COL719: Lab 1
Design of Display Timer Unit

Varun Desai
2018EE10511

A brief description of the design is as follows.

The display timer unit is essentially a Finite State Machine. It has various states corresponding to when lane has the green light, N, S, E, W, along with an orange state, with states for all-red corresponding to the Pedestrian and Emergency states.

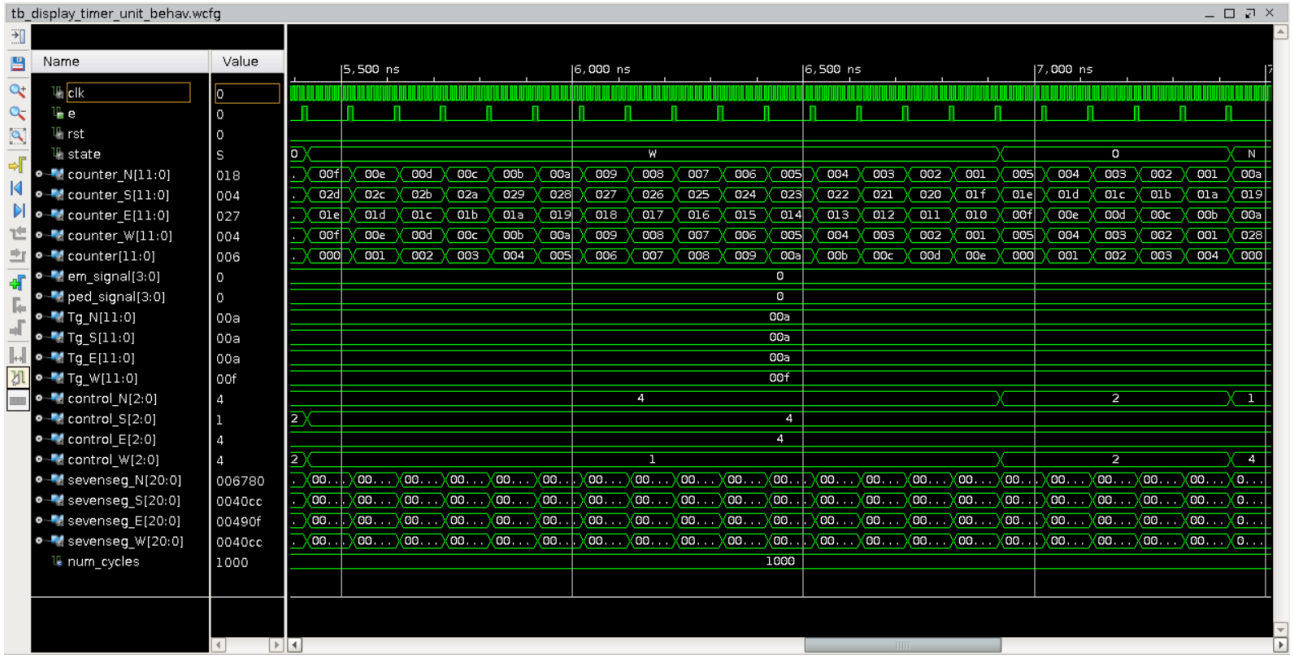
It is assumed that the system is clocked at a higher frequency that 1 Hz, and I have used a module which is a counter and signals whenever a second has elapsed. The frequency at which the system runs is currently assumed to be 10 Hz, but this can be set arbitrarily according to the clock frequency. All of the transition of the finite state machine occur only after a second has elapsed.

Whenever the traffic light system goes into an emergency state, the current state is skipped, and then the green light is signalled for the next lane. Which means, say the light is green for the N road, and then an emergency request is received, so an orange signal is sent on the N traffic lights which is followed by an all-red phase for 10 s. This is followed by a green state for the E road.

