Engineering 32B

Lab 9, 10, and 11: Traffic Lights Final Report

Bench: 1

Writer: Sam Rork, Danny Elhalamawy, Mirsan Padilla

Partner(s): Sam Rork, Danny Elhalamawy, Mirsan Padilla

Report Completed: 16 December 2023

Instructor: Kwong

Abstract

The goal of this project is to make a traffic light logic system capable of handling many different situations for a traffic light situated at an intersection between a busy highway and a less busy farm road. An initial project file was given to work off of, and the aim of the project was to implement many improvements on top of the given code.

I. Specification

The given project file had functionality for a consistent green light on the highway, unless a car was detected waiting to cross through the farm road. Then, if the highway had been green for long enough already, the light would turn, the farm road car(s) would go, and the light would give way for the highway cars once again.

The improvements applied to this initial system are as follows:

- a coinciding red-light cycle, so that all cars are stopped for a brief period of time after yellow-lights
- protected left-turn signals, initiated by sensors on each road
    - this means 2 different new state cycles, one for highway turns and one for farm road turns
- crosswalk functionality, one initiated by an input system
    - to cross across the highway, input is needed, but to cross across the farm road, pedestrians must simply wait until the highway has a green light
- Tri-color LED functionality for the farm road and highway lights

In total, these improvements resulted in 18 new states, 4 new inputs and 8 new outputs.

II. Project Design Description

This project represents a working traffic light system because it is able to handle most interactions that may happen at an FM-Highway intersection. The only interactions not handled by this system are Emergency Vehicle Overrides for police vehicles or fire trucks, and hardware failure-induced stop signs.

a. Complete Circuit Schematics

The block diagram below represents the generalized form of the top.sch schematic file. The round blocks represent inputs and outputs (and the pentagon), the blocks represent logic systems, and the arrows follow the direction of logical flow.

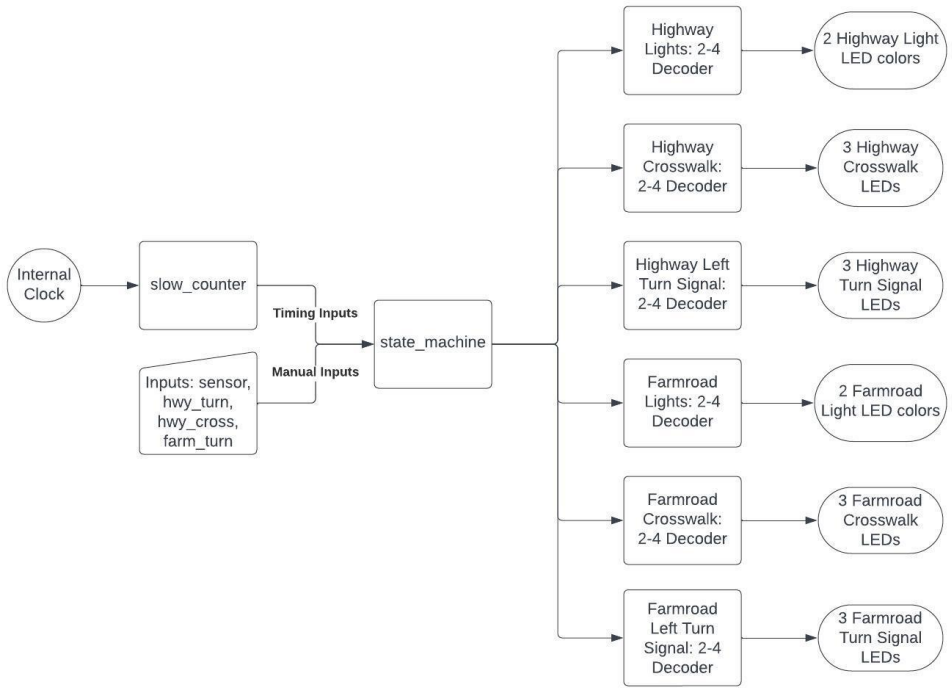

Figure 1: Block diagram of the .sch file, created using lucid.app

The block labeled slow_counter takes the internal clock and parses it to give slower clocks as output. The block labeled state_machine takes these timing inputs, along with manual inputs “sensor”, “hwy_turn”, “hwy_cross”, and “farm_turn” to facilitate the transition between different states. The output from state_machine is fed into 6 different 2-4 decoders, which activate and deactivate the various different LED lights to indicate different signals for the traffic light.

