Students: -Ahmet BİÇER Department: Comp Eng %100 ID: 17290089

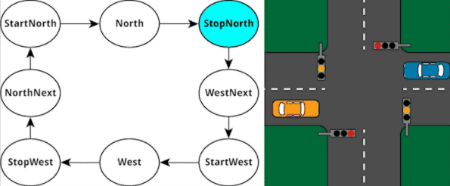
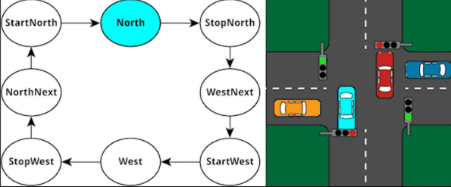
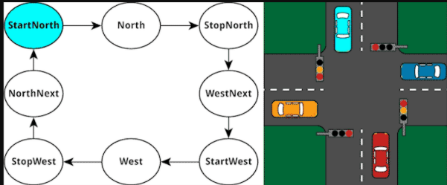
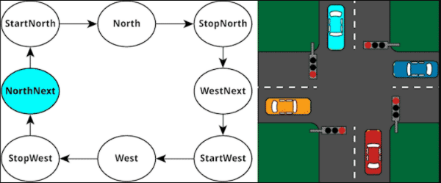
-Sinan Onur ÖZEN Department: Comp Eng %100 ID: 16290116

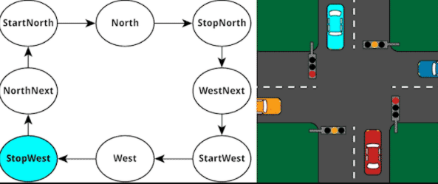
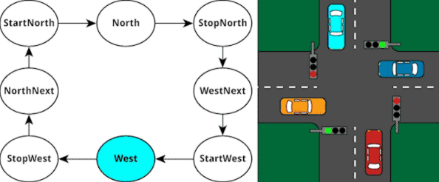
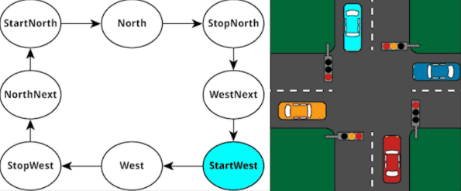
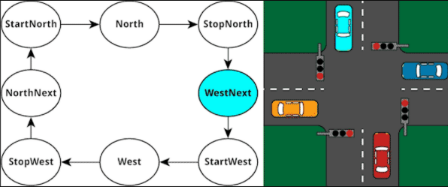
Problem: Managing traffic lights with finite state machine.

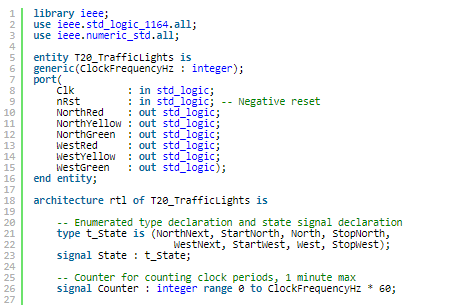
*Demonstration*: https://drive.google.com/file/d/1VmiCNQ8hskTEIZABqMiez3aimsmxYWUV/view?usp=sharing

**Module:**

Finite State Machine(FSM) is a model that based on one or more state computation. In this machine, only one state can be active in the running time. This means machine can make transition from a state to another to show different events. This device stores events or states of something at given time. States will change based on given inputs or outputs. In traffict lights problem we will cover 8 states of traffic lights with FSM. These are;





First of all, in our code we have to import our libraries. Library ieee and their lower parts std\_logic\_1164.all, numeric\_std.all are used for declaring our light signals and then in our entity we will use generic keyword to pass clockfrequency to manage our states. 