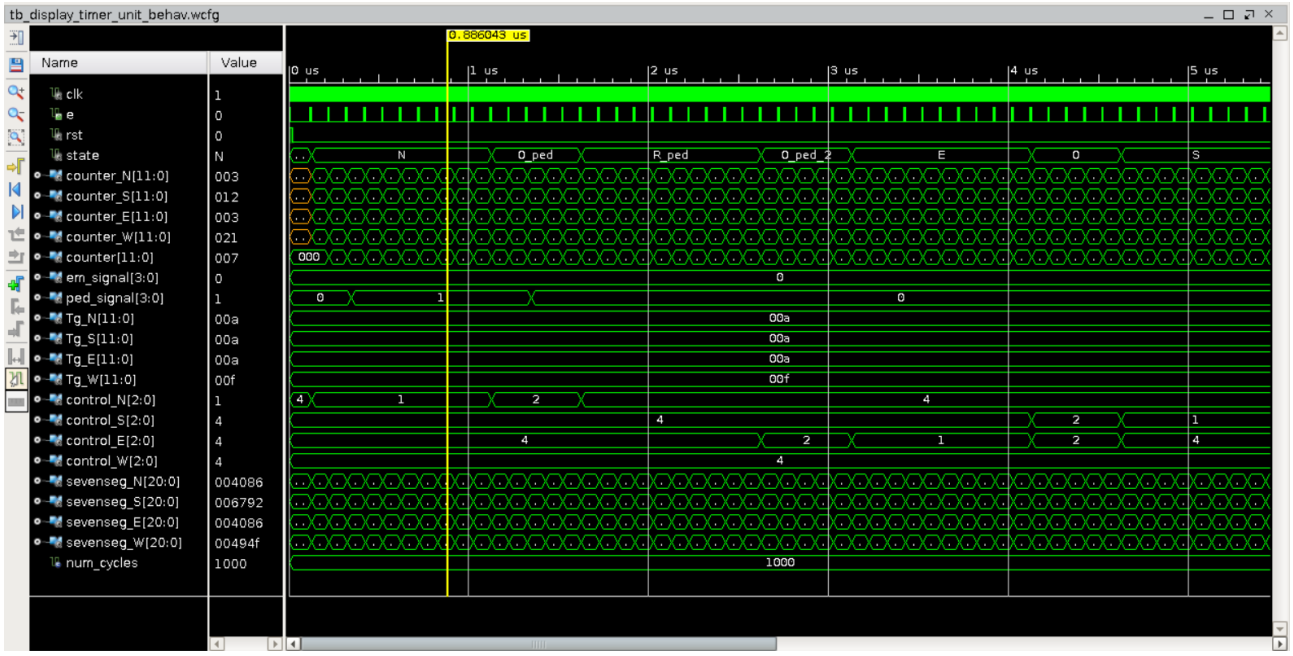
I have also made a module which converts a hexadecimal number into signals for led-pins on a seven segment display. I have used this module as a building block for creating the seven segment displays for the different roads.

The input values are updated only when entering the ‘N’ state. They are kept constant for one cycle, and only updated when a cycle ends. It is assumed that the values that are inputted are already rounded off to multiples of tau.

