**ECE 4250/ 7250: VHDL and Programmable Logic Devices  
Laboratory**

**Lab #6  
Lab Title: Design and Simulation of Traffic Light Controller**

**Group #2  
Group Names: Chris Smith, Benjarit Hotrabhavananda**

**Teaching Assistant Use Only:**

**Points Earned Reasons for Deduction**

**Pre-lab:**

**Post Lab report:**

**Demonstration:**

**Final Lab Grade:**

**Comments to students:**

**I. Objective**

The objective is to learn the VHDL modeling of a State Machine. A traffic light controller is selected as an example.

**II. Introduction**

Our system should be designed to represent the scenario of enabling the students to cross safely between the EBW and EBE, any pedestrian who wants to cross the street needs to push the button once to get a “WALK” signal; so we need this push button as input in addition to clock signal; and we need Green; yellow and Red signals for cars and walk and nowalk signals for pedestrian as output, However this system should follow some rules as follows:

• Yellow light for cars will last 4 seconds.

• The green light for cars will last at least 32 seconds.

• The “WALK” sign for the pedestrian will last 20 seconds including 4 seconds of

“WALK” flashing.

• There will be “NO WALK” sign when the yellow or green light is on.

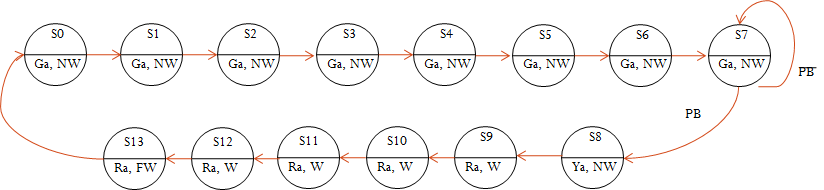
• When no one intends to pass the street, the green light for cars should remain ‘ON’

even after 32 seconds.

• The clock period will be 4 seconds. Another clock with a period of 1 second should be

available for flashing the “WALK” sign.

**Lab Work**



**Fig 1**

We noticed from the **Fig 1** that the states will be change from 0 to 13 spending 4 seconds in each state; so the cars will move according to the Ga which will be one for 32 seconds from S0 to S7 with NW signal to the pedestrian; then if the PB is pushed; this will lead to change Ga to 0 and give one to the Ya signal to warn the cars to stop (this happened in state 8) ; then after 4 seconds, state 9 to 12 will enable Ra to cars and W to pedestrian (16 seconds); after that the W signal begin to flash also for 4 seconds to warn the pedestrians that Ga for cars will begin again and NW will be on.

Our system will consist of two processes, one process is to select the next state according to the current state and PB, also the process will update the output signals; the other process will update the state according to 4 seconds clock generated by using the Gen clock generator from the last lab; it has been included as component; we need this component to generate two types of clock which has been named to clock1 and clock2, clock 1 is 4 second for updating the states, clock2 is 1 second for flashing the walk signal; this system is well demonstrated in fig 2 so that you can see the input, output, clock generator and how the internal signals are connected.

**IV. Conclusion:**

A traffic light controller system has been designed, connected to the LEDs on Spartan-3 FPGA, and loaded it to it, it has been implemented successfully in which we can see the output signals go on, off and also we can notify the flashing of W signal after we push the PB button, some problems occurred through the programming such as forgetting to define the internal signals and incompatibility problems, but they are all solved and work successfully.

**The Code:**

**Traffic\_light.vhd**

library ieee;

use ieee.std\_logic\_1164.all;

ENTITY traffic\_light3 is

PORT (clk,PB: in std\_logic;

Ga, Ya, Ra, WALK, NOWALK: out std\_logic);

END traffic\_light3;

architecture behave of traffic\_light3 is

signal state,nextstate: integer range 0 to 13:=0;

signal clock1, clock2: std\_logic := '1' ;

begin

process(state,PB,clock2)

begin

Ga <= '0'; Ya <= '0'; Ra <= '0'; WALK <= '0'; NOWALK <= '0'; --sFW <= '0';

case state is

when 0 to 6 =>

nextstate <= state + 1; Ga <= '1'; NOWALK <= '1';

when 7 =>

if PB = '1' then nextstate <= 8; NOWALK <= '1'; Ga <= '1';

else nextstate <= 7; Ga <= '1';

end if;

when 8 =>

nextstate <= state + 1; Ya <= '1'; NOWALK <= '1';

when 9 to 11 =>

nextstate <= state + 1; Ra <= '1'; WALK <= '1';

when 12 =>

nextstate <= state + 1; Ra <= '1'; WALK <= '1';

when 13 =>

nextstate <= 0; Ra <= '1'; WALK <= clock2;

end case;

end process;

process(clock1)

begin

if clock1'event and clock1 = '1' then

state <= nextstate;

end if;

end process;

--For Simulation purpose

clock1 <= not clock1 after 4000 ms;

clock2 <= not clock2 after 1000 ms;

--For Synthesis purpose

--c1: GenClock generic map(1) port map(clk, clock1);

--c2: GenClock generic map(2) port map(clk, clock2);

end behave;