**FINAL PROJECT: TRAFFIC LIGHT CONTROLLER**

Rachael Hsu and Rajat Kuthiala

University of Rochester

May 1, 2015

***Abstract* –** This project implemented elements learned from previous labs and were taken to a more advanced level. The objective was to design a traffic light controller. A VHDL code was implemented on the DE2 Development Environment using the Altera Programmable Logic Device. This environment was used to test the validity of the logic in the code and the Altera DE 2 board showed visual results of the simulation. The simulation gave results that were fairly close to what was expected.

***Requirements/Specifications:*** This project requires a traffic light controller that can simulate a traffic light intersection. The materials needed for this project were a computer with the DE2 Development Environment and the Altera Programmable Logic Device. On the Altera board, the LEDs and switches were the objects used for the simulation. The LEDs represented the outputs of the traffic lights while the switches acted as the input sensors at the intersection and turning lanes. This was required because in real life, traffic intersections have sensors built into the ground to make traffic run more efficiently instead of having traffic lights run on a consistent clock; in this case, the clock on the board ran at 1Hz. There would be an expected time delay on the changing lights due to the “sensors”. With all of these factors taken into account, a VHDL was written and compiled into the DE2 Development Environment and programmed into the Altera board for simulation.

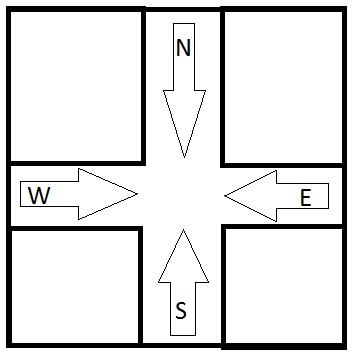
***Description of Operation:*** The lights would operate one after the other. Traffic from only one side of the road would move at a time. To obtain the functionality, the implemented circuit was based on an 1Hz clock so that the desired time interval shown below could be achieved.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | NORTH | SOUTH | EAST | WEST |
| RED | LEDR0 | LEDR3 | LEDR6 | LEDR9 |
| YELLOW | LEDR1 | LEDR4 | LEDR7 | LEDR10 |
| GREEN | LEDR2 | LEDR5 | LEDR8 | LEDR11 |

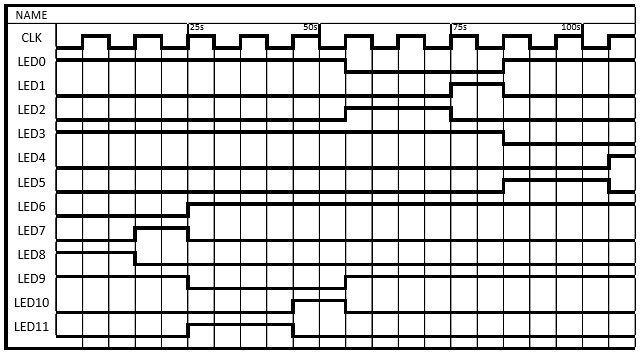
LED ASSIGNMENT

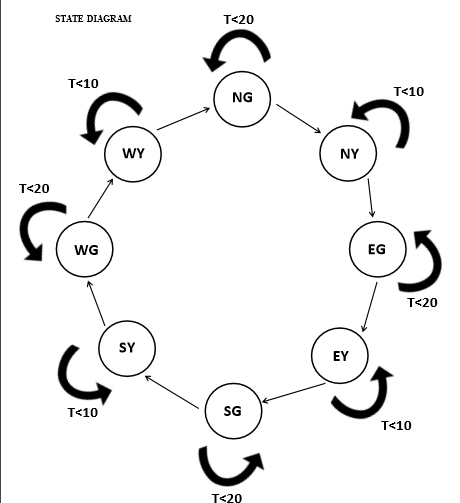
|  |  |
| --- | --- |
|  | Time |
| RED | 80 seconds |
| YELLOW | 10 seconds |
| GREEN | 20 seconds |

Time per Light

***Diagram & Drawing***: 

INTERSECTION LAYOUT

TIMING DIAGRAM



***Programs for Programmable Devices:*** The following VHDL code was compiled into the Altera QUARTUS II CAD system and was implemented into the DE2 Development Environment using the Altera Programmable Logic Device.