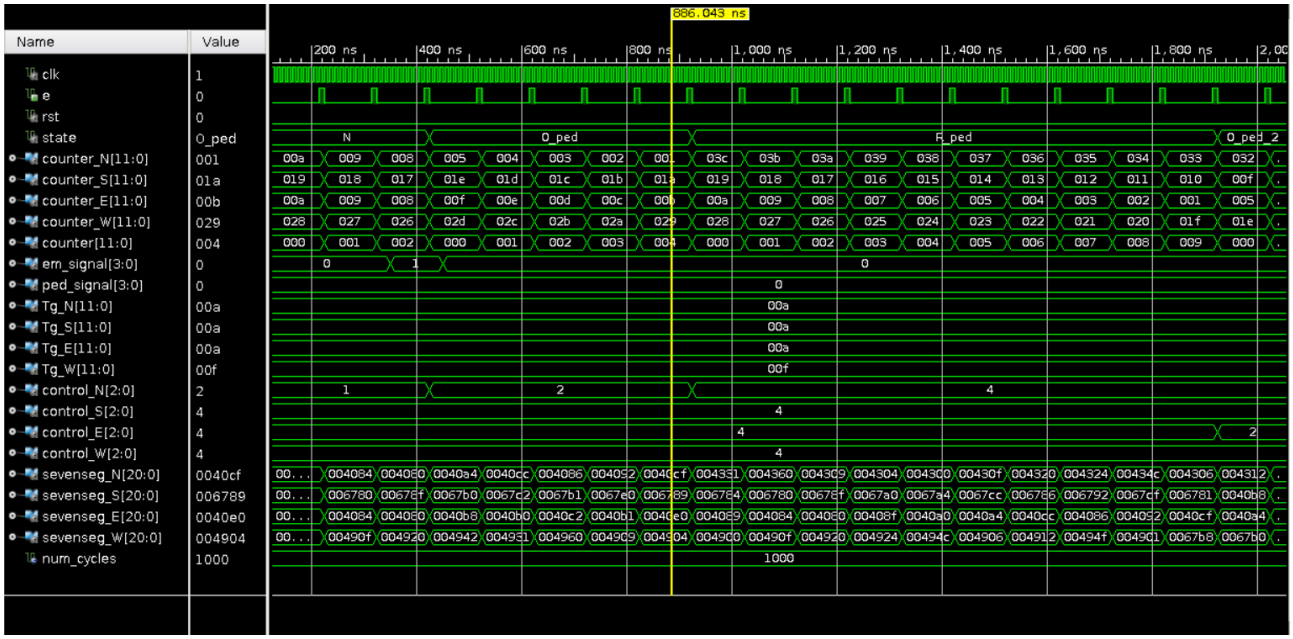
Simulation Waveforms



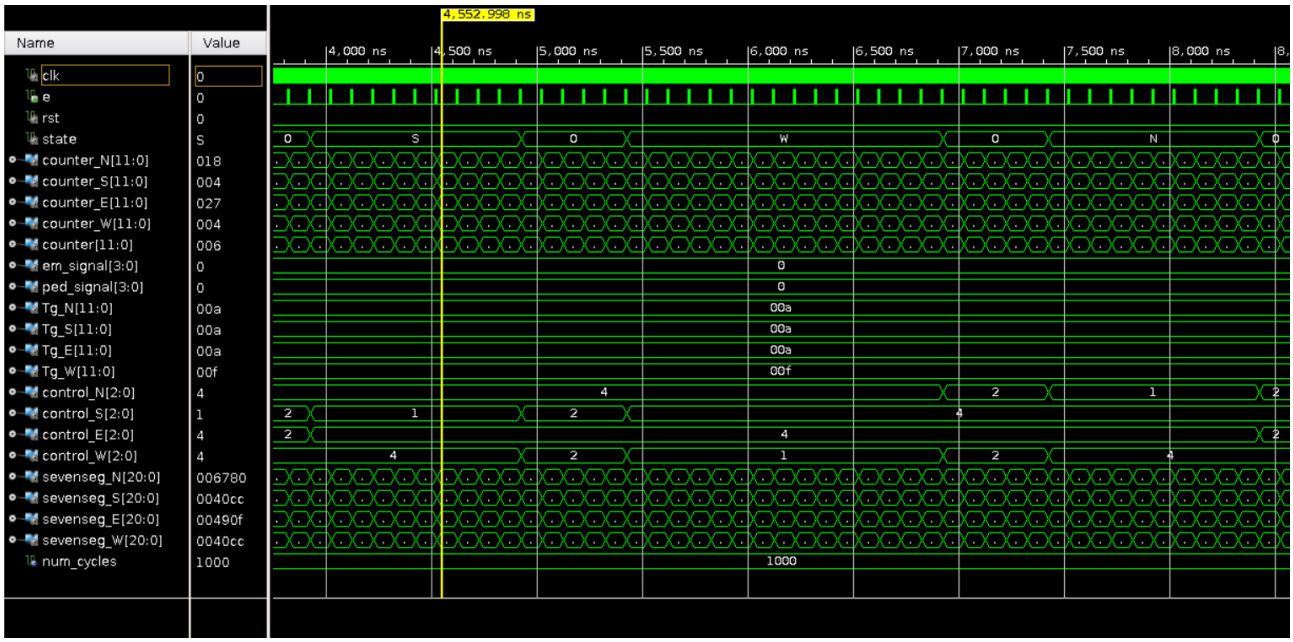
One can see the values for the different counters updating in the waveform above: one can see the counter_W and counter_N changing when the state changes from W(west) to O(orange). The control_{N,S,E,W} represent the traffic light. 4 is red, 2 is orange and 1 is green.



