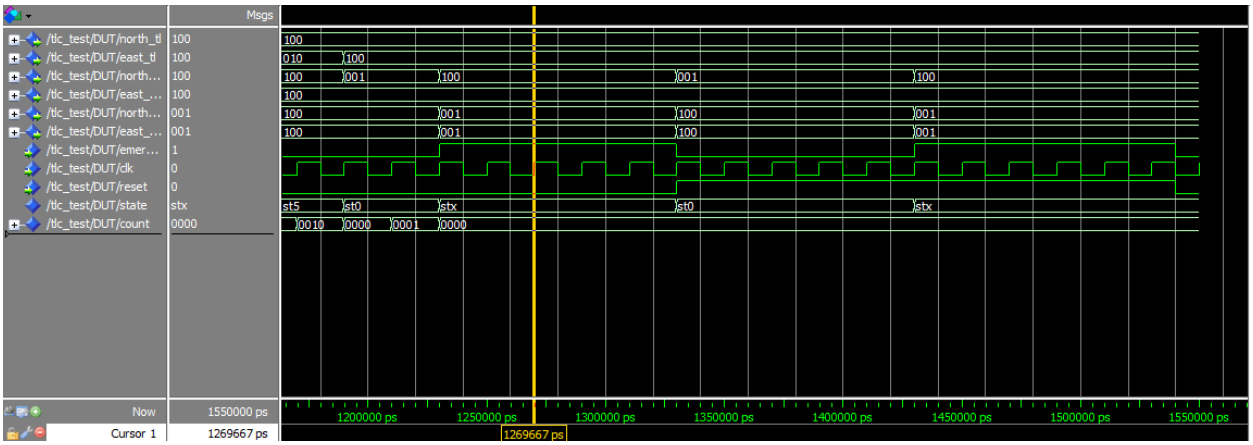
SIMULATION WAVE BITMAPS










Msgs
/tlc_test/DUT/north_tl
/tlc_test/DUT/east_tl
/tlc_test/DUT/north...
/tlc_test/DUT/east_...
/tlc_test/DUT/north...
/tlc_test/DUT/east_...
/tlc_test/DUT/emer...
/tlc_test/DUT/clk
/tlc_test/DUT/reset
/tlc_test/DUT/state
/tlc_test/DUT/count
100
010
001
0010
0000
0001
st5
st0
stx
Now
1550000 ps
Cursor 1
1269667 ps
1200000 ps
1250000 ps
1300000 ps
1350000 ps
1400000 ps
1450000 ps
1500000 ps
1550000 ps