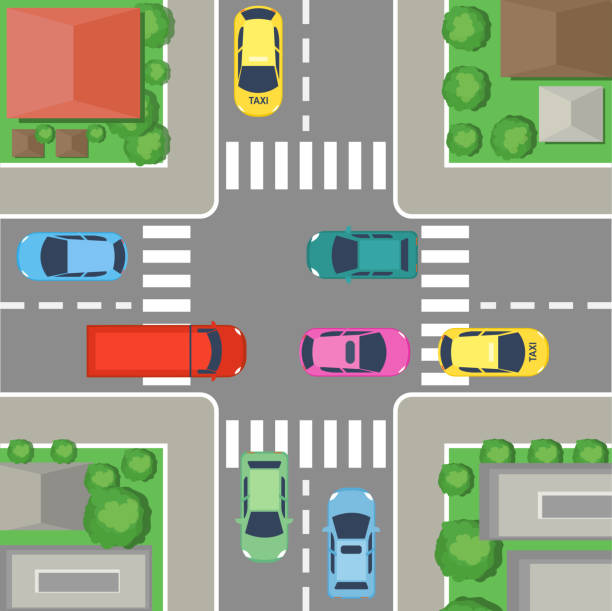
Lab 3 CSE140L

**Traffic Signal**

**Part 1**

1.1 In the H&H textbook, a conventional traffic light controls the right-of-way at two intersecting streets. (This is a typical right angle four-way intersection, with both streets going through.)