Pedestrian request: ped_singal[3:0] is high, and thus after the 'N' window is over, we have an orange phase followed by an all red phase.



Emergency Request: One can see the ’N’ state transitioning to an Orange Phase as soon as the request is raised (em_signal[3:0]) takes which is followed by an all-red phase. One can also see the counters getting updated correspondingly.



Standard operation of the FSM without any interrupts.

COL719: Lab Assignment 1
Submission Part 2

Varun Desai, 2018EE10511

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 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.

Adaptation Unit

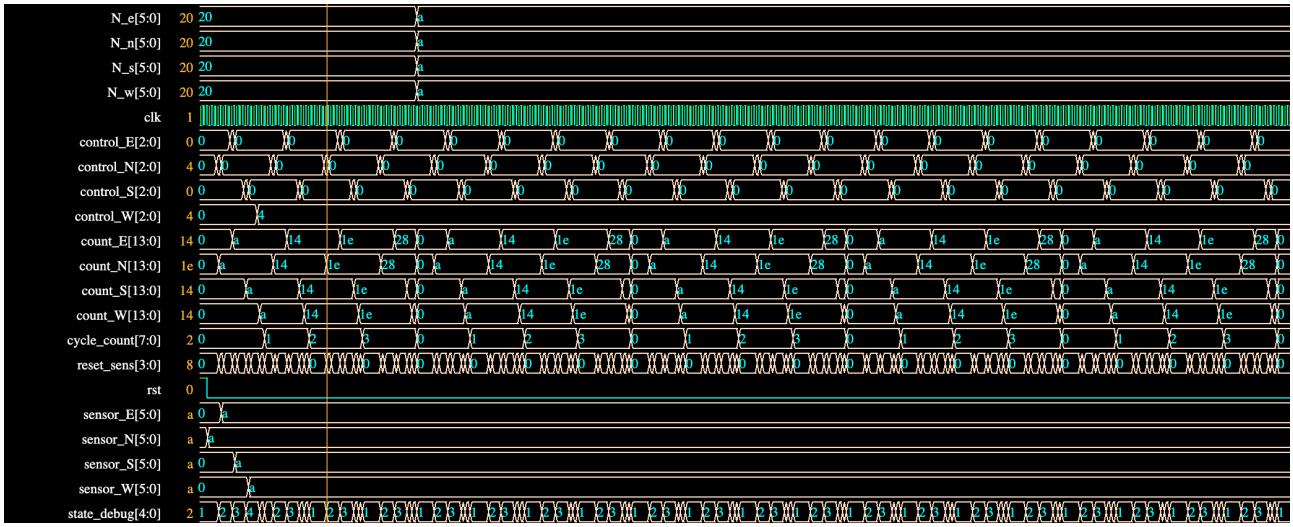
The adaptation unit takes as input the values $N_i, i \in \{N, S, E, W\}$ from the sensor unit. It takes in four generic arguments, k, `init_value`, β (here, it is kept as an integer. 1 or 2 correspond to the respective values and $\beta = 0$ represents the value 0.5.) and τ. It also takes in the input 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.