b. Component Schematics

The schematic file shown below is the final top.sch file for the project. All interactions between parts are accomplished here, bringing each individual circuit together to form a cohesive whole. On the very left side, the slow_counter vhdl module handles slowing down the internal clock of the FPGA board (from the input labeled “clock”) to create the long, medium, and short timers necessary for timing in the next module, state_machine. The state_machine module handles all the logic for travel between states, and thus also requires the new inputs hwy_turn, farm_turn, and hwy_cross. The outputs of the state_machine are 2-bit logic for each set of outputs to be fed into 2-to-4 decoders, which finally reaches the final output LEDs. The highway and farm road lights use the tri-color LED system, and therefore have a slightly different output.

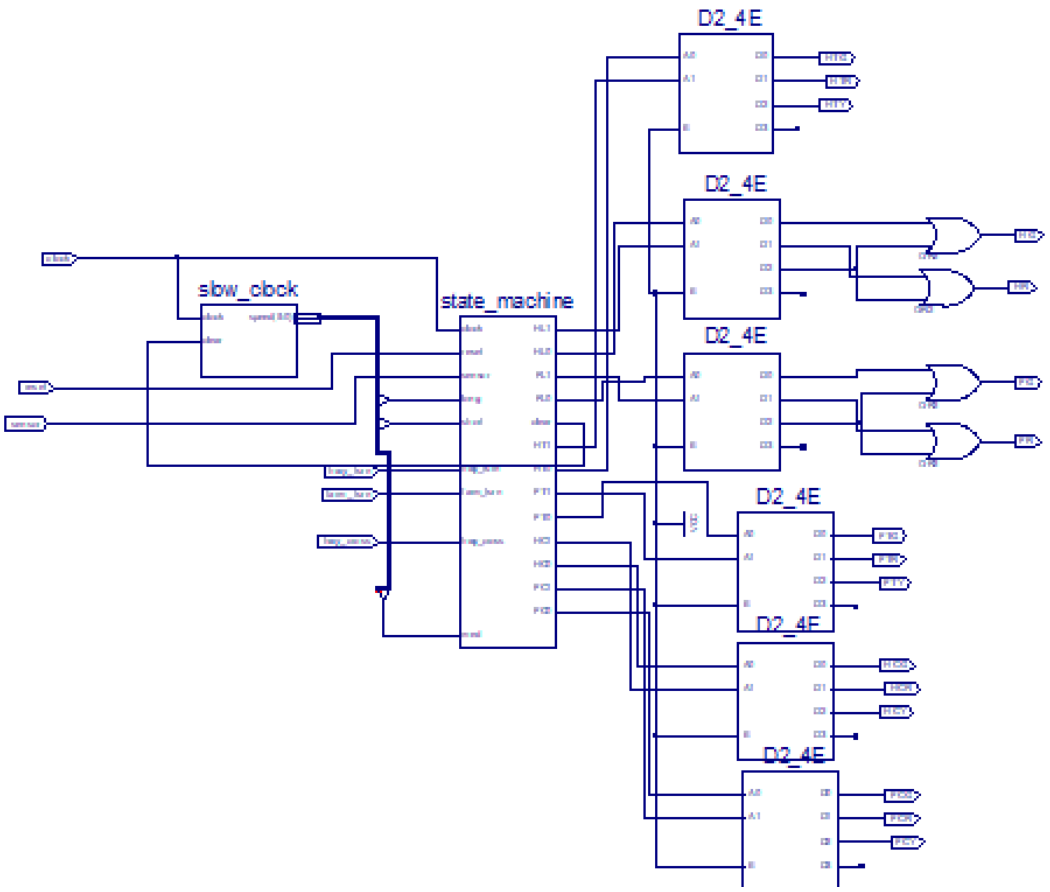

Figure 2: top.sch, the main schematic file for the project.