Also we have to declare our signals in our port.Because we will use port map in testbench to bring our signals to reuse. Their types are std\_logic. In FSM,when clock is open, we will change our states. This means, FSM machine will work if clock is open. Clock is declared like “ Clk”. “nRst” Is used for becoming state to default. Both “Clk” and “nRst” are showed like “in”type. When we start executing they will be running.They are independent from other signals but “out” signals are dependent on clock signal. Another types of signals are “NorthRed” “NorthYellow “,”NorthGreen”,”WestRed”,” WestYellow” and “ WestGreen”.Their type is “out”.Because they will change after time that we will declare with counter.In our architecture,we will execute our code.Then” type t\_state is (states)“ behaves like enums we can put our states in.For using our states we should declare states to signal.First state is “NorthNext”.In this state (in first picture),lights in all directions should be red so “NorthRed” and ”WestRed” are equal to 1.Also in this code south and east directions should be same as north and west directions(westRed->eastRed- northRed->southRed).After 5 seconds counter will be changed to 0,other state “StartNorth” is executed.In this state,Red and Yellow in north-south directions will be executed 5 seconds(NorthRed<=’1’,NorthYellow<=’1’, SouthRed<=’1’, South Yellow <=’1’).After “StartNorth” counter will be 0.Thus,after every state is executed,counter will be 0.When “StartNorth” finishes,”North”

state will be executed.In this state north and state direction will be green 60 seconds. So west-east direction will be red. (NorthGreen <=‘1’,WestRed<='1') After a minute counter will be again 0. Then stopNorth state will begin. In this state,north-south direction lights will be yellow color and west-east direction will be red (NorthYellow <= '1',WestRed <= '1'). Like this, when every state finishes also counter will be

0 and other counter will be executed. Yellow and Red lights will run 5 seconds. Green light will run 60 seconds. Order of states are like t\_states (NorthNext,StartNorth, North, StopNorth, WestNext, StartWest, West, StopWest). After StopNorth, WestNext occurs.In WestNext,

All directions are red (NorthRed <= '1', WestRed  <= '1'). The counter will be zero and StartWest starts. In StartWest west-east direction lights will be red and yellow then north-south direction will be red (NorthRed<='1', WestRed<= '1'WestYellow <= '1'). After this state counter again will be zero, then counter will be again 0. After counter, west state occurs. In this state west and east direction lights will be green 60 seconds and also north-south direction will be red (NorthRed  <= '1', WestGreen <= '1'). Then counter will be zero. After Weststate, StopWest begins. In StopWest, west-east direction lights will be yellow then north-south direction will be red (NorthRed<=1', WestYellow <= '1'). After this state, counter will be zero and NorthNext will start again. Only north and west state will run 60 seconds Others are 5. In this module, there are some deficiencies. Cars can go only one direction. They can go left or right. This can be developed. Or we can add intersection island with more lights maybe. Module is subprogram of the testbench. Module will be executed in test bench.

**Testbench:**

After we complete the module part we need to create the testbench to be able to control the time by using wait commands and to create some signals to send them to the module. First of all, we need to import our libraries like we did on module. Library ieee and their lower parts std\_logic\_1164.all, numeric\_std.all are used for declaring our light signals. Now, we can create our signals before the body of architecture, that means before the “begin” keyword. We need some constant values to adapt our program in real time. These are “ClockFrequencyHz” and “Clockperiod”. The type of “ClockFrequencyHz” is integer and the type of “ClockPeriod” is time. We give them initial values because they will not be change anymore . These initial values 100(Hz) for “ClockFrequencyHz” and (1000 ms)/ClockFrequencyHz for “ClockPeriod” .We can start to create our signals now, We give “Clk” and “nRst” signals initial values ‘1’ and ‘0’ so that the program be reset situation until we want it to work. The meaning of assigning ‘0’ to the “nRst” is making our simulator default situation. Default situation means that our all lights are red so that when program isn’t working our intersection would be safe. We don’t give any initial values to other signals because they will be assigned in the module depends on “nRst” and the rising edges of “Clk” signals.

 Next step, at the beginning of entity we need to pass “ClockFrequencyHz” as a generic map. We use port map to assign our signals to the signals that we use in the module. These signals have the same name as the other signals in the module, so we can easily assign them without any complexity. But there may be some confusion in the assignment above. Actually that just changes the value of Clk when half of the ClockPeriod time passed. That provides us some edges on the graphic of ”Clk” to use them

to switch between states.

Then, in our single process we are taking the devise under test(DUT) out of reset, means starting our program by passing time by 2 rising edge time of “Clk” and assigning 1 to“nRst”. If we don’t change it “0” to “1”, our simulator will be in passive mode forever and no car can move because every light would be red in the reset mod.

*Demonstration in mp4 format:* https://drive.google.com/file/d/1VmiCNQ8hskTEIZABqMiez3aimsmxYWUV/view?usp=sharing