-- NAMES: Rajat Kuthiala & Rachael Hsu-- ASSIGNMENT: final project (traffic light controller)

-- DATE: 5/1

-- Library import stuff

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.numeric\_std.all;

-- The main entity of the project. Contains the port mappings to the board,

-- with clock for timing, switches for control, red LEDs for light outputs

-- and the 8 HEX displays for information output as well.

ENTITY project IS

PORT( CLOCK\_50 : IN STD\_LOGIC;

SW : IN STD\_LOGIC\_VECTOR(17 DOWNTO 0);

LEDR : OUT STD\_LOGIC\_VECTOR(17 DOWNTO 0);

HEX0 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX1 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX2 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX3 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX4 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX5 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX6 : OUT STD\_LOGIC\_VECTOR(0 TO 6);

HEX7 : OUT STD\_LOGIC\_VECTOR(0 TO 6));

END project;

-- Start definition of the behavior of the project.

ARCHITECTURE behavior OF project IS

-- Defining the various components.

-- In this code, the components handle the various output devices for the controller.

-- hex\_num component controls the countdown timer displayed on two of the hex displays.

COMPONENT hex\_num

PORT ( rr : IN std\_logic;

clk\_50 : IN std\_logic;

DISPLAY0 : OUT std\_logic\_vector(0 to 6);

DISPLAY1 : OUT std\_LOGIC\_VECTOR(0 to 6));

END COMPONENT;

-- direction\_hex controls the active directions display on two of the hex displays.

COMPONENT direction\_hex

PORT ( clk\_50 : IN std\_logic;

n, s, e, w,

nl, sl, el, wl,

rr : IN std\_logic;

DISPLAY0 : OUT std\_logic\_vector(0 to 6);

DISPLAY1 : OUT std\_logic\_vector(0 to 6));

END COMPONENT;

-- crosswalk\_hex controls the 4 crosswalk displays on the remaining hex displays.

COMPONENT crosswalk\_hex IS

PORT ( clk\_50 : IN std\_LOGIC;

dir, dir\_l, rr : IN std\_logic;

DISPLAY : OUT std\_logic\_vector(0 to 6));

END COMPONENT;

-- lights component controls the 4 sets of red leds that controls the

-- directions Red, Yellow, Green, and Left turn lights.

COMPONENT lights IS

PORT ( clk\_50 : IN std\_logic;

dir, dir\_l, rr : IN std\_logic;

DATA\_OUT : OUT std\_logic\_vector(3 downto 0));

END COMPONENT;

-- Signals used in the main controller.

-- counts keeps track of when a second has gone by and

-- seconds\_left is the countdown to the light change.

-- The rest represent each light and left turn state.

SIGNAL count : integer RANGE 0 to 50000000;

SIGNAL seconds\_left : integer RANGE 0 to 30;

SIGNAL north, north\_left,

south, south\_left,

east, east\_left,

west, west\_left : std\_logic;

BEGIN

-- create all the mappings between components and controllers.

count\_display : hex\_num PORT MAP (SW(17), CLOCK\_50, HEX4, HEX5);

dir\_display : direction\_hex PORT MAP ( CLOCK\_50,

north, south, east, west,

north\_left, south\_left, east\_left, west\_left,

SW(17), HEX7, HEX6);

north\_cross\_display : crosswalk\_hex PORT MAP (CLOCK\_50, north, north\_left, SW(17), HEX3);

east\_cross\_display : crosswalk\_hex PORT MAP (CLOCK\_50, east, east\_left, SW(17), HEX2);

south\_cross\_display : crosswalk\_hex PORT MAP (CLOCK\_50, south, south\_left, SW(17), HEX1);

west\_cross\_display : crosswalk\_hex PORT MAP (CLOCK\_50, west, west\_left, SW(17), HEX0);

north\_lights : lights PORT MAP (CLOCK\_50, north, north\_left, SW(17), LEDR(15 DOWNTO 12));

east\_lights : lights PORT MAP (CLOCK\_50, east, east\_left, SW(17), LEDR(11 DOWNTO 8));

south\_lights : lights PORT MAP (CLOCK\_50, south, south\_left, SW(17), LEDR(7 DOWNTO 4));

west\_lights : lights PORT MAP (CLOCK\_50, west, west\_left, SW(17), LEDR(3 DOWNTO 0));