The slow_counter file had no changes since the original project initiation, except for the fact that the fourth bit of output was used as the new timer “med”, or medium, while only bits 3 and 5 were used originally for “short” and “long”, respectively. This new input was called “med” rather than “medium” because “medium” is a reserved keyword in the project file. slow_counter.vhd is shown below, unchanged.

```
--slow_clock.vhd begins here--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity slow_clock is
	Port ( clock, clear : in STD_LOGIC;
		speed : out STD_LOGIC_VECTOR (5 downto 0) );

end slow_clock;


architecture Behavioral of slow_clock is
	Signal cnt: STD_LOGIC_VECTOR (30 downto 0);
begin
	process(clock, clear)
		begin
			if clear='1' then cnt(30 downto 25) <= "000000";
			elsif (clock'event and clock='0') then cnt <= cnt + 1;
			end if;
	end process;
	speed <= cnt(30 downto 25);
end Behavioral;
```

Figure 3: slow_counter.vhd

The next section of the logic system is the largest as well: state_machine.vhd. This hold all the different states, inputs, and outputs, and handles the logic for the transitions between each state. Before the state_machine, the state diagram is shown below. This simply describes the overall logic between the states in a diagram form, to make it more digestible.

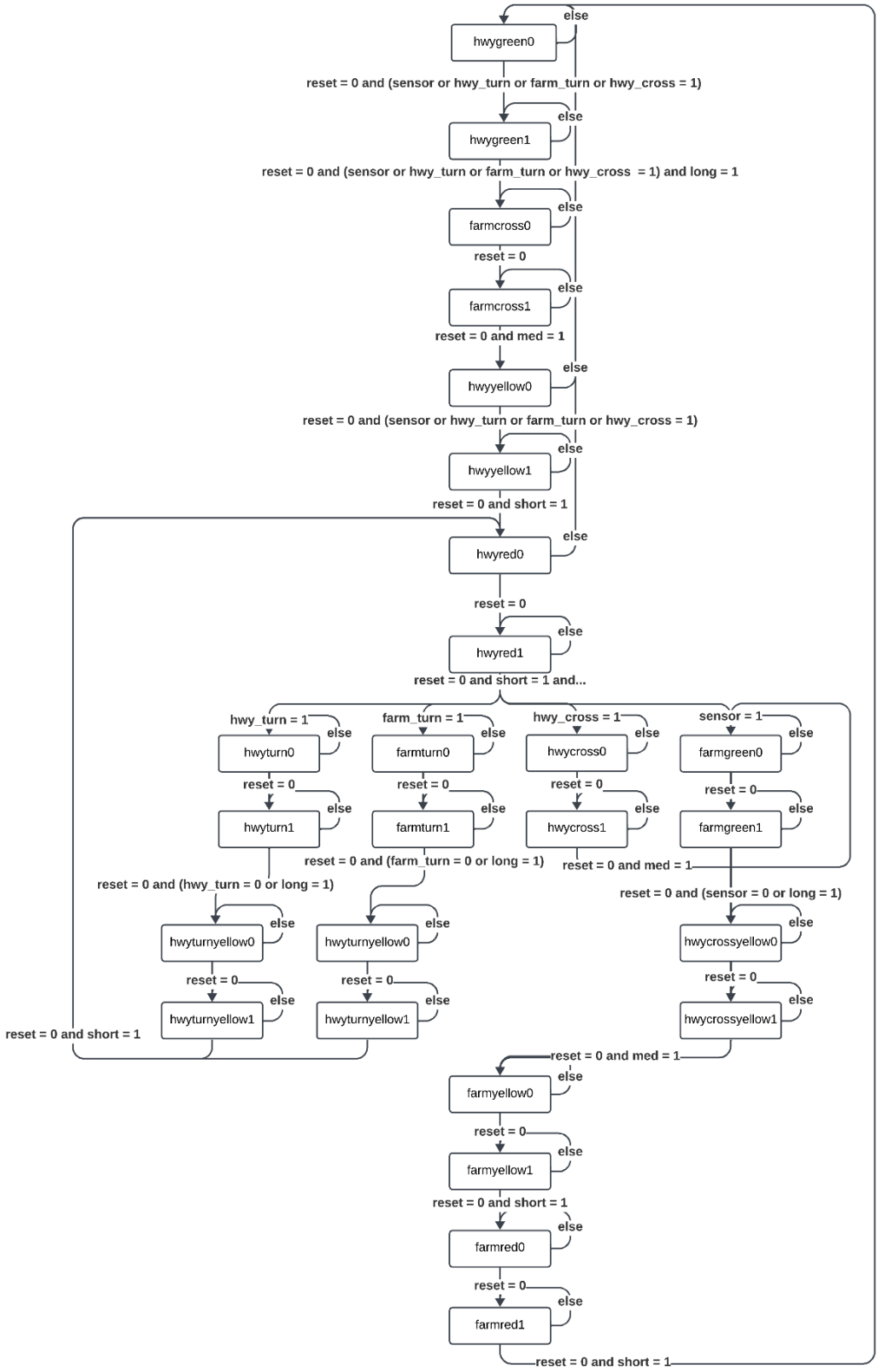

Figure 4: State diagram for the final project, made with lucid.app

state_machine.vhd follows the exact logic shown above, but in a form that can be understood and implemented in computation.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity state_machine is
	Port ( clock, reset, sensor, long, med, short, hwy_turn, farm_turn, hwy_cross : in
std_logic;
		HL1, HL0, FL1, FL0, HT1, HT0, FT1, FT0, HC1, HC0, FC1, FC0, clear : out
std_logic );

