Traffic Light Control

Melvin Pulianthuruthil, Jason Fisher, Joseph Yatooma, Merna Aziz

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: pulianthuruthil@oakland.edu, jasonfisher@oakland.edu, batoma@oakland.edu, Maziz@oakland.edu

***Abstract* – The main focus of this project is to use Vivado's VHDL program and the Nexys A7 board to develop a traffic light control system for the day and the night. The logic for the circuit is created and implemented in VHDL. The traffic light consists of two modes, one that will be employed at night (called night mode) and another that will be used during the day (day mode.) The most difficult part of the project was implementing all the timers and synchronizing them to operate correctly that control the transitions of the traffic light.**

I. INTRODUCTION

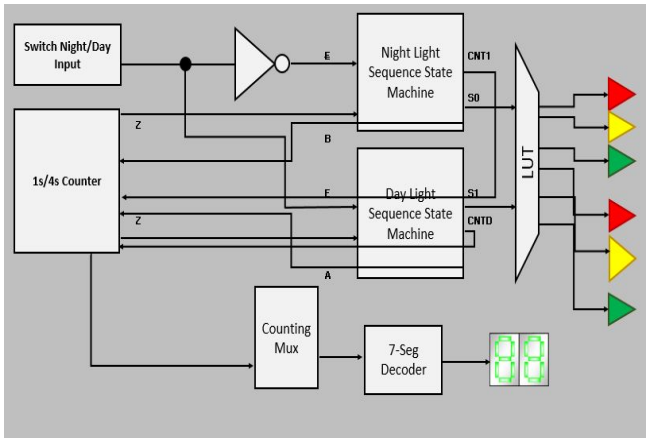
The goal of this project is to learn and develop a traffic light control system for day and night; during the day more cars and people will be on the streets which require longer periods of transitions in busy areas to improve the system and ease traffic. However, at night some intersections become almost obsolete where cars are going only one way. In such cases, it's better to keep the light on green for longer for the busier side of the road. This report will cover the methods, tools, components and the code used to build the circuit that can control the traffic light.

The main motivation for this project comes from daily lives. There are times where traffic is non-existent one way and yet the lights stay on red wasting valuable time and gas. The ultimate goal of this project is to explore ideas of building smarter traffic systems that can reduce wait time for drivers, saving them time and saving the environment by reducing carbon emissions from cars stopping at red lights. The current advancements in technology allow us to create an efficient traffic light control system that is able to analyze the current situation of traffic and produce an appropriate lights cycle for the lights to allow cars and people move easily, quickly and more efficiently. The project consists of four main blocks: the mode will be controlled by a switch on the Nexys board, during the day, the green light is going to stay lit for 20 seconds, and in night for 12 seconds. Yellow light stays for 4 seconds both day and night. Red light will stay lit for 24 seconds in the day and 16 seconds in the night. There will be a short delay before the green light turns on in one way after the red light is turned on the other way.

II. METHODOLOGY

a) Mode Switch

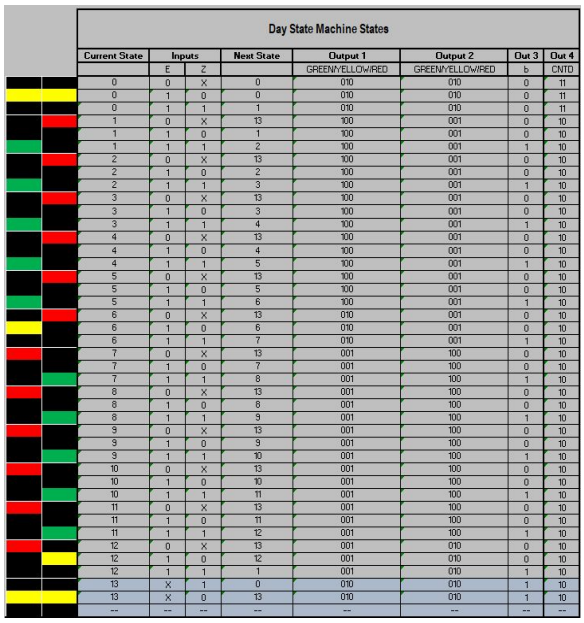
There are different ways to implement this design; the way it was implemented in this project was using the techniques taught mainly in class. A switch 1 on the Nexys board is going to determine the mode of the lights, the command is sent to two different finite state machines that take that input and the output of multiple counters and determines the desired output needed to control the counter and the LEDs.



b) FSM

The finite state machine is the main controller of the circuit and is the one responsible for providing the logic for controlling the LEDs and the counter. Two

different state machines were designed for the two modes of the light. The first state machine for the day mode is shown below in figure 2.