-- The main controller of the project. Only performs on clock\_50 rising edge.

-- When a second has gone by, it updates the direction and left turn signals

-- to their appropriate states. If railroad is active, however, it holds a steady state

-- of east west active, at 5 seconds left on timer.

PROCESS(CLOCK\_50)

BEGIN

IF( CLOCK\_50'EVENT AND rising\_edge(CLOCK\_50)) THEN

IF (count >= 50000000) THEN

count <= 0;

IF (SW(17) = '0') THEN

seconds\_left <= seconds\_left -1;

if (seconds\_left <= 0) THEN

-- This is checking if the new set of directions have a left turn waiting.

IF (((SW(3) = '1' OR SW(1) = '1') AND east = '1') OR

((SW(2) = '1' OR SW(0) = '1') AND north = '1')) THEN

seconds\_left <= 30;

-- The first two check if both new directions have left turns waiting.

-- If so, set the two left turns and not the actual directions.

IF (SW(3) = '1' AND SW(1) = '1' AND east = '1') THEN

north <= '0';

south <= '0';

east <= '0';

west <='0';

north\_left <= '1';

south\_left <= '1';

east\_left <= '0';

west\_left <= '0';

ELSIF ((SW(2) = '1' AND SW(0) = '1') AND north = '1') THEN

north <= '0';

south <= '0';

east <= '0';

west <= '0';

north\_left <= '0';

south\_left <= '0';

east\_left <= '1';

west\_left <= '1';

-- Otherwise, check for the single left turns and set that direction and its left turn to '1'.

ELSIF (SW(3) = '1' AND east = '1')THEN

north <= '1';

south <= '0';

east <= '0';

west <= '0';

north\_left <= '1';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

ELSIF (SW(2) = '1' AND north = '1')THEN

north <= '0';

south <='0';

east<= '1';

west <='0';

north\_left <='0';

south\_left <='0';

east\_left <='1';

west\_left <='0';

ELSIF (SW(1) = '1' AND east = '1')THEN

north <='0';

south <='1';

east <='0';

west <='0';

north\_left <='0';

south\_left<= '1';

east\_left <='0';

west\_left <='0';

ELSIF (SW(0) = '1' AND north = '1')THEN

north <='0';

south <='0';

east <='0';

west <='1';

north\_left <='0';

south\_left <='0';

east\_left <='0';

west\_left <='1';

END IF;

ELSE

-- There are no left turns, so set the next set of directions.

seconds\_left <= 30;

IF (north = '1')THEN

north <= '0';

south <= '0';

east <= '1';

west <= '1';

north\_left <= '0';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

ELSE

north <= '1';

south <= '1';

east <= '0';

west <= '0';

north\_left <= '0';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

END IF;

END IF;

ELSIF (seconds\_left = 20) THEN

-- At 20 seconds, turn off all left turn lights.

IF (west\_left = '1' OR east\_left = '1') THEN

north <= '0';

south <= '0';

east <= '1';

west <= '1';

north\_left <= '0';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

ELSIF (north\_left = '1' OR south\_left = '1') THEN

north <= '1';

south <= '1';

east <= '0';

west <= '0';

north\_left <= '0';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

END IF;

end if;

ELSE

-- For railroad, hold the controller in this state indefinitely.

north <= '0';

south <= '0';

east <= '1';

west <= '1';

north\_left <= '0';

south\_left <= '0';

east\_left <= '0';

west\_left <= '0';

count <= 0;

seconds\_left <= 5;

END IF;

ELSE

count <= count + 1;

END IF;

END IF;

END PROCESS;

END;

-- LIBRARY STUFF

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

-- Define the ports for the entity

-- This controls 2 hex displays for displaying a

-- countdown until the lights switch.