end state_machine;


architecture Behavioral of state_machine is
	type state_type is (hwygreen0, hwygreen1, hwyyellow0, hwyyellow1, hwyred0,
hwyred1, farmgreen0, farmgreen1,
		farmyellow0, farmyellow1, farmred0, farmred1, hwyturn0, hwyturn1,
hwyturnyellow0, hwyturnyellow1, farmturn0,
		farmturn1, farmturnyellow0, farmturnyellow1, farmcross0, farmcross1,
hwycross0, hwycross1, hwycrossyellow0, hwycrossyellow1);
	attribute enum_encoding: string;
	attribute enum_encoding of state_type: type is "000000000000000000000001
000000000000000000000010 000000000000000000000100 000000000000000000001000
000000000000000000010000 000000000000000000100000 000000000000000001000000
000000000000000010000000 000000000000000100000000 000000000000001000000000
000000000000010000000000 000000000000100000000000 000000000001000000000000
000000000010000000000000 000000000100000000000000 000000001000000000000000
000000010000000000000000 000000100000000000000000 000001000000000000000000
000010000000000000000000 001000000000000000000000 010000000000000000000000
100000000000000000000000";


signal CS, NS: state_type;

begin
	process (clock, reset)
	begin
		if (reset='1') then
			CS <= hwygreen0;
		elsif rising_edge(clock) then
			CS <= NS;
```

```
		end if;
	end process;
	process (reset, sensor, long, short, med, hwy_turn, farm_turn, hwy_cross, cs)
	begin
		case CS is
		when hwygreen0 =>
			HL1 <= '0'; HL0 <= '0';
			FL1 <= '0'; FL0 <= '1';
			HT1 <= '0'; HT0 <= '1';
			FT1 <= '0'; FT0 <= '1';
			HC1 <= '0'; HC0 <= '1';
			FC1 <= '0'; FC0 <= '0';
			clear <= '1';
			if (reset='0' and (sensor='1' or hwy_turn='1' or farm_turn='1' or
hwy_cross='1') ) then
				NS <= hwygreen1;
			else
				NS <= hwygreen0;
			end if;

		when hwygreen1 =>
			HL1 <= '0'; HL0 <= '0';
			FL1 <= '0'; FL0 <= '1';
			HT1 <= '0'; HT0 <= '1';
			FT1 <= '0'; FT0 <= '1';
			HC1 <= '0'; HC0 <= '1';
			FC1 <= '0'; FC0 <= '0';
			clear <= '0';
			if (reset='0' and (sensor='1' or hwy_turn='1' or farm_turn='1' or
hwy_cross='1') and long='1' ) then
				--clear <= '1'; this may have broken it
				NS <= farmcross0;
			else
				NS <= hwygreen1;
			end if;

		when farmcross0 =>
			HL1 <= '0'; HL0 <= '0';
			FL1 <= '0'; FL0 <= '1';
			HT1 <= '0'; HT0 <= '1';
			FT1 <= '0'; FT0 <= '1';
			HC1 <= '0'; HC0 <= '1';
			FC1 <= '1'; FC0 <= '0';
			clear<='1';
			if (reset='0') then
				NS <=farmcross1;
```

```
                    else
                        NS <=farmcross0;
                    end if;

                when farmcross1 =>
                    HL1 <= '0'; HL0 <= '0';
                    FL1 <= '0'; FL0 <= '1';
                    HT1 <= '0'; HT0 <= '1';
                    FT1 <= '0'; FT0 <= '1';
                    HC1 <= '0'; HC0 <= '1';
                    FC1 <= '1'; FC0 <= '0';
                    clear <= '0';
                    if(reset='0' and med='1') then
                        NS <= hwyyellow0;
                    else
                        NS <=farmcross1;
                    end if;