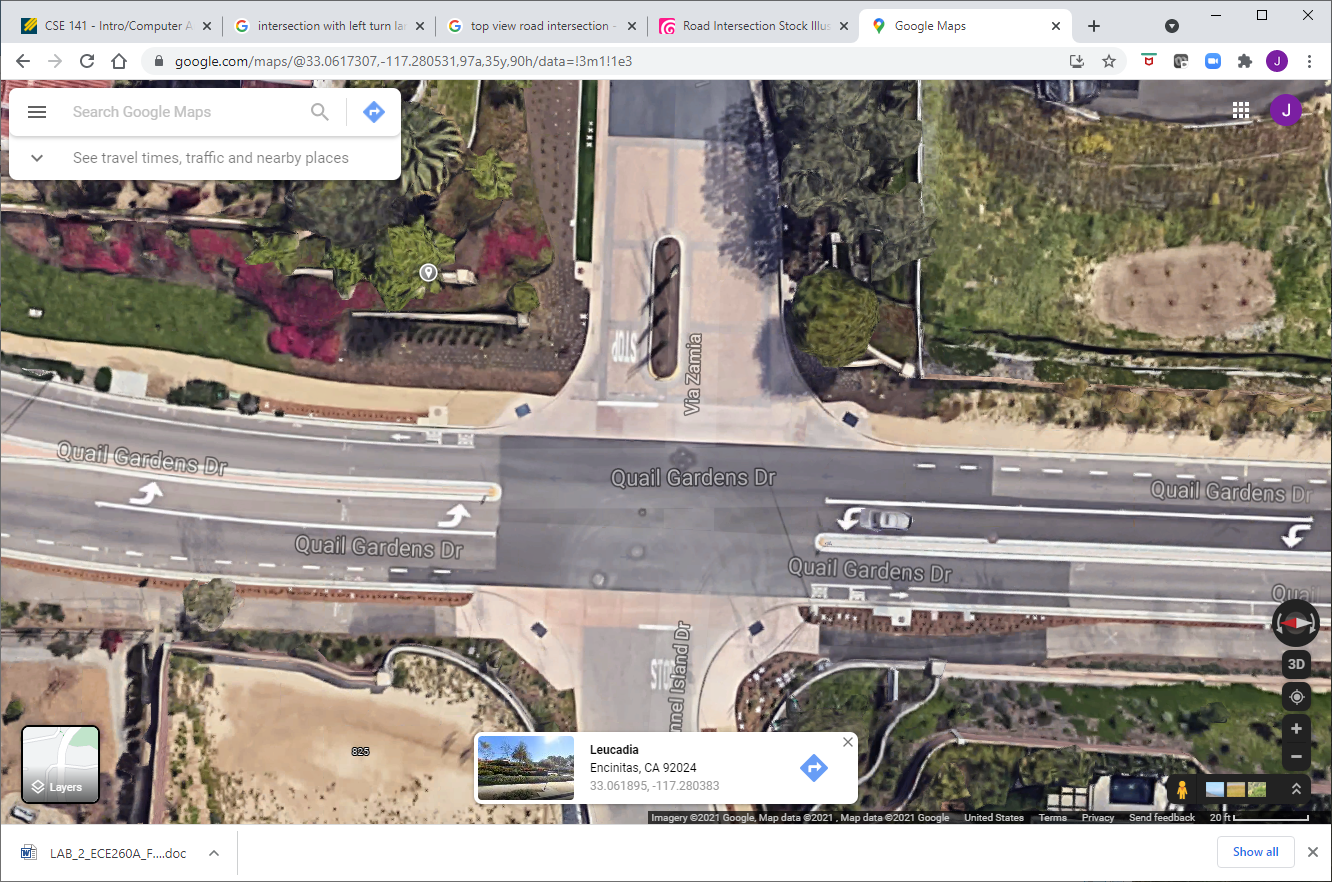


1.2 Each light has three colors: Green, Yellow, and Red. After a green light, the light will **always** turn yellow for 2 clock cycles before turning red, and the all-red (clear intersection).

1.2a The colors are encoded as follows: Green = 2, Yellow = 1, and Red = 0

(Note: Harris & Harris use Green, Yellow, Red = 0, 1, 2, instead)

1.3 In our lab assignment, each direction on the east-west (EW) street (Quail Gardens Drive in this satellite view stolen from Google Maps and rotated 90 degrees to fit our problem definition) contains a straight lane and a separate left turn lane. Each left turn lane is **protected**, i.e., it has a dedicated left arrow turn signal.



1.4 Both directions on the north-south (NS) street contain only straight lanes. If a car wants to make a left turn, it must make an **unprotected left turn**. Thus, there are two traffic signals (left turn arrow and straight green light) facing each direction on the main (EW) street, but only one set of lights facing each direction of the minor (NS) street.

1.5 The lights for the east and west directions are symmetric, meaning that the west direction will never have both its turn signal and straight signal green at the same time, with the east direction all red. Either both east and west straight lanes are green, or both east and west left turn lanes are green and the straight lanes are red, to prevent a left-cross collision. The north and south direction lights are identical to each other.

Allowable states:

|  |  |  |
| --- | --- | --- |
| EW straight = EWS = S | EW left arrow = EWL = L | NS (straight or unprotected left) = N |
| Red | Red | Red |
| Green | Red | Red |
| Yellow | Red | Red |
| Red | Green | Red |
| Red | Yellow | Red |
| Red | Red | Green |
| Red | Red | Yellow |

1.6 All lanes have traffic sensors that can trigger a change in the lights. The EW lanes have separate traffic sensors for cars going straight and cars in either of the left turn lanes. The NS lanes have sensors which do not distinguish between going straight or turning left (there are no dedicated protected left turn lanes).

1.7 The starting state of the intersection is all red lights (all-red phase) for EWL, EWS, and NS signals, with EWS having the highest priority for the next green. Absent traffic, the system will always eventually return to all red, with green change priority depending on the most recent green-yellow sequence.

1.8 TIMERS: If the signals are all-red and traffic is detected on one or more sensors, the corresponding light turns green and stays green until either the 5-counter or the 10-counter (more on those follows) reaches a limit, whichever comes first.

The 5 cycle counter starts when a break in same-direction traffic is detected, and it continues to count, even if same-direction traffic resumes. When the count reaches 5, the light turns yellow for 2 clock cycles and then red. As long as traffic is detected continuously (no breaks) and on this sensor only, it will remain green, and this counter will remain at 0.

The 10-cycle counter starts when conflicting traffic is detected in one or more directions. When it reaches its maximum, the light will turn yellow for two cycles, then red.

Once there is a break in green-direction traffic, the 5-cycle countdown begins, even if traffic reappears during the countdown. Likewise, the 10-cycle countdown begins as soon as conflicting traffic appears. If same-direction traffic goes away during the 10-cycle countdown, the 5-cycle countdown commences and may trigger an earlier transition to yellow.

