Andrew Dority
EECS 443 Final Project
5/3/2023

Traffic Light Controller

A finite state machine was created to model a traffic light controller. (Note: to keep the diagram simple, transitions due to the reset signal are not present)

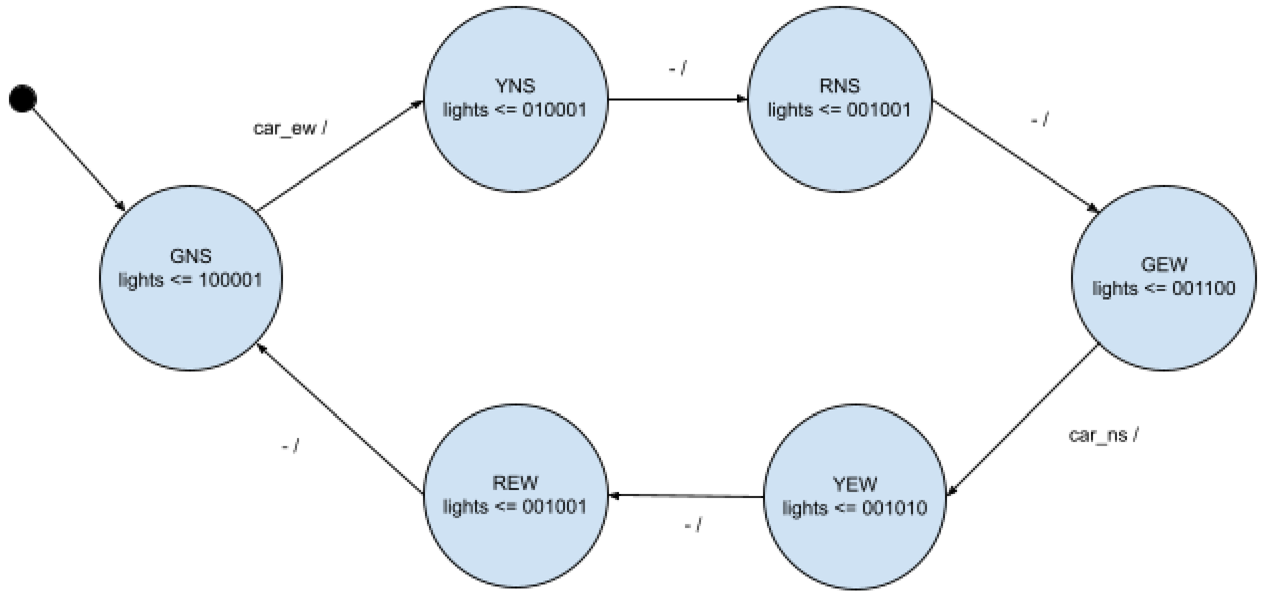

Fig. 1: Moore FSM Diagram

To keep the next-state logic combinational, there is not a separate timer. This means that the lights will switch from green to yellow to red (for their respective direction) as quick as the clock can tick. Therefore, the clock signal's period must be reasonably large enough to give drivers time to stop before a yellow light turns red.

Below is a hardware block diagram (next page).