                when hwyyellow0 =>
                    HL1 <= '1'; HL0 <= '0';
                    FL1 <= '0'; FL0 <= '1';
                    HT1 <= '0'; HT0 <= '1';
                    FT1 <= '0'; FT0 <= '1';
                    HC1 <= '0'; HC0 <= '1';
                    FC1 <= '0'; FC0 <= '1';
                    clear <= '1';
                    if (reset='0' and (sensor='1' or hwy_turn='1' or farm_turn='1' or
hwy_cross='1')) then
                        NS <= hwyyellow1;
                    else
                        NS <= hwygreen0;
                    end if;

                when hwyyellow1 =>
                    HL1 <= '1'; HL0 <= '0';
                    FL1 <= '0'; FL0 <= '1';
                    HT1 <= '0'; HT0 <= '1';
                    FT1 <= '0'; FT0 <= '1';
                    HC1 <= '0'; HC0 <= '1';
                    FC1 <= '0'; FC0 <= '1';
                    clear <= '0';
                    if (reset='0' and short='1') then
                        NS <= hwyred0;
                    else
                        NS <= hwyyellow1;
```

```
            end if;

        when hwyred0 =>
            HL1 <= '0'; HL0 <= '1';
            FL1 <= '0'; FL0 <= '1';
            HT1 <= '0'; HT0 <= '1';
            FT1 <= '0'; FT0 <= '1';
            HC1 <= '0'; HC0 <= '1';
            FC1 <= '0'; FC0 <= '1';
            clear <= '1';
            if (reset='0') then
                NS <= hwyred1;
            else
                NS <= hwygreen0;
            end if;

        when hwyred1 =>
            HL1 <= '0'; HL0 <= '1';
            FL1 <= '0'; FL0 <= '1';
            HT1 <= '0'; HT0 <= '1';
            FT1 <= '0'; FT0 <= '1';
            HC1 <= '0'; HC0 <= '1';
            FC1 <= '0'; FC0 <= '1';
            clear <= '0';
            if (reset='0' and short='1' and hwy_turn='1') then
                NS <= hwyturn0;
            elsif (reset='0' and short='1' and farm_turn='1') then
                NS <= farmturn0;
            elsif (reset='0' and short='1' and hwy_cross='1') then
                NS <= hwycross0;
            elsif (reset='0' and short='1') then
                NS <= farmgreen0;
            else
                NS <= hwyred1;
            end if;

        when hwycross0 =>
            HL1 <= '0'; HL0 <= '1';
            FL1 <= '0'; FL0 <= '0';
            HT1 <= '0'; HT0 <= '1';
            FT1 <= '0'; FT0 <= '1';
            HC1 <= '0'; HC0 <= '0';
            FC1 <= '0'; FC0 <= '1';
            clear<='1';
            if(reset='0') then
                NS <= hwycross1;
```

```
		else
			NS <= hwycross0;
		end if;

	when hwycross1 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '0';
		HT1 <= '0'; HT0 <= '1';
		FT1 <= '0'; FT0 <= '1';
		HC1 <= '0'; HC0 <= '0';
		FC1 <= '0'; FC0 <= '1';
		clear<='0';
		if(reset='0' and med='1') then
			NS <= farmgreen0;
		else
			NS <= hwycross1;
		end if;

	when farmturn0 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '1';
		HT1 <= '0'; HT0 <= '1';
		FT1 <= '0'; FT0 <= '0';
		HC1 <= '0'; HC0 <= '1';
		FC1 <= '0'; FC0 <= '1';
		clear <= '1';
		if (reset='0') then
			NS <= farmturn1;
		else
			NS <= farmturn0;
		end if;

	when farmturn1 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '1';
		HT1 <= '0'; HT0 <= '1';
		FT1 <= '0'; FT0 <= '0';
		HC1 <= '0'; HC0 <= '1';
		FC1 <= '0'; FC0 <= '1';
		clear <= '0';
		if (reset='0' and ( farm_turn='0' or long='1' ) ) then
			NS <= farmturnyellow0;
		else
			NS <= farmturn1;
		end if;
```