ENTITY hex\_num IS

PORT ( rr : IN std\_logic;

clk\_50 : IN std\_logic;

DISPLAY0 : OUT std\_logic\_vector(0 to 6);

DISPLAY1 : OUT std\_logic\_vector(0 to 6));

END hex\_num;

-- define the logic for the entity.

ARCHITECTURE Behavioral OF hex\_num IS

SIGNAL count : integer RANGE 0 to 50000000;

SIGNAL seconds\_left : integer RANGE 0 to 30;

SIGNAL ones : integer RANGE 0 to 9;

SIGNAL tens : integer RANGE 0 to 9;

BEGIN

-- Process creates a counter to keep track of time.

PROCESS (clk\_50)

BEGIN

if (clk\_50'EVENT and rising\_edge(clk\_50)) then

if (rr = '0') then -- if railroad isn't active, count down

if (count >= 50000000) then

count <= 0;

seconds\_left <= seconds\_left -1;

if (seconds\_left <= 0) then

seconds\_left <= 30;

end if;

elsE

count <= count + 1;

end if;

else -- If railroad is active, hold at 5 seconds left.

count <= 0;

seconds\_left <= 5;

end if;

end if;

-- This is the actual logic of the entity.

-- Get the ones place and the tens place digits of the time left

ones <= seconds\_left mod 10;

tens <= seconds\_left / 10;

-- Find the correct code for the 7 segment display for a given number.

CASE ones IS

WHEN 0 => DISPLAY0 <= "0000001";

WHEN 1 => DISPLAY0 <= "1001111";

WHEN 2 => DISPLAY0 <= "0010010";

WHEN 3 => DISPLAY0 <= "0000110";

WHEN 4 => DISPLAY0 <= "1001100";

WHEN 5 => DISPLAY0 <= "0100100";

WHEN 6 => DISPLAY0 <= "0100000";

WHEN 7 => DISPLAY0 <= "0001111";

WHEN 8 => DISPLAY0 <= "0000000";

WHEN 9 => DISPLAY0 <= "0000100";

WHEN OTHERS => DISPLAY0 <= "1111111";

END CASE;

-- Same for 10s place.

CASE tens IS

WHEN 0 => DISPLAY1 <= "0000001";

WHEN 1 => DISPLAY1 <= "1001111";

WHEN 2 => DISPLAY1 <= "0010010";

WHEN 3 => DISPLAY1 <= "0000110";

WHEN 4 => DISPLAY1 <= "1001100";

WHEN 5 => DISPLAY1 <= "0100100";

WHEN 6 => DISPLAY1 <= "0100000";

WHEN 7 => DISPLAY1 <= "0001111";

WHEN 8 => DISPLAY1 <= "0000000";

WHEN 9 => DISPLAY1 <= "0000100";

WHEN OTHERS => DISPLAY1 <= "1111111";

END CASE;

-- If railroad is active, display "rr", not the numbers.

if (rr = '1') THEN

DISPLAY0 <= "1111010";

DISPLAY1 <= "1111010";

END IF;

END PROCESS;

END Behavioral;

-- LIBRARY STUFF

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

-- Entity declaration.

-- This displays the two current active directions.

-- For example, if North and South are active, it displays "ns".

ENTITY direction\_hex IS

PORT ( clk\_50 : IN std\_logic;

n, s, e, w,

nl, sl, el, wl,

rr : IN std\_logic;

DISPLAY0 : OUT std\_logic\_vector(0 to 6);

DISPLAY1 : OUT std\_logic\_vector(0 to 6));

END direction\_hex;

-- Behavior declaration

-- Just like all entities, this one begins with the clock logic.

-- After that, it has some logic to figure out what letters it is displaying,

-- for example a North South green light is different than a North North Left light.

