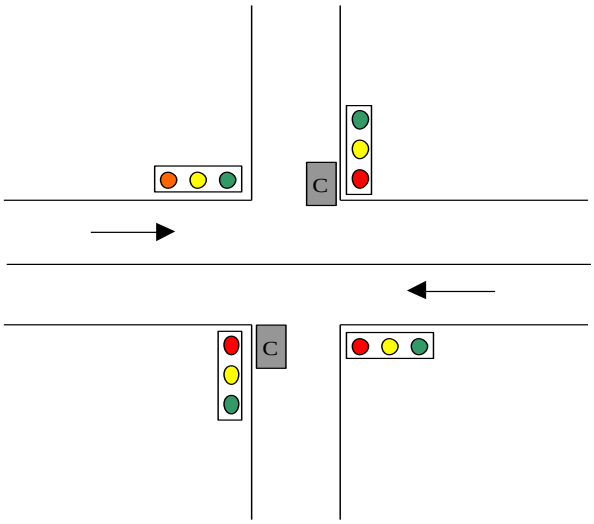
Lab V

Sequential circuit design

1. **Sequence Detector** A digital system has one input **X** and two outputs **Y** and **Z**. The output **Y=1** occurs every time the input sequence **101** is completed provided that the sequence **011** has never occurred. The output **Z=1** occurs each time the output **011** is completed, i.e. once **Z=1** has occurred, **Y=1** can never occur but not vice versa. Design the Mealy state machine and implement it using Verilog.

2. **Traffic light controller for NITK**



Model a traffic light controller for NITK using Verilog and implement on FPGA. The National highway NH66 is intersected by the NITK road as shown in Figure. Detectors are placed along the NITK road to raise the signal **C** as long as a vehicle is waiting to cross the highway. The traffic light controller should operate as follows: as long as no vehicle is detected on the NITK road the lights should remain green in the highway direction. If a vehicle is detected on the NITK road, the highway lights should change from green to yellow to red, allowing the NITK road lights to become green. The NITK road lights stay green only as long as a vehicle is detected on the NITK road and never longer than a set interval to allow the traffic to flow on the highway. If these conditions are met the NITK road lights change from green to yellow to red, allowing the highway lights to return to green. Even if vehicles are waiting to cross the highway, the highway lights should remain green for a set interval. You may assume that there is an external timer that, once set via the control signal **ST** (set timer) will assert the signal **TS** after a short time interval has expired (used for timing yellow lights), **TLH** and **TLF** after a longer time interval (for green lights, highway and NITK road respectively). The timer is automatically reset when **ST** is asserted.

Note

- You can use the push button switches KEY0-3 for giving single pulses instead of a clock to observe the function
- Before compiling your code it is necessary to explicitly tell the Synthesis tool in Quartus that you wish to have the finite state machine implemented using the state assignment specified in your Verilog code. If you do not explicitly give this setting to Quartus, the Synthesis tool will automatically use a state assignment of its own choosing, and it will ignore the state codes specified in your Verilog code. To make this setting, choose **Assignments > Settings** in Quartus, and click on the Compiler Settings

item on the left side of the window, then click on the **Advanced Settings (Synthesis)** button, change the parameter **State Machine Processing** to the setting **User-Encoded**.

- To examine the circuit produced by Quartus open the RTL Viewer tool. Double-click on the box shown in the circuit that represents the finite state machine.
- To see the state codes used for your FSM, open the **Compilation Report**, select the **Analysis and Synthesis section** of the report, and click on **State Machines**.
- To see the result of removing this setting, open again the Quartus settings window by choosing **Assignments > Settings**, and click on the **Compiler Settings** item on the left side of the window, then click on the **Advanced Settings (Synthesis)** button. Change the setting for **State Machine Processing** from **User-Encoded** to (a) **Auto** (b) **One-Hot**. Recompile the circuit and then open the **report** file, select the **Analysis and Synthesis** section of the report, and click on **State Machines**. Compare the state codes and discuss any differences that you observe.