```
when farmturnyellow0 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '0'; FL0 <= '1';
        HT1 <= '0'; HT0 <= '1';
        FT1 <= '1'; FT0 <= '0';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '1';
        if (reset='0') then
                NS <= farmturnyellow1;
        else
                NS <= farmturnyellow0;
        end if;

when farmturnyellow1 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '0'; FL0 <= '1';
        HT1 <= '0'; HT0 <= '1';
        FT1 <= '1'; FT0 <= '0';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '0';
        if (reset='0' and short='1') then
                NS <= hwyred0;
        else
                NS <= farmturnyellow1;
        end if;

when hwyturn0 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '0'; FL0 <= '1';
        HT1 <= '0'; HT0 <= '0';
        FT1 <= '0'; FT0 <= '1';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '1';
        if (reset='0') then
                NS <= hwyturn1;
        else
                NS <= hwyturn0;
        end if;

when hwyturn1 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '0'; FL0 <= '1';
        HT1 <= '0'; HT0 <= '0';
```

```
		FT1 <= '0'; FT0 <= '1';
		HC1 <= '0'; HC0 <= '1';
		FC1 <= '0'; FC0 <= '1';
		clear <= '0';
		if (reset='0' and ( hwy_turn='0' or long='1' ) ) then
			NS <= hwyturnyellow0;
		else
			NS <= hwyturn1;
		end if;

	when hwyturnyellow0 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '1';
		HT1 <= '1'; HT0 <= '0';
		FT1 <= '0'; FT0 <= '1';
		HC1 <= '0'; HC0 <= '1';
		FC1 <= '0'; FC0 <= '1';
		clear <= '1';
		if (reset='0') then
			NS <= hwyturnyellow1;
		else
			NS <= hwyturnyellow0;
		end if;

	when hwyturnyellow1 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '1';
		HT1 <= '1'; HT0 <= '0';
		FT1 <= '0'; FT0 <= '1';
		HC1 <= '0'; HC0 <= '1';
		FC1 <= '0'; FC0 <= '1';
		clear <= '0';
		if (reset='0' and short='1') then
			NS <= hwyred0;
		else
			NS <= hwyturnyellow1;
		end if;

	when farmgreen0 =>
		HL1 <= '0'; HL0 <= '1';
		FL1 <= '0'; FL0 <= '0';
		HT1 <= '0'; HT0 <= '1';
		FT1 <= '0'; FT0 <= '1';
		HC1 <= '0'; HC0 <= '0';
		FC1 <= '0'; FC0 <= '1';
		clear <= '1';
```

```
                if (reset='0') then
                        NS <= farmgreen1;
                else
                        NS <= hwygreen0;
                end if;

        when farmgreen1 =>
                HL1 <= '0'; HL0 <= '1';
                FL1 <= '0'; FL0 <= '0';
                HT1 <= '0'; HT0 <= '1';
                FT1 <= '0'; FT0 <= '1';
                HC1 <= '0'; HC0 <= '0';
                FC1 <= '0'; FC0 <= '1';
                clear <= '0';
                if (reset='0' and ( sensor='0' or long='1' ) ) then
                        clear <='1';
                        NS <= hwycrossyellow0;
                else
                        NS <= farmgreen1;
                end if;

        when hwycrossyellow0 =>
                HL1 <= '0'; HL0 <= '1';
                FL1 <= '0'; FL0 <= '0';
                HT1 <= '0'; HT0 <= '1';
                FT1 <= '0'; FT0 <= '1';
                HC1 <= '1'; HC0 <= '0';
                FC1 <= '0'; FC0 <= '1';
                clear<='1';
                if(reset='0') then
                        NS <= hwycrossyellow1;
                else
                        NS <= hwycrossyellow0;
                end if;

        when hwycrossyellow1 =>
                HL1 <= '0'; HL0 <= '1';
                FL1 <= '0'; FL0 <= '0';
                HT1 <= '0'; HT0 <= '1';
                FT1 <= '0'; FT0 <= '1';
                HC1 <= '1'; HC0 <= '0';
                FC1 <= '0'; FC0 <= '1';
                clear<='0';
                if(reset='0' and med='1') then
                        NS <= farmyellow0;
                else
```

```
                NS <=hwycrossyellow1;
        end if;

when farmyellow0 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '1'; FL0 <= '0';
        HT1 <= '0'; HT0 <= '1';
        FT1 <= '0'; FT0 <= '1';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '1';
        if (reset='0') then
                NS <= farmyellow1;
        else
                NS <= hwygreen0;
        end if;

when farmyellow1 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '1'; FL0 <= '0';
        HT1 <= '0'; HT0 <= '1';
        FT1 <= '0'; FT0 <= '1';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '0';
        if (reset='0' and short='1') then
                NS <= farmred0;
        else
                NS <= farmyellow1;
        end if;

