Lab 7 Report: Traffic Light State Machine

Seth Kurtenbach
Angelique Taylor
Dylan Samson

1. Objective

The objective of this lab is to simulate a controller for a simple traffic light, in order to gain experience with VHDL's state machine modeling capabilities. We assume the traffic light sits at the intersection of a crosswalk and a street. Thus, the states of the system shall be the color status of the traffic light conjoined with the “walk” “do not walk” signals for the pedestrians. The system must be responsive to a pedestrian input. We simulate this system on a Xilinx Project Navigator using seven-segment displays and a single input button.

2. Lab Work

This section presents the code we implemented in simulating the described traffic light system. Our traffic system was to meet the following requirements:

1. The yellow light for cars will last 4 seconds.
2. The green light for cars will last at least 32 seconds.
3. The WALK sign for pedestrians will last 20 seconds, including 4 seconds of WALK flashing on/off in 1 second cycles.
4. There will be NO WALK sign when the yellow or green light is on.
5. When no one intends to cross the street, the green light for cars should remain on after the initial 32 seconds elapses.
6. The clock period will be 4 seconds. Another clock with a period of 1 second will be available for flashing the WALK sign.

We define the traffic system functionality in the file Traffic_Light.vhd, which we present below.

```
ENTITY traffic_light is
PORT (clk, SB: in std_logic;
      Ga, Ya, Ra, WALK, NOWALK: inout std_logic);
END traffic_light;
```

The entity describes the clock and pedestrian input SB, which equals 1 when a pedestrian is pushing it, 0 otherwise. The outputs to the entity are the light signals: Ga is the green light for cars, Ya is yellow, Ra is red, WALK is the pedestrians' walk signal, and NOWALK is the pedestrians' signal to not walk, all of which equal 1 when the respective signal is lit up, 0 otherwise. We turn now to the entity's architecture.

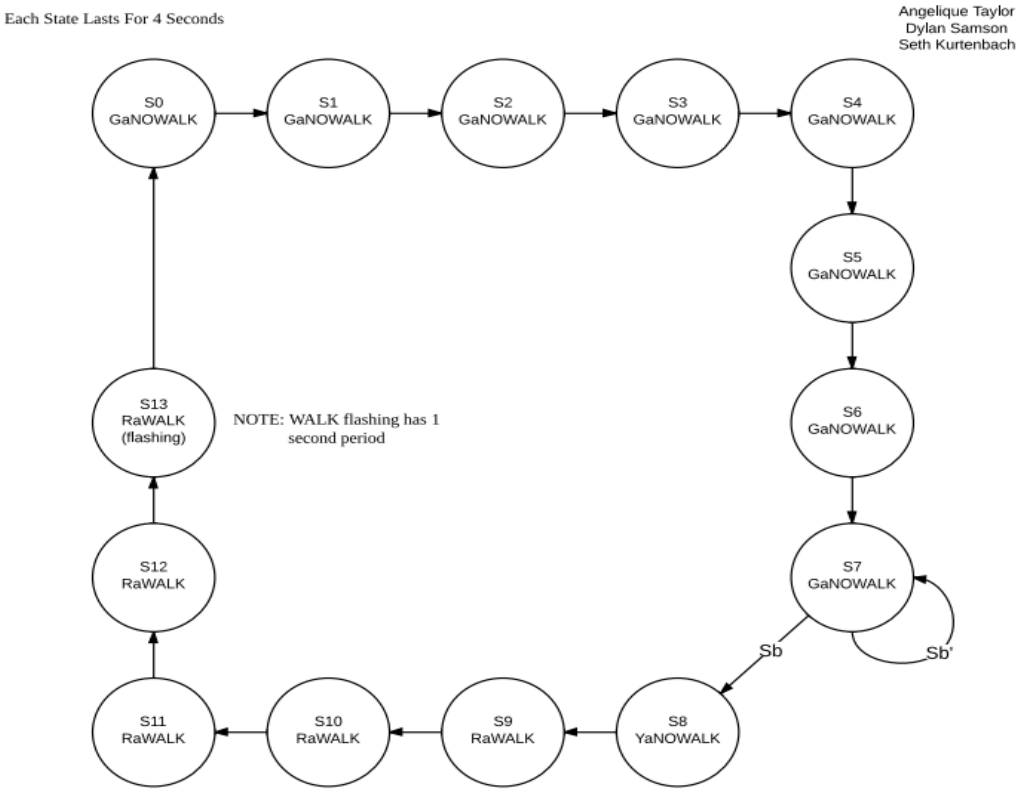
```
architecture behavior of traffic_light is

component lab5 is
generic( time_period : integer range 1 to 4 );
port( clk: std_logic;
      Clock : out std_logic );
end component;

signal state, nextstate: integer range 0 to 13;
type light is (R, Y, G);
signal lighta: light;
signal clk1, clk2: std_logic;
signal SOLIDWALK, FLASHWALK: std_logic;
```

