COL719: Synthesis of Digital Systems
Lab Assignment 1, Part 3

Varun Desai, 2018EE10511

Description of the Logic

As mentioned in the problem statement, the design is split up into three parts. The display unit, the sensor unit, and the adaptation unit. I describe all the three parts here in detail, as the display unit from the first submission has undergone considerable amount of changes.

Part 1: The Display and Timer Unit

The display and timer unit is comprises of a finite state machine which controls which light is green at a given instant, and drives the inputs for the seven segment displays. There also exist module to convert the binary values provided by the finite state machine to the correct driving signals for LEDs on a seven segment display.

It is assumed that the entire circuit runs at a high frequency and thus a module named seconds_clock is created which can be configured on the basis of the clock frequency and signals whenever a second has elapsed. All the transitions of the FSM occur when a second has elapsed.

The FSM has 8 states namely `{IDLE, N, S, E, W, O, O_ped, R_ped}`. The states `N, S, E, W` are the states which corresponding to the green lights for the N, S, E, W directions. `O` is when the light is orange. `O_ped` is also when the light is orange(for one particular direction), `R_ped` is a state where all the signals are red, which is held for a configurable amount of time, which can be specified by T_{ped}. `O_ped` is different from `R_ped` in the sense that the FSM directly jumps to `R_ped` when it is in state `O_ped` for T_o time. When it is state `O`, it goes to one of `N, S, E, W`, depending on what was the previous state (these are stored in variables `prev_state` and `next_state`). The FSM may also jump to state `R_ped` from O if any of the pedestrian request or emergency request signals are high. The counter values are updated accordingly.

The counters are updated every time a state transition occurs or when the new $T_{g,i}$ values are put into use for the first time.

Part 2: Sensor Unit

The sensor unit takes input from four sensors, one for each of the different directions: N, S, E, W. It takes two generic arguments, k, the number of cycles over which the values need to be averaged, and `init_value`, which specifies how $N_i, i \in \{N, S, E, W\}$ are initialised. It takes in input from the sensors as the total aggregate count, and sends a reset signal to the sensors after taking the reading. The reading is taken for every N_i just when the light for the corresponding roadway becomes green. Thus, it also takes in input the control signals for the various traffic lights.

It is essentially an FSM which has states wait_{N, S, E, W} via which it keeps track of how many cycles have elapsed.

It keeps count of the total vehicles in the variable `count_{N, S, E, W}`, which is averaged after k cycles have elapsed. The bit width of these signals is kept large to ensure that they don't overflow for the given value of k. Once the specified number of cycles have elapsed, the values are updated after taking the average and brought out to the output.

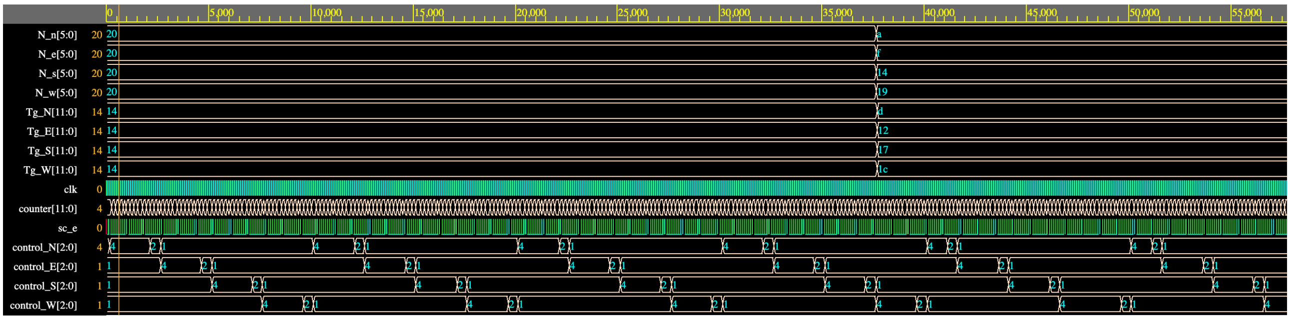
Part 3: Adaptation Unit

The adaptation unit takes as input the values $N_i, i \in \{N, S, E, W\}$ from the sensor unit. It takes in four generic arguments, k, `init_value`, β (here, it is kept as an integer. 1 or 2 correspond to the respective values and $\beta = 0$ represents the value 0.5.) and τ. It also takes in the input the control signals to the traffic lights to keep track of how many cycles have elapsed. It outputs $T_{G,i}$ for the different values of i.