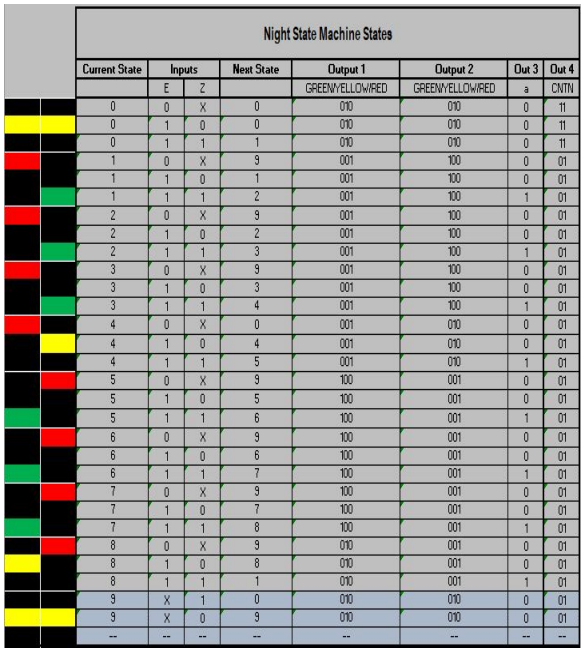


Day State Machine States

Current State	Inputs		Next State	Output 1	Output 2	Out 3	Out 4
	E	Z		GREEN/YELLOW/RED	GREEN/YELLOW/RED	b	CNTD
0	0	X	0	010	010	0	11
0	1	0	0	010	010	0	11
0	1	1	1	010	010	0	11
1	0	X	13	100	001	0	10
1	1	0	1	100	001	0	10
1	1	1	2	100	001	1	10
2	0	X	13	100	001	0	10
2	1	0	2	100	001	0	10
2	1	1	3	100	001	1	10
3	0	X	13	100	001	0	10
3	1	0	3	100	001	0	10
3	1	1	4	100	001	1	10
4	0	X	13	100	001	0	10
4	1	0	4	100	001	0	10
4	1	1	5	100	001	1	10
5	0	X	13	100	001	0	10
5	1	0	5	100	001	0	10
5	1	1	6	100	001	1	10
6	0	X	13	010	001	0	10
6	1	0	6	010	001	0	10
6	1	1	7	010	001	1	10
7	0	X	13	001	100	0	10
7	1	0	7	001	100	0	10
7	1	1	8	001	100	1	10
8	0	X	13	001	100	0	10
8	1	0	8	001	100	0	10
8	1	1	9	001	100	1	10
9	0	X	13	001	100	0	10
9	1	0	9	001	100	0	10
9	1	1	10	001	100	1	10
10	0	X	13	001	100	0	10
10	1	0	10	001	100	0	10
10	1	1	11	001	100	1	10
11	0	X	13	001	100	0	10
11	1	0	11	001	100	0	10
11	1	1	12	001	100	1	10
12	0	X	13	001	010	0	10
12	1	0	12	001	010	0	10
12	1	1	1	001	010	1	10
13	X	1	0	010	010	1	10
13	X	0	13	010	010	1	10
--	--	--	--	--	--	--	--

Figure 2: Day State Machine States

Another state machine was used to control the LEDs and the stopwatch that in turn controls the seven segment display on the Nexys board. Figure 3 show the night mode state machine.



Night State Machine States

Current State	Inputs		Next State	Output 1	Output 2	Out 3	Out 4
	E	Z		GREEN/YELLOW/RED	GREEN/YELLOW/RED	a	CNTN
0	0	X	0	010	010	0	11
0	1	0	0	010	010	0	11
0	1	1	1	010	010	0	11
1	0	X	9	001	100	0	01
1	1	0	1	001	100	0	01
1	1	1	2	001	100	1	01
2	0	X	9	001	100	0	01
2	1	0	2	001	100	0	01
2	1	1	3	001	100	1	01
3	0	X	9	001	100	0	01
3	1	0	3	001	100	0	01
3	1	1	4	001	100	1	01
4	0	X	0	001	010	0	01
4	1	0	4	001	010	0	01
4	1	1	5	001	010	1	01
5	0	X	9	100	001	0	01
5	1	0	5	100	001	0	01
5	1	1	6	100	001	1	01
6	0	X	9	100	001	0	01
6	1	0	6	100	001	0	01
6	1	1	7	100	001	1	01
7	0	X	9	100	001	0	01
7	1	0	7	100	001	0	01
7	1	1	8	100	001	1	01
8	0	X	9	010	001	0	01
8	1	0	8	010	001	0	01
8	1	1	1	010	001	1	01
9	X	1	0	010	010	0	01
9	X	0	9	010	010	0	01
--	--	--	--	--	--	--	--

c) LUT

The look up table provides the output that turns the LEDs on. The input of the look-up table comes from the output of the state machine.