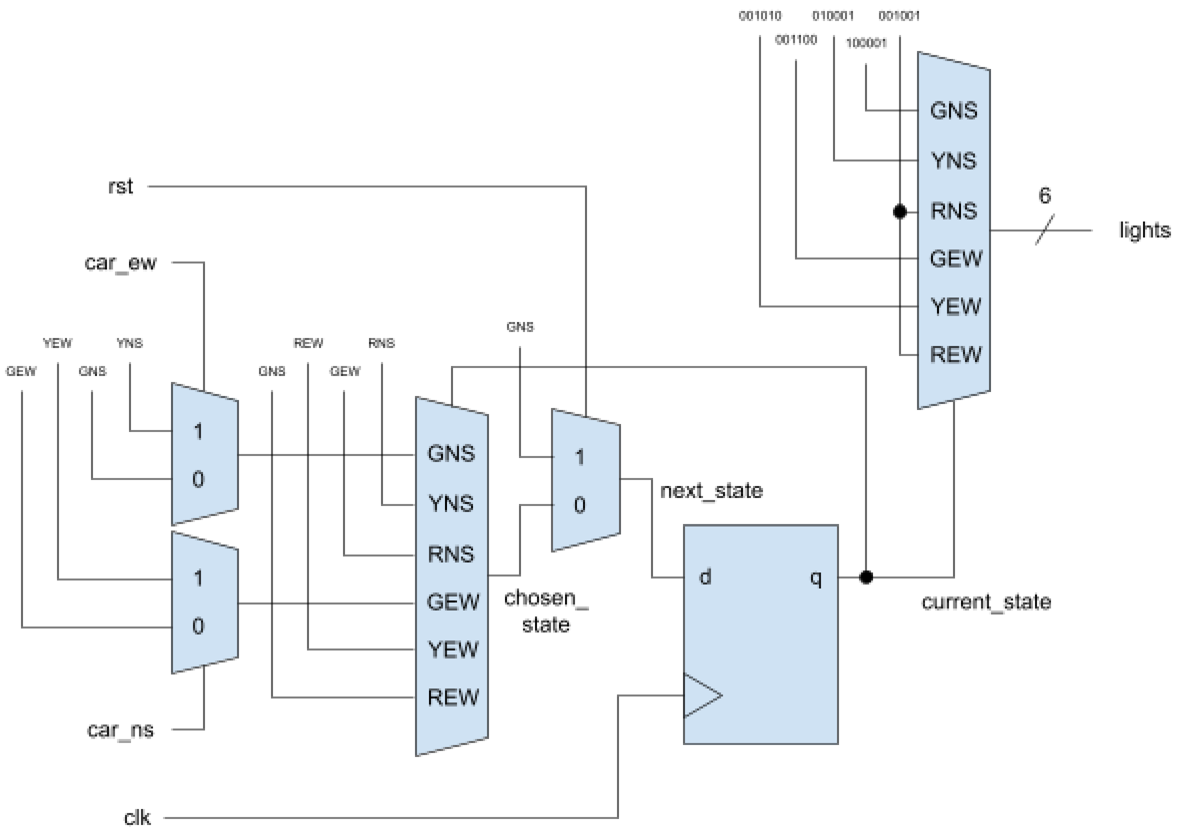

Fig. 2: Hardware Diagram

The VHDL code implementation is below. Note that a clock divider and a count signal were used to make the state change every 5 seconds. Another change (not included here) was the outputs for YNS and YEW, changed to "101001" and "001101". The point of this change is to make a yellow light a combination of red and green instead of having its own bit. This way, the bits can be assigned to RGB lights, and when the yellow lights should come on, the combination of red and green makes the RGB light turn yellow. This was not included in the final code as real traffic lights aren't RGB.

```
entity traffic_light_controller is
port (
        car_ew, car_ns, clk, rst: in std_logic;
        lights: out std_logic_vector(5 downto 0)
);
end traffic_light_controller;
```

```
architecture arch of traffic_light_controller is
        type traf_state is
        (GNS, YNS, RNS, GEW, YEW, REW);
        signal current_state, next_state: traf_state;
        signal chosen_state: traf_state;
        signal GNS_next, GEW_next: traf_state;
        signal slow_clk: std_logic;
        signal count: std_logic_vector(3 downto 0) := (others => '0');

        component clock_divider is
        generic(f_in : natural := 100E6; --fin = 100MHz - FPGA's clock frequency
        f_out: natural := 1   --fout = 1Hz
        );
        port(
        clk: in std_logic;
        f_o: out std_logic
        );
end component;
begin
        -- clock divider
        CD: clock_divider port map(clk => clk, f_o => slow_clk);
        -- d FF
        dFF: process(slow_clk)
        begin
        if (rising_edge(slow_clk)) then
        if (count = "0101") then
        count <= "0000";
        current_state <= next_state;
        else
        count <= count+1;
        end if;
        end if;
        end process;
        -- Next State Logic
        with rst select
        next_state <= GNS when '1',
                chosen_state when others;
        with car_ew select
        GNS_next <= YNS when '1',
                GNS when others;
        with car_ns select
        GEW_next <= YEW when '1',
                GEW when others;
        with current_state select
        chosen_state <= GNS_next when GNS,
                RNS when YNS,
                GEW when RNS,
                GEW_next when GEW,
                REW when YEW,
                GNS when others;
        -- Output Logic (Moore)
        with current_state select
        lights <= "100001" when GNS,
                "010001" when YNS, -- "101001" for RGB
                "001100" when GEW,
                "001010" when YEW, -- "001101" for RGB
                "001001" when others;
end arch;
```

Fig. 3: VHDL Code Implementation

The behavior of the circuit is almost exactly as expected. The only unexpected result is that the clock signal's period varied slightly (± ~0.5s*). This means that testing the board at a low period did not guarantee that all state outputs would be seen, as some would switch through too quickly. Since only one timer (the clock signal) is used, the wait time between all states is the same. Since the clock is not dependent on the state or the state change of the circuit, if a car is detected right before the next rising edge of the clock, the other light might switch to yellow almost instantly. A better design might be to start a timer whenever a car is detected so that the time between when a car is detected and the lights switching is constant. A better design might also use a timer to ensure a green light stays green for a longer period of time. That way, on a busy street, the lights won't change as soon as the clock lets them, and allow greater throughput.