Internally the $T_{G,i}$'s are represented as signed values with larger widths to account for the fact that they can take up negative values during an update and to also prevent overflow.

Whenever k cycles have elapsed, it inserts a buffer state in between to account for the fact that the sensor unit takes an extra clock cycle to calculate the N_i values. It then calculates the average of the N_i's, performs the update and checks whether the values have overflown. Then, it brings these values to the output so that the display unit can use these in the next iteration.

Simulation Results



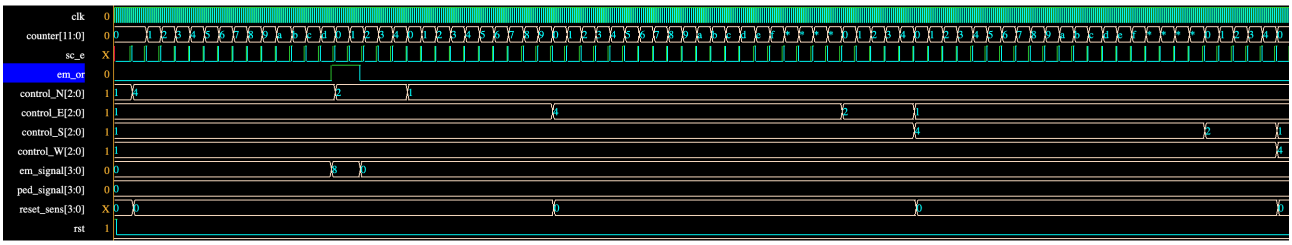
Running the simulation of all the three units together

The simulation was carried out with $\tau = 10$, $T_o = 5$, $\beta = 1$. The value of k was set to 4 so that the results could be easily seen in simulation, the system will still work when the value of k is set between 50 and 100.

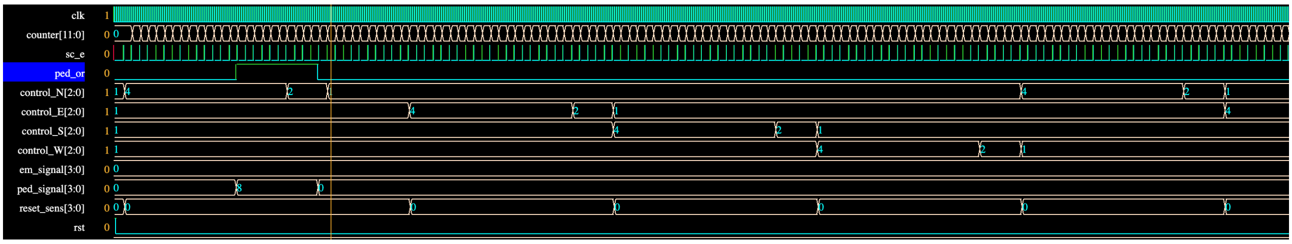
The only inputs given to the module are the sensor values which are not shown in the plot. The control_{N, S, E, W} signals represent the control signals for the traffic light, with the value '4' representative of a green signal, '2' representing orange signal and '1' representing red signal. It can be seen that initially the duration of all the green lights are the same, but once the update is performed (after $k = 4$ cycles have elapsed), the lengths of the green lights are unequal. It was verified that the values that the $T_{G,i}$ values take after the update are correct which signifies that the adaptation, sensor, and the display unit are working in unison as expected.

The simulation was re-run with different values of β, τ and the outputs were as they were expected.

Pedestrian/Emergency signals were also introduced, and the simulation still runs as expected.



emergency signal resulting in abrupt switching of the green light



insertion of all red phase due to pedestrian request

Note: In the first part, I had assumed that the sequence of lights is red -> orange -> green -> orange -> red, rather than, red -> green -> orange -> red. My mistake became clear to me after the timing/computation specifications were shared. Thus, there were modifications in the adaptation unit.