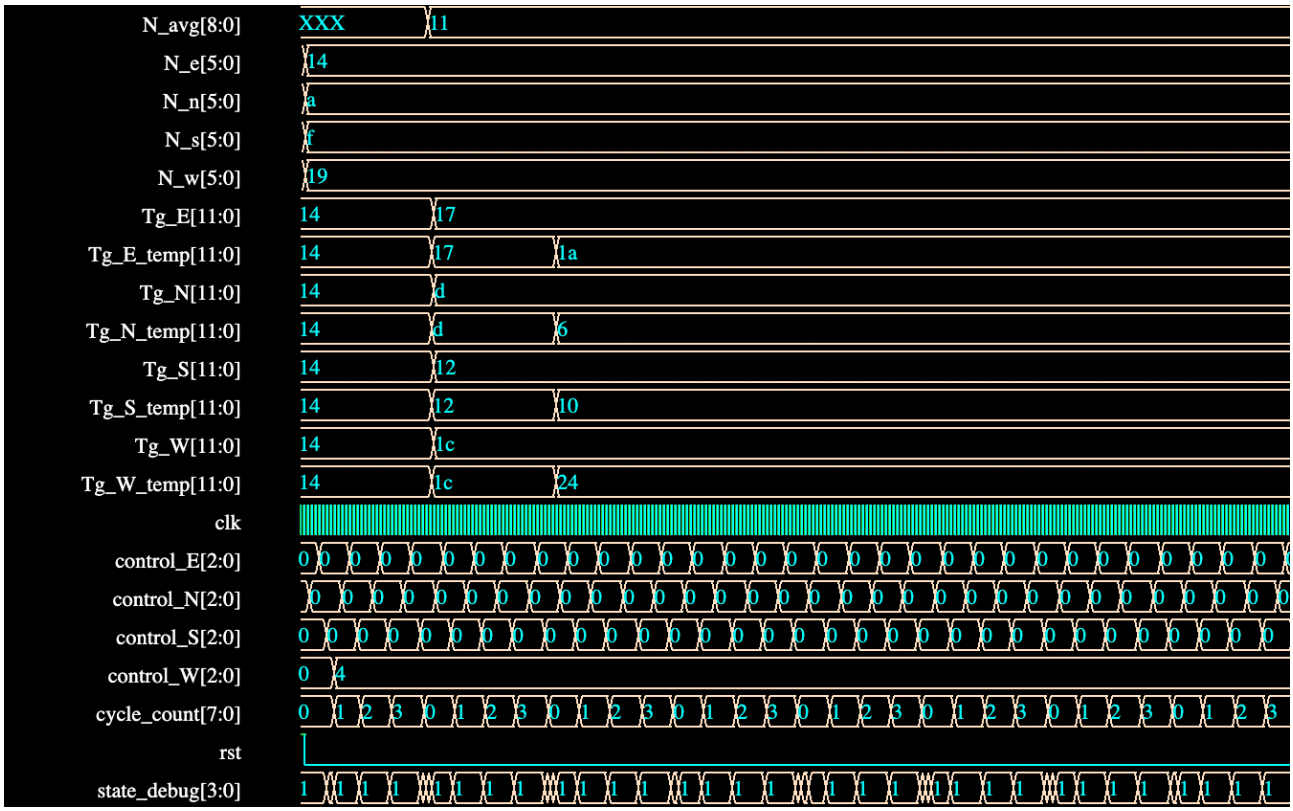
Simulation Results

Sensor Unit



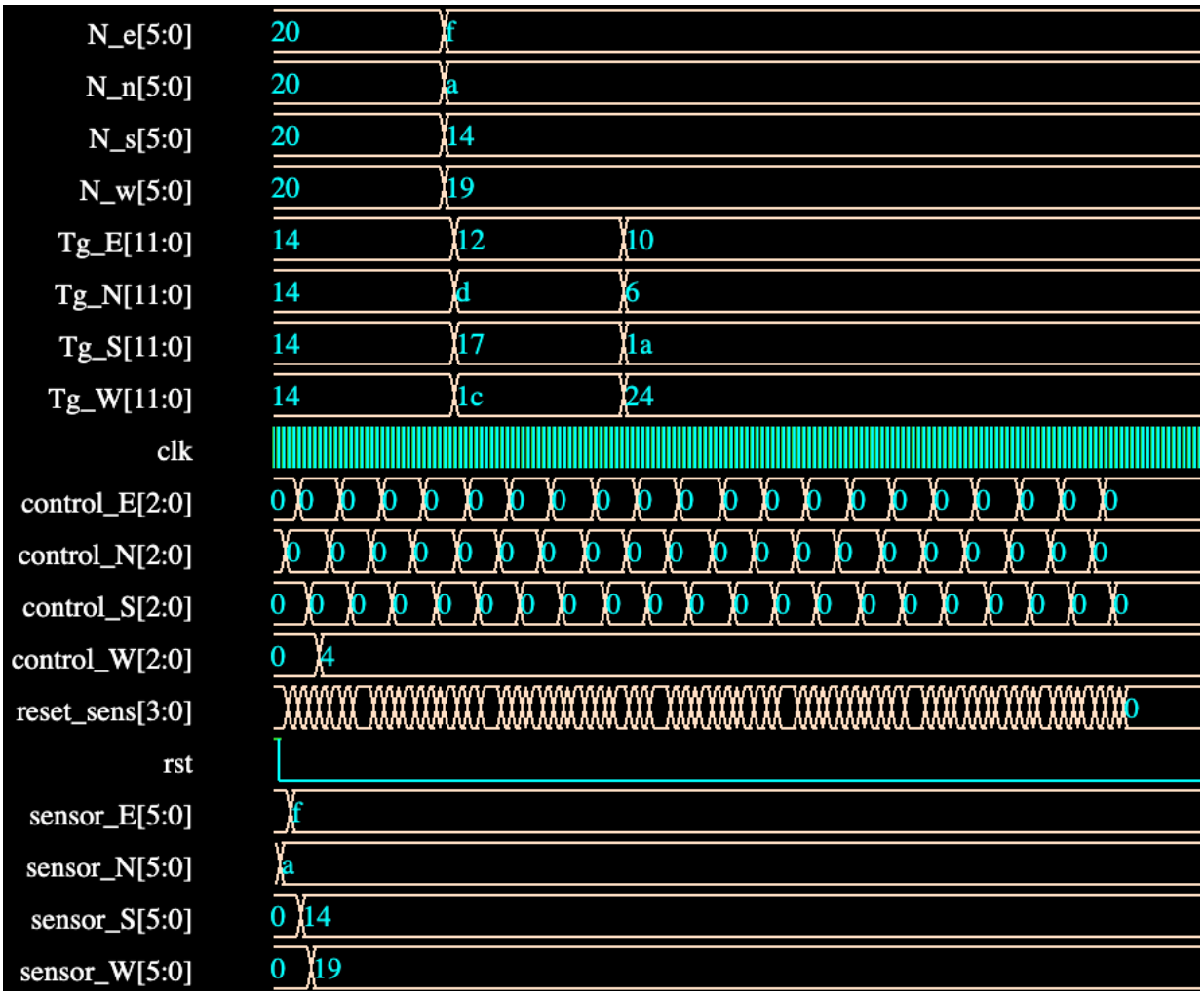
Here the value of k was set to 4 for convenience of simulation. One can see that the values count_{N, S, E, W} keep increasing until 4 cycles have elapsed, at which point the average value is taken and forwarded to the display unit. A reset signal which is sent to the sensors can also be seen here.

Adaptation Unit



Here k was set to 4 for convenience of simulation, it could have been kept as 50 and no difference would have been seen except the stretching of the updates over time. β was set to 1, and τ set to 10. It can be seen that the updates take place only as long as all values are above τ, as soon as $T_{G,N}$ falls to 6 the update is skipped and the values are constant for the future.

Both in Unison



Here, both the units are run together. The value of τ was set to 5 just for the purpose of simulation and verification of the code. The control signals were changed with time and the sensor values were given as input. It can be seen that the N_i, $T_{G,i}$ value are being updated.