-- Once its figured, the correct 7seg bit vectors are determined and displayed.

ARCHITECTURE Behavior OF direction\_hex IS

SIGNAL disN0, disE0, disS0, disW0, disL0,

disS1, disW1, disL1, odd : std\_logic;

SIGNAL count : integer RANGE 0 to 50000000;

SIGNAL seconds\_left : integer RANGE 0 to 30;

BEGIN

PROCESS (clk\_50)

BEGIN

if (clk\_50'EVENT and rising\_edge(clk\_50)) then

if (rr = '0') then

if (count >= 50000000) then

count <= 0;

seconds\_left <= seconds\_left -1;

if (seconds\_left <= 0) then

seconds\_left <= 30;

end if;

elsE

count <= count + 1;

end if;

elsE

count <= 0;

seconds\_left <= 5;

end if;

end if;

if ((seconds\_left mod 2) = 0) THEN

odd <= '0';

ELSE

odd <= '1';

END IF;

disN0 <= (n AND s) OR (nl AND (NOT(sl) OR odd));

disE0 <= (e AND w) OR (el AND (NOT(wl) OR odd));

disS0 <= sl AND NOT(nl);

disW0 <= wl AND NOT(el);

disL0 <= not(odd) AND ((nl AND sl) OR (el AND wl));

disS1 <= (n AND s) OR (nl AND sl AND NOT(odd));

disW1 <= (e AND w) OR (el AND wl AND NOT(odd));

disL1 <= NOT(disS1 OR disW1);

DISPLAY0(0) <= disW0 OR disL0 or disN0;

DISPLAY0(1) <= disN0 OR disS0 OR disE0 OR disL0;

DISPLAY0(2) <= disE0 OR disW0 OR disL0;

DISPLAY0(3) <= disN0;

DISPLAY0(4) <= disS0 OR disW0;

DISPLAY0(5) <= disN0;

DISPLAY0(6) <= disW0 OR disL0;

DISPLAY1(0) <= disW1 OR disL1;

DISPLAY1(1) <= disS1 OR disL1;

DISPLAY1(2) <= disW1 OR disL1;

DISPLAY1(3) <= '0';

DISPLAY1(4) <= disS1 OR disW1;

DISPLAY1(5) <= '0';

DISPLAY1(6) <= disW1 OR disL1;

END PROCESS;

END Behavior;

-- LIBRARY STUFF

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

-- Entity definition for a single crosswalk display.

-- Takes in the clock, direction, left turn for direction, railroad and hex display.

-- Displays the appropriate Hex symbol for the given crosswalk state.

-- - Nothing if the crosswalk is active and above 10 seconds.

-- - A number for active crosswalk between 0 and 9 seconds

-- - an X for a non-active crosswalk.

ENTITY crosswalk\_hex IS

PORT ( clk\_50 : IN std\_LOGIC;

dir, dir\_l, rr : IN std\_logic;

DISPLAY : OUT std\_logic\_vector(0 to 6));

END crosswalk\_hex;

-- Define behavior of entity.

-- Same as other entities, begin with the clock logic.

-- After that, figure out if the crosswalk is active.

-- If not, display the X.

-- If the railroad is active, display nothing. (as it is an active crosswalk)

-- Otherwise, display the number if the seconds left is <= 9,

-- otherwise display nothing.

ARCHITECTURE Behavior of crosswalk\_hex IS

SIGNAL cross : std\_logic;

SIGNAL count : integer RANGE 0 to 50000000;f

SIGNAL seconds\_left : integer RANGE 0 to 30;

BEGIN

PROCESS (clk\_50)

BEGIN

if (clk\_50'EVENT and rising\_edge(clk\_50)) then

if (rr = '0') then

if (count >= 50000000) then

count <= 0;

seconds\_left <= seconds\_left -1;

if (seconds\_left <= 0) then

seconds\_left <= 30;

end if;

elsE

count <= count + 1;

end if;

elsE

count <= 0;

seconds\_left <= 5;

end if;

end if;

cross <= not(dir) AND not(dir\_l);

if (cross = '0') THEN

DISPLAY <= "1001000";