when farmred0 =>
        HL1 <= '0'; HL0 <= '1';
        FL1 <= '1'; FL0 <= '0';
        HT1 <= '0'; HT0 <= '1';
        FT1 <= '0'; FT0 <= '1';
        HC1 <= '0'; HC0 <= '1';
        FC1 <= '0'; FC0 <= '1';
        clear <= '1';
        if (reset='0') then
                NS <= farmred1;
        else
                NS <= hwygreen0;
        end if;

when farmred1 =>
```

```
                        HL1 <= '0'; HL0 <= '1';
                        FL1 <= '0'; FL0 <= '1';
                        HT1 <= '0'; HT0 <= '1';
                        FT1 <= '0'; FT0 <= '1';
                        HC1 <= '0'; HC0 <= '1';
                        FC1 <= '0'; FC0 <= '1';
                        clear <= '0';
                        if (reset='0' and short='1') then
                                NS <= hwygreen0;
                        else
                                NS <= farmred1;
                        end if;

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

Figure 5: state_machine.vhd

Finally, the inputs and outputs that are sent into to top.sch are routed to pins on the FPGA board through the top.ucf file. Each input is sent to a switch (save the reset button), and each output is sent to an LED, with the highway and farm road lights being sent to the tri-color LEDs. The file top.ucf is shown below.

```
#Miscellaneous
NET clock LOC=E3 | IOSTANDARD=LVCMOS33; #clock
NET reset LOC=P18 | IOSTANDARD=LVCMOS33; #BTND

#Highway Lights
NET HR LOC=N16 | IOSTANDARD=LVCMOS33; #LD17, red
NET HG LOC=R11 | IOSTANDARD=LVCMOS33; #LD17, green

#Highway Turn
NET HTG LOC=V11 | IOSTANDARD=LVCMOS33; #LD15
NET HTY LOC=V12 | IOSTANDARD=LVCMOS33; #LD14
NET HTR LOC=V14 | IOSTANDARD=LVCMOS33; #LD13

#Highway Crosswalk
NET HCG LOC=V15 | IOSTANDARD=LVCMOS33; #LD12
NET HCY LOC=T16 | IOSTANDARD=LVCMOS33; #LD11
NET HCR LOC=U14 | IOSTANDARD=LVCMOS33; #LD10
```

```
#Farmroad Lights
NET FR LOC=N15 | IOSTANDARD=LVCMOS33; #LD16, red
NET FG LOC=M16 | IOSTANDARD=LVCMOS33; #LD16, green

#Farmroad Turn
NET FTG LOC=V17 | IOSTANDARD=LVCMOS33; #LD5
NET FTY LOC=R18 | IOSTANDARD=LVCMOS33; #LD4
NET FTR LOC=N14 | IOSTANDARD=LVCMOS33; #LD3

#Farmroad Crosswalk
NET FCG LOC=J13 | IOSTANDARD=LVCMOS33; #LD2
NET FCY LOC=K15 | IOSTANDARD=LVCMOS33; #LD1
NET FCR LOC=H17 | IOSTANDARD=LVCMOS33; #LD0

#Highway inputs
NET hwy_turn LOC=V10 | IOSTANDARD=LVCMOS33; #SW15
NET hwy_cross LOC=U11 | IOSTANDARD=LVCMOS33; #SW14

#Farmroad inputs
NET farm_turn LOC=L16 | IOSTANDARD=LVCMOS33; #SW1
NET sensor LOC=J15 | IOSTANDARD=LVCMOS33; #SW0
```

Figure 6: top.ucf for the final project

It is worth noting that there is no "yellow" input for the tri-color LEDs but sending logic 1 to both red and green creates a yellow light. Therefore, when either light is intended to be yellow, both output signals for the tri-color LEDs are utilized. Below is the pin assignment table for the project.

Table 1: Pin assignment table for final project

Functionality	Input/Output	FPGA Pin	Peripheral
-	clock	E3	Internal clock
-	reset	P18	BTND
Highway Red	HR	N16	LD17: red
Highway Green	HG	R11	LD17: green
Highway Left Turn Green	HTG	V11	LD15
Highway Left Turn Yellow	HTY	V12	LD14
Highway Left Turn Red	HTR	V14	LD13
Highway Crosswalk Green	HCG	V15	LD12
Highway Crosswalk Yellow	HCY	T16	LD11

