Lab 3 Fall 2020 CSE140L

**Traffic Signal**

**Part 1**

1.1 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

1.3 Each direction on the east-west (EW) street 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**.

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 directions are also symmetric.

Allowable states:

|  |  |  |
| --- | --- | --- |
| EW straight = EWS | EW left arrow = EWL | NS (straight or unprotected left) |
| 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 any 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. Absent traffic, the system will always eventually return to this state.

1.8 If the signals are all-red and traffic is detected on one and only one sensor, the corresponding light turns green and stays green until 5 clock cycles after traffic is no longer detected, at which point it turns yellow for 2 clock cycles and then red. As long as traffic is detected on this sensor only, it will remain green.

However, it will turn yellow if it has been green for at least 10 clock cycles and traffic is detected on one or both of the other sensors. Further clarification: think of 5 clock cycles as a minimum green duration (car detected for only one clock pulse) and 10 clock cycles as a maximum, even if traffic remains present, to give cross traffic a fair chance. (Maximum green time is unlimited if traffic remains present in a given direction, with none in the other directions.)

1.9 From the 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, the EWL will become green first.

1.a 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.

**PART 2**

2.1 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:

westbound left and westbound straight = WL & WS

eastbound left and eastbound straight = EL & ES

eastbound straight and westbound straight = ES & WS

eastbound left and westbound left = EL & WL

northbound and southbound = NS

Let’s make EL&ES, WL&WS and EL&WL as left turns. Generally, we want to prioritize straights (ES&WS) over left turns over NS. The priority is similar to the following graph: Diagram

Description automatically generated

Thus, the general order will be firstly ES&WS, and then one of the left turns, and finally NS. When going from ES&WS, the priority of the left turns will be EL&WL over EL&ES over WL&WS; otherwise, the priority of the left turns will be EL&ES over WL&WS over EL&WL.

Augment the test bench to try out the various cases listed. When changing traffic light states, you may find it simplest go through an all-red state, rather than keeping, for example, ES green while changing from EL green to EL yellow to EL red to WS green. You do have some flexibility in this interpretation, provided that you give everyone a turn and, of course, don’t allow conflicting traffic movements.

Overall rules:

1) A given green will stay that way, absent any conflicting traffic.

2) An existing green will go into countdown. If it no longer has traffic demand itself, it counts for 5 clocks, then turns yellow for 2. If it still has traffic demand, it counts for 10 clocks, then turns yellow for 2 as soon as conflicting traffic appears, potentially 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.

Hint: You may want to do a round-robin on priorities. Thus, following ES or WS, you may want to demote these to the bottom of the priority stack. Following EL or WL, you may want NS to bubble to the top of the priority stack.

**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?**