Introduction

In this exercise, you will create a verilog implementation of a Finite State Machine (FSM) which corresponds to a **traffic light controller (tlc)**. Assume a crossing where a busy highway crosses a farmroad. There are sensors to detect the presence of cars waiting on the farm road. The traffic light controller has the following properties:

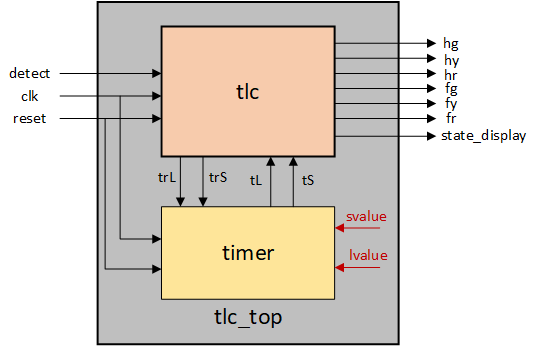
* With no car on the farmroad, light remains green in the highway direction.
* If sensors detect a vehicle on the farmroad, the highway lights go from **green** to **yellow** to **red**, allowing the farmroad lights to become**green**.
* The farm road lights stay **green** only as long as a farmroad car is detected but never longer than a fixed interval of **10 cycles**.
* After this, the farmroad lights transition from **green** to **yellow** to **red**, allowing the highway to retrun to **green**.
* Even if farmroad vehicles are waiting, highway gets at least a fixed interval of **10 cycles**as **green**.

**Important Points:**

**Assume: Short interval of yellow light = 5 cycles**

**Maximum interval for green light on the farm road and minimum interval for green on highway = 10 cycles**

The top level block diagram of the traffic light controller system is shown below:

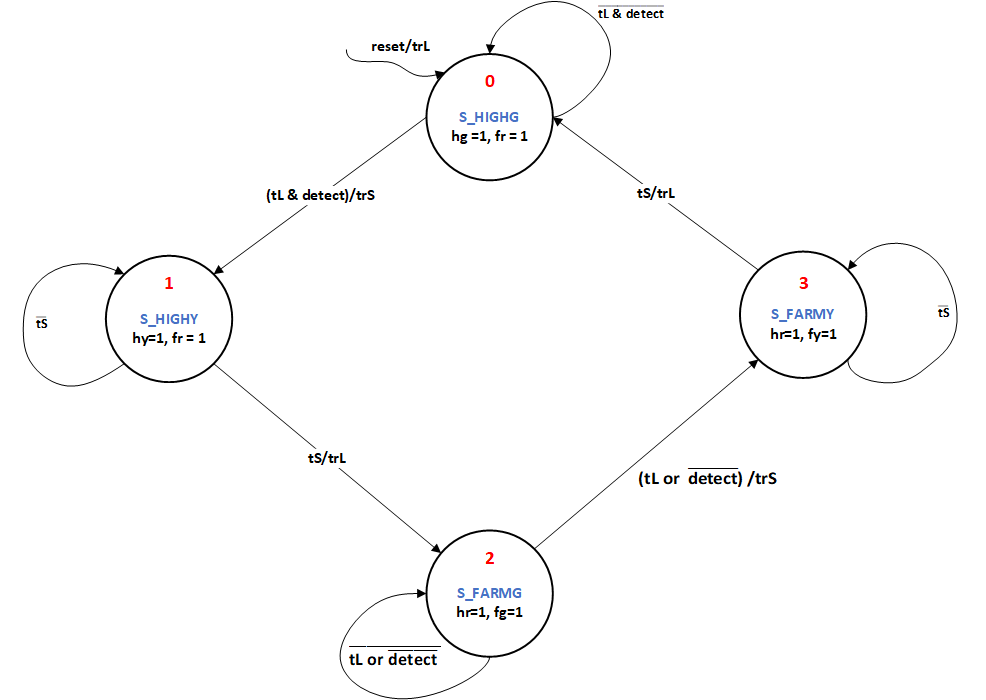


As shown in the diagram above, the traffic light controller system consists of a FSM module named **tlc**and a timer module named **timer**. A module named **tlc\_top** wraps these two modules. The details of the signals are given below:

* **detect (1 bit) -** sensor outputdetecting the presence of a car on the farm road
* **trL(1 bit) -**trigger signal to start counting for **lvalue**cycles
* **trS (1 bit) -**trigger signal to start counting for **svalue** cycles
* **tL (1 bit) -**signal indicating expiration of a time interval of **lvalue** cycles
* **tS (1 bit) -** signal indicating expiration of a time interval of **svalue** cycles
* **state\_display (3 bits)-**current state of the FSM module **tlc**
* **hg (1bit) -** highway green
* **hy (1bit) -**highway yellow
* **hr (1bit) -** highway red
* **fg (1bit) -**farm green
* **fy (1bit) -**farm yellow
* **fr (1bit) -**farm red

Assignment

In this assignment, you have to write the FSM module named **tlc**.  It takes input signals **clk, reset, detect, tL, tS** and output signals **hg, hy, hr, fg, fy, fr, state\_display. trL, trS**. Assume all input and output signals to be 1 bit wide except state\_display which is 3 bits wide and little endian. The state\_display must output the number of the current state using the same numbers as the states in the FSM diagram above (shown in red). The **reset**input always makes a transition to the **START** state. The **reset**output is a **synchronous** signal (i.e. it goes high only at posedge of clock). The state transition diagram for the FSM module is given below:



**Note**: The **timer** module is already provided. The **timer** module synchronizes with the **tlc** module using **tL** and **tS** signals. Configure the parameters **svalue** and **lvalue**as per the design of your **tlc**module to obtain the following behaviour:

* A delay of 10 cycles while transitioning from state 0 to state 1.
* A delay of 5 cycles while transitioning from state 1 to state 2.
* A delay of 10 cycles while transitioning from state 2 to state 3.
* A delay of 5 cycles while transitioning from state 3 to state 0.

Use the template given below for desiging the FSM:

* `timescale 1ns / 1ps
* module tlc(clk, reset, detect, hg, hy, hr, fg, fy, fr, tL, tS, state\_display, trL, trS);
* <<Write your FSM logic here>>
* endmodule
* module tlc\_top(clk,reset,detect, hg, hy, hr, fg, fy, fr, state\_display);
* input clk, reset, detect;
* output hg, hy, hr, fg, fy, fr;
* output [2:0] state\_display;
* wire trL, trS, tL, tS;
* parameter svalue = <<Change svalue to make tS=5 cycles>>;
* parameter lvalue = <<Change lvalue to make tL=10 cycles>>;
* timer #(
* .svalue(svalue),
* .lvalue(lvalue))
* timer\_inst(
* .clk(clk),
* .reset(reset),
* .trL(trL),
* .trS(trS),
* .tL(tL),
* .tS(tS));
* tlc tlc\_inst(
* .clk(clk),
* .reset(reset),
* .detect(detect),
* .hg(hg),
* .hy(hy),
* .hr(hr),
* .fg(fg),
* .fy(fy),
* .fr(fr),
* .trL(trL),
* .trS(trS),
* .tL(tL),
* .tS(tS),
* .state\_display(state\_display));
* endmodule