Highway Crosswalk Red	HCR	U14	LD10
Farmroad Red	FR	N15	LD16: red
Farmroad Green	FG	M16	LD16: green
Farmroad Left Turn Green	FTG	V17	LD5
Farmroad Left Turn Yellow	FTY	R18	LD4
Farmroad Left Turn Red	FTR	N14	LD3
Farmroad Crosswalk Green	FCG	J13	LD2
Farmroad Crosswalk Yellow	FCY	K15	LD1
Farmroad Crosswalk Red	FCR	H17	LD0
Highway Left Turn Sensor	hwy_turn	V10	SW15
Highway Crosswalk Request	hwy_cross	U11	SW14
Farmroad Left Turn Sensor	farm_turn	L16	SW1
Farmroad Straight Sensor	sensor	J15	SW0

c. Testing Procedure

The testing procedure for this project was simple, following a checklist for each new state cycle. An example of a testing procedure is shown below.

Testing Farmroad Left Turn

- Starts at base state (HG, FR, FCG, HCR, HTR, FTR)
- After time [LONG], moves to crosswalk yellow (HG, FR, FCY, HCR, HTR, FTR)
- After time [MED], moves to highway yellow (HY, FR, FCR, HCR, HTR, FTR)
- After time [SHORT], moves to full red state (HR, FR, FCR, HCR, HTR, FTR)
  (General state testing ends here: rest is for Farmroad Left Turn only)
- Farmroad left turn green state occurs (HR, FR, FCR, HCR, HTR, FTG)
- After time [LONG] or sensor turns off, Farmroad left turn yellow (HR, FR, FCR, HCR, HTR, FTY)
- After time [SHORT], full red state (HR, FR, HCR, HCR, HTR, FTR)
- Back to base state (HG, FR, FCG, HCR, HTR, FTR)

Figure 7: Farmroad left turn testing checklist.

For most new states, the first four checkboxes are the same. Essentially, this testing procedure just provides a rigorous confirmation that the state diagram and state machine are aligned. If a state is being used incorrectly, notes must be taken, and changes must be made.

III. Cost and Size Analysis

In general, this didn't add too many new components to the traffic light. In our case, we used fewer LEDs to indicate the traffic lights for the highway and farm road, although in a realistic scenario, 3 separate lights would be used anyways. Then, 3 lights were added on each side for the protect left signals, and 6 LEDs on the FPGA board or 4 light panels in real life are added in each direction to constitute the crosswalks. This cost does add up but is also necessary for safety.

On the FPGA board, 3 new sensors were added, where the real-life equivalent is 4 buttons on the cross-highway crosswalks and 4 car sensors in the roads for the left turn lanes.

Inside the logic system, 4 new 2-4 decoders were added to the block, so the cost of these units needs to be considered as well.

In total, on the FPGA board, 12 LEDs were added and 4 were taken away, with 2 staying, bringing the total of LEDs used on the FPGA board to be 14, over double the original 6, but for triple the functionality. Then, 4 two-to-four decoders were added in the logic system, along with the logic switches mentioned earlier, being 3 total on the board.

In total, in real life, 12 lights were added for the protected left lanes, and 8 LED panels for the crosswalks. The buttons to initiate the cross across the crosswalk would be fairly cheap, however the car sensors in the road for the left turn lanes would be quite expensive.

IV. Status and Retrospective

At this point, the traffic light works as initially intended. The crosswalks and turn signals have logical flow and the tri-color LEDs almost work as wanted. The only issue with the LEDs is the brightness, but to fix that, the duty cycle for the LEDs must be adjusted, which is above our expertise at this time. Additionally, states for Emergency Vehicle Override and Hardware Failure fallback were initially intended to be added to the logic system, however the risk of adding these new states in contrast with the minimal amount of additional complexity it brings made it seem ill-advised to continue to pursue this goal.

One thing to consider when transferring this system into a real world scenario is the fact that the sensors for the farmroad and turning lanes would have some time where they don't sense anything, as cars are moving across it. For example, if there are 2 cars waiting to cross through the farmroad, the sensor would show 1, but as the first car leaves, the second car may not have reached the sensor, leading to a 0 on the sensor, and the light to instantly turn yellow. So, a buffer needs to be added here, and on the left turn lanes.

In all, the traffic light system is significantly more robust than its original status, and the added complexity brings the traffic light system to be more akin to that of a real life traffic light.

V. Reference

1. The laboratory manual.
2. Nexys A7 FPGA board reference manual.

VI. Equipment

1. Nexys A7 FPGA board
2. lucid.app