d) Counter

The counter purpose is to trigger the different seven segment displays as well as triggering the state machines. It also receives reset inputs from the output of the state machines.

e) Top File:

The top file is where all the components that make up the circuit are connected together using mainly port map statements in VHDL.

The counter and the switch are connected to the two state machines to control the mode and the time of the light cycles.

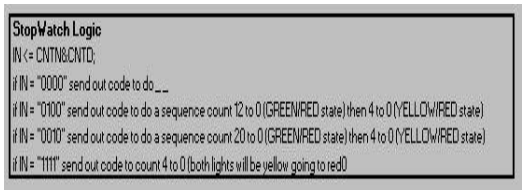


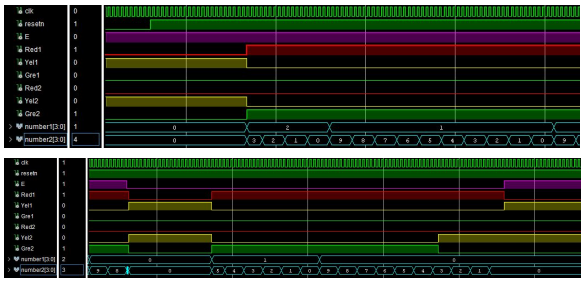
Figure 4: Stopwatch Logic

III. EXPERIMENTAL SETUP

The components that make up this project were built in VHDL using Vivado., each component was built separately to minimize errors and simplify the debugging process; many of the components were simulated individually using a custom test bench to verify their functionality.

The top file which is where all components come together is the most important one to simulate. A test bench was written to verify that the circuit produces the desired outcome.

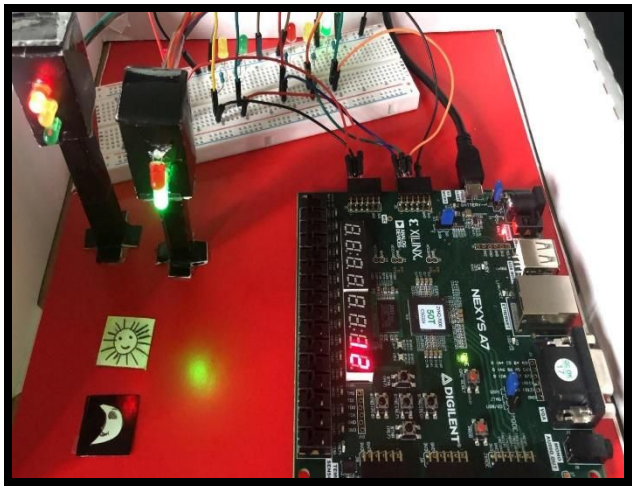
To further verify that the circuit works a breadboard and LEDs were used to test the code before building any structure or physical system



```
begin
cnt: FinalTop port map(resetn=>resetn,clk=>clk,segs=>
clk_process: process
    begin
    clk<='0'; wait for 5ns;
    clk<='1'; wait for 5ns;
    end process;

    stimulation_process: process
    begin
        resetn <= '0';
        E <= '1';
        wait for 120ns;
        resetn <= '1';
        wait for 1000ns;
        E <= '0';
        wait for 1000ns;
        E <= '1';
        wait for 600ns;
        E <= '0';

        wait;
    end process;

end Behavioral;
```

IV. Results

As previously mentioned the circuit was built using the techniques learnt in the digital logic design course. The circuit functions exactly like expected where a state machine is the main controller and other components like counters, LUT and logic gates are controlled by that state machine. A physical model representing a traffic light was built to test the behavior of the circuit. The picture below shows the model



CONCLUSION

In conclusion, a traffic light digital control system can be built relatively easily with basic understanding of the main building blocks of a digital system, and some VHDL coding knowledge. A lot of improvements can be implemented to this design that will make it more useful. Machine vision can be utilized to detect if there are cars going a certain way and decide to activate the light or not, a flashing yellow light can be used at times where there is no traffic and a lot of other minor improvements.

REFERENCES

[1]. Llamocca, Daniel. VHDL Coding for FPGAs, www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html