This section of code shows our traffic signal incorporating a clock component defined in a previous lab. Lab5's clock is really four clocks of varying periodicity, (.5s, 1s, 2s, 4s). We use VHDL's generics feature to refer to grant us freedom concerning which clock period to instantiate. We use internal signals to store the value of the system's current state, *state*, and its next state, *nextstate*, which can each take an integer value in the range of 0 to 13, as the system has a total of 14 possible states. Additionally, we define a new data type, *light*, which instantiates to values *R, Y, G*, representing red, yellow, and

green. This aspect of the code is for simulation purposes only, and does not affect the traffic system's implementation on the Xilinx board. The internal signals *clk1, clk2, SOLIDWALK,* and *FLASHWALK* store flags for handling the two different clocks of the system and the flashing of the WALK signal in the final 4 seconds.

```
L1: lab5 generic map(2) port map(clk, clk2);
L2: lab5 generic map(4) port map(clk, clk1);
process(state, SB)
begin
Ra <= '0'; Ga <= '0'; Ya <= '0'; NOWALK <= '0'; SOLIDWALK <= '0'; FLASHWALK <= '0';
case state is
when 0 to 6 => Ga <= '1'; NOWALK <= '1'; nextstate <= state + 1;
when 7 => if Sb = '1' then nextstate <= 8; Ya <= '1'; nowalk <= '1';
                      else nextstate <= 7; Ga <= '1'; NOWALK <= '1'; end if;
when 8 to 12 => Ra <= '1'; SOLIDWALK <= '1'; FLASHWALK <= '0'; NOWALK <= '0'; nextstate <= state + 1;
when 13 => Ra <= '1';
    FLASHWALK <= '1';
    SOLIDWALK <= '0';
    nextstate <= 0;
end case; end process;
process(clk1)
begin
if clk1'event and clk1 = '1' then
state <= nextstate; end if;
end process;
lighta <= R when Ra = '1' else Y when Ya = '1' else G when Ga = '1';
WALK <= '1' when SOLIDWALK = '1'
        else clk2 when FLASHWALK = '1';
end behavior;
```

We instantiate the lab 5 clock component twice, once with the 1s period clock, and once with the 4s period clock. Our architecture proceeds in two processes. The first process details the assignment of values to the internal signals *nextstate, FLASHWALK, SOLIDWALK*, and the output signals WALK, NOWALK, Ga, Ra, Ya. The second process manages the transition from the current state to the next state. The first process is based on the following state transition diagram.



In the diagram, the presence of a signal name in a state indicates that its value is 1, and its absence from the state indicates its value is 0. So, for example, in state 10, Ra = 1 and WALK = 1, that is, the cars' light is red and the pedestrians' WALK sign is lit up. As is clear from the diagram, the light is green for 32 seconds, and then repeats state 7 until the pedestrian presses the button. Once pressed, the system transitions to state 8, where a 4 second period of yellow light and No Walk occurs. From 8, it transitions to state 9, where begins a 16 second period of red light and solid walk, followed by a 4 second period of red light and flashing walk. Thus, all of the above requirements are met by our system.

In order to implement the system on the Xilinx system, we made the following adjustments to the following code, which was provided by the lab instructors.

```
53    TrafficLight:   traffic_light port map (clk,Sb,Ga,Ya,Ra,WALK,NOWALK);   --clk is 50 MHz
54    LEDDisplay0:    LEDDisplay  port map (Ga, Ya, Ra, WALK, NOWALK, counter,
55                                          segment_a, segment_b, segment_c, segment_d,
56                                          segment_e, segment_f, segment_g);
57    ANDisplay:  AnodeControl port map (clk, counter, Anode);
58
59    end Behavioral;
```

We provided the appropriate port map values in the LEDDisplay instance LEDDisplay0. These values manage the interface between the program and the Xilinx 7-segment display.

```
32    process (WALK, NOWALK)  -- Fill in the sensitivity list
33    begin
34        if (WALK='1' and NOWALK='0') then
35            hex_digit1 <= "0000";
36        elsif (WALK='0' and NOWALK='1') then
37            hex_digit1 <= "0010";
38        else
39            hex_digit1 <= "0010";
40        end if;
41    end process;
42
43    process (Ga, Ya, Ra)    -- Fill in the sensitivity list
44    begin
45        if (Ga='1' and Ya='0' and Ra='0') then
46            hex_digit2 <= "0010";
47        elsif (Ga='0' and Ya='1' and Ra='0') then
48            hex_digit2 <= "0000";
49        elsif (Ga='0' and Ya='0' and Ra='1') then
50            hex_digit2 <= "0001";
51        else
52            hex_digit2 <= "0010";
53        end if;
54    end process;
```

The 7-segment display is managed by the above processes, for which we included the appropriate sensitivity lists. The first process manages the interface between the WALK/NOWALK values and the 7-segment display representing those signals. The display was to read '2' for NOWALK, '0' for WALK, and alternate between '0' and '2' to indicate a flashing WALK signal. Thus, the process assigns the hexadecimal value '0' when WALK = 1 and NOWALK = 0. If WALK = 0 and NOWALK = 1, the hexadecimal value '2' is assigned to the 7-segment digit. As this process depends on the values of

NOWALK, and WALK, we include them in the sensitivity list. Similar reasoning applies to our decision to include Ga, Ya, and Ra in the sensitivity list of the second process. This yields the following results:

1. “22” displayed for green light and no walk.
2. “02” displayed for yellow light and no walk.
3. “10” displayed for red light and solid walk.
4. “10/12” interchanging display for red light and flashing walk. The system then returns to 1.

3. Conclusion

We experienced several problems in completing this lab. The first concerned incorporating the clock from Lab 5 into this lab's code. The difficulty here was in understanding how the Lab 5 code related to our code on a system level, that is, how to connect the appropriate aspects of each system, and where. Solving this problem simply took time and discussion of the code at hand.

Another problem we experienced was in representing the flashing walk sign. We included the 1 second period clock from Lab 5 and mapped it to the signal *clk1*, and created a process with *clk1* in the sensitivity list. To capture the change in WALK sign value for every clock tick, we initially based the alternating value on each clock tick event, rather than on each rising clock edge. We solved this problem by discussing the code with the lab instructor.

Finally, we struggled in implementing the system on the Xilinx machine, because in order for the machine to remain in the state indicating a yellow light and a No Walk sign, the pedestrian's button needed to be continuously held, due to our construction in of the process managing *nextstate* and color values. It took us some time to discover that our system was actually working correctly, and it was a matter of experimenting with a series of push button and hold button inputs.