1.8 RED TO GREEN: From the initial/reset/starting all-red phase, if traffic appears simultaneously at multiple sensors, the precedence is as follows:

EWS > EWL > NS

For example, if there is a car on the EWS sensor, the EWS signal will become green, regardless of the other lanes. If there is traffic on both the EWL and the NS simultaneously and none on EWS, the EWL will become green first.

1.8a “Fairness doctrine”:The green light precedence rotates according to which direction most recent had a green-yellow sequence. Thus, if EWS just had a turn, then the precedence becomes: EWL > NS > EWS. We can get two EWS green-yellow-red cycles in a row, but only absent traffic in the other two directions.

If the EWL most recently had a green-yellow cycle, then NS has top priority, and EWL has lowest priority. Following an NS green-yellow, EWS again has top priority, followed by EWL and then NS. HINT: As mentioned in class, you probably want three separate all-red control states, one for each "from yellow" scenario.

1.8b If all sensors always have traffic, then the precedence is the same. That is, the intersection will cycle between EWS green for 10 clocks, then yellow for 2 clocks, then all red for 1, then EWL green for 10 clocks, then yellow for 2, then all red for 1, then NS green for 10 and yellow for 2 and all red for 1, before EWS becomes green again.

1.8c **You have two decisions to make: How long a given light stays green before turning yellow, and which green light follows a given all-red phase.**

**PART 2**

With the similar setting of **PART 1**, let's decouple the left turn and through greens for our EW streets. We can now accommodate five different green light states, in this initial order of descending priority:

eastbound straight and westbound straight = ES & WS = S

eastbound left and eastbound straight = EL & ES = E

westbound left and westbound straight = WL & WS = W

eastbound left and westbound left = EL & WL = L

northbound and southbound = NS = N

These priorities will rotate round-robin, just as under our 3-state scenario. Thus, after S has received a green and a yellow, the priority is now:

eastbound left and eastbound straight = EL & ES = E

westbound left and westbound straight = WL & WS = W

eastbound left and westbound left = EL & WL = L

northbound and southbound = NS = N

eastbound straight and westbound straight = ES & WS = S

HINT: I solved this with 20 states: one green, two yellows, one all-red for each of the five possible directonal scenarios.

**Overall rules**:

1) A given green will stay that way, absent any conflicting traffic, as long as same-direction traffic is present.

2) An existing green will go into countdown if it no longer has traffic demand itself. It will count for 5 clocks, then turn yellow for 2. If it has continuous traffic demand, it counts for 10 clocks, then turns yellow for 2 as soon as conflicting traffic appears, halting traffic in its own direction to give others a chance. If conflicting traffic is present before the 10 clocks, the light turns yellow immediately after the 10 clocks.

3) Yellow phases are always two clocks in duration, and are always followed by an all-red phase.

Hint: When going into all-red, you may wish to keep track of your most recent green state, so that you can rotate turns and be fair to all directions. It is very easy to lock out NS, for example.

**Your mission**

3.1 Design state machines which controls the three traffic signals (for **PART1** and **PART2**). You may design three separate state machines, one for the EWS, one for EWL, and one for NS, but if you take this approach, you will, of course, have to figure out how to make them communicate and synchronize with each other. You may instead choose to build a single state machine. Either approach is acceptable, provided that you meet the two prime directives of safety (no conflicting traffic light permissions, so no more than one green or yellow at a time, per table above) and fairness (NS in particular does not get locked out of a turn at green).

3.2 Figure out what states you need and allocate enough bits to contain them in your clocked state register. How do you want to handle the timing on green and yellow? (Counters? Multiple replicates of states? Either is acceptable.)

3.3 Write out the next state transition(s) from each state and figure out the logic for these.

3.4 Use the attached top-level shell and test bench.

3.5 Turn in for **both** parts: (**see submission instructions**)

**3.5.1 a full set of working Verilog code, including test bench;**

**3.5.2 screenshots of your transcript, or your output file (print to screen);**

**3.5.3 screenshot of your waveform viewer, showing the presence of traffic and the states of the traffic signals. (Note that my testbench includes a typedef enum {green, yellow, red} to display names of colors instead of just 2, 1, 0, for user convenience.)**

**3.5.4 a very brief summary -- did everything work as expected? If not, what worked / what didn't? Anything the TAs or I need to know to read and run your code easily and effficiently?**