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.

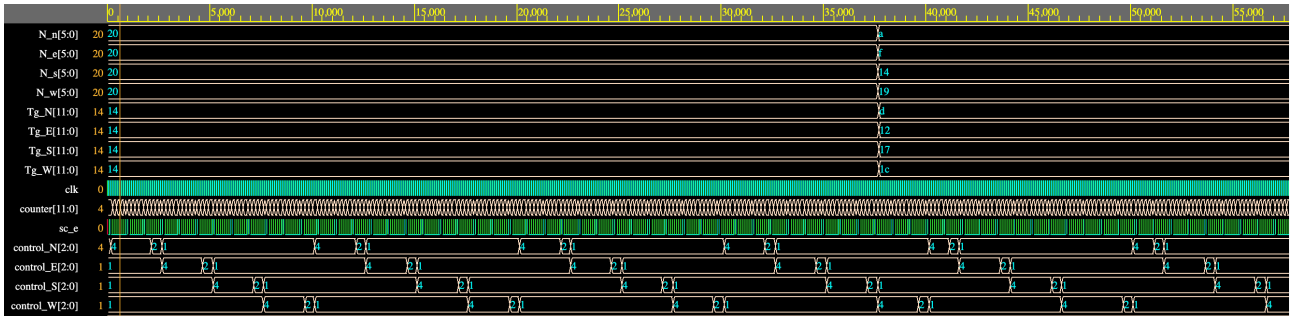
Part 3: Adaptation Unit

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



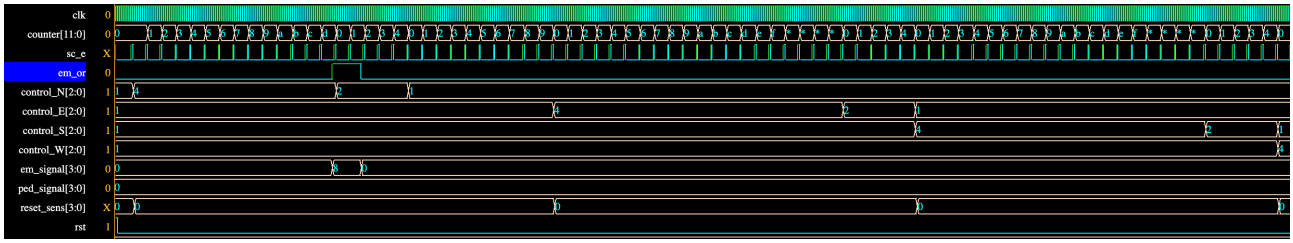
Running the simulation of all the three units together

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.

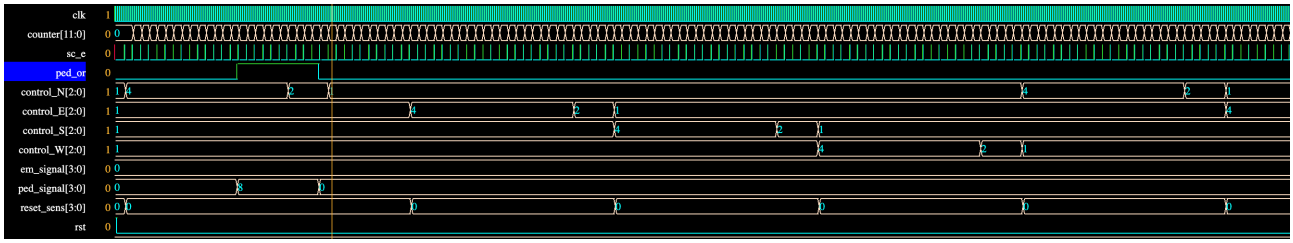
The only inputs given to the module are the sensor values which are not shown in the plot. The control_{N, S, E, W} signals represent the control signals for the traffic light, with the value ‘4’ representative of a green signal, ‘2’ representing orange signal and ‘1’ representing red signal. It can 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.