ELSIF (rr = '1') THEN

DISPLAY <= "1111111";

ELSE

CASE seconds\_left IS

WHEN 0 => DISPLAY <= "0000001";

WHEN 1 => DISPLAY <= "1001111";

WHEN 2 => DISPLAY <= "0010010";

WHEN 3 => DISPLAY <= "0000110";

WHEN 4 => DISPLAY <= "1001100";

WHEN 5 => DISPLAY <= "0100100";

WHEN 6 => DISPLAY <= "0100000";

WHEN 7 => DISPLAY <= "0001111";

WHEN 8 => DISPLAY <= "0000000";

WHEN 9 => DISPLAY <= "0000100";

WHEN OTHERS => DISPLAY <= "1111111";

END CASE;

END IF;

END PROCESS;

END Behavior;

-- LIBRARY STUFF

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

-- Entity definition

-- This entity controls the 4 lights that represent Red, Yellow, Green, and Left Turn.

-- It takes in the clock, direction, left turn for direction, and railroad, and outputs a 4 bit vector.

ENTITY lights IS

PORT ( clk\_50 : IN std\_logic;

dir, dir\_l, rr : IN std\_logic;

DATA\_OUT : OUT std\_logic\_vector(3 downto 0));

END lights;

-- Define the behavior for the entity.

-- A temp 4 bit vector is used to hold the values of the lights to be passed to the output bit vector.

-- As with the other entities, the clock logic is at the beginning.

-- After that, figure out the set of lights from the direction, left direction, railroad, and time.

-- The order of bits is Red, Yellow, Green, Left.

ARCHITECTURE Behavior of lights IS

SIGNAL temp : std\_logic\_vector(3 downto 0);

SIGNAL count : integer RANGE 0 to 50000000;

SIGNAL seconds\_left : integer RANGE 0 to 30;

BEGIN

DATA\_OUT <= temp;

PROCESS (clk\_50)

BEGIN

if (clk\_50'EVENT and rising\_edge(clk\_50)) then

if (rr = '0') then

if (count >= 50000000) then

count <= 0;

seconds\_left <= seconds\_left -1;

if (seconds\_left <= 0) then

seconds\_left <= 30;

end if;

elsE

count <= count + 1;

end if;

elsE

count <= 0;

seconds\_left <= 5;

end if;

end if;

-- If this direction isn't active, figure out if the left turn is, and update the vector accordingly.

IF (dir = '0') THEN

IF (dir\_l = '1') THEN

temp <= "1001";

ELSE

temp <= "1000";

END IF;

ELSE

-- This means the direction is active. If railroad is active, then hold it at green indefinitely.

-- Otherwise, if its more than 5 seconds left, set it to green, and if left turn is active, set that as well.

-- If its less than 5 seconds, that means its yellow.

IF (rr = '1') THEN

temp <= "0010";

ELSIF (seconds\_left > 5) THEN

IF (dir\_l = '1') THEN

temp <= "0011";

ELSE

temp <= "0010";

END IF;

ELSE

IF (dir\_l = '1') THEN

temp <= "0101";

ELSE

temp <= "0100";

END IF;

END IF;

END IF;

END PROCESS;

END Behavior;

***Conclusion:*** After the VHDL code was compiled onto the DE2 Development Environment and programmed to the Altera Programmable Logic Device, the simulation produced results that were fairly close to what was expected . The project met all the specifications kept in mind for the design. During the simulation, the lights switched after the proper time interval that was defined in the VHDL code. When measuring the time taken to change from one state to another, the measured time was greater than the expected time by .4 seconds, which could be due to the time delay in the circuit.

***Weaknesses:*** 1)The traffic lights are programmed to switch in a sequence; thus meaning that

the lights simultaneously change first in the north and south lanes, then the

east-west lanes.

2) Drivers need to wait even though there may be no other cars in the other

junction.

3) The timing ratio of Yellow light to Green light is not much, which needs to be fixed.