

**Fall 2024**

**Selected Topics in Computer Design-CSE416s**

**Final Project**

**Adaptive Traffic Light Controller-Part A**

**Team #4**

Members:

|  |  |  |
| --- | --- | --- |
| Name | Code | Department |
| Fares Khalaf Salman Sultan | 2101371 | CSE |
| Youssef Hany Elreweny | 2101449 | CSE |
| Ahmed Ashraf Ali | 2100255 | CSE |
| Youssef Ossama Sayed | 2101240 | CSE |

* **Controller Specifications:**

The controller is designed for use in a four-way intersection. It uses sensors readings to prioritize traffic flow dynamically. The controller operates through a finite state machine to cycle through traffic light states based on traffic conditions and timing, and an internal counter loaded with an appropriate Value based on the Traffic light.

**1. Features:**

* **Dynamic Traffic Management**: Adjusts traffic light durations based on real-time vehicle density from four directions (A, B, C, D).
* **Sequential State Transition**: Includes green, orange, and red states for each direction.
* **Sensor-Based Prioritization**: Uses sensor readings (Sa, Sb, Sc, Sd) to determine the next state.
* **Counter-Based Timing**: A configurable counter determines the duration of each traffic light state.

**2. Inputs and Outputs:**

* **Inputs**:
  + **Sa, Sb, Sc, Sd:** Two-bit signals representing traffic density in each direction (00→No cars, 01→Light Traffic, 11→Heavy Traffic).
  + **clk:** Clock signal.
  + **rst\_n**: Asynchronous Active-low reset signal.
* **Outputs**:
  + **Ta, Tb, Tc, Td:** Three-bit signals representing the traffic light state for directions A, B, C, and D (001 → green, 010 →orange, 100 → red).

**3. State Description:**

* **Green States (Ga, Gb, Gc, Gd)**:
  + The corresponding direction has a green light.
  + Counter set to 30 seconds.
  + Transitions to orange state (Oa, Ob, Oc, Od) when the counter expires.
* **Orange States (Oa, Ob, Oc, Od)**:
  + The corresponding direction has an orange light.
  + Counter set to 3 seconds.
  + Transitions to the green state of the next prioritized direction.

**4. Timing:**

* Green light duration: 30 seconds (adjustable via load\_value).
* Orange light duration: 3 seconds (fixed).
* Counter decrements each clock cycle.

**5. Priority Rules:**

* The direction with the highest traffic density has priority for the next green light.
* If multiple directions have equal density, default to a fixed priority order:

**(A → B → C → D).**

* **Traffic Light Algorithm:**

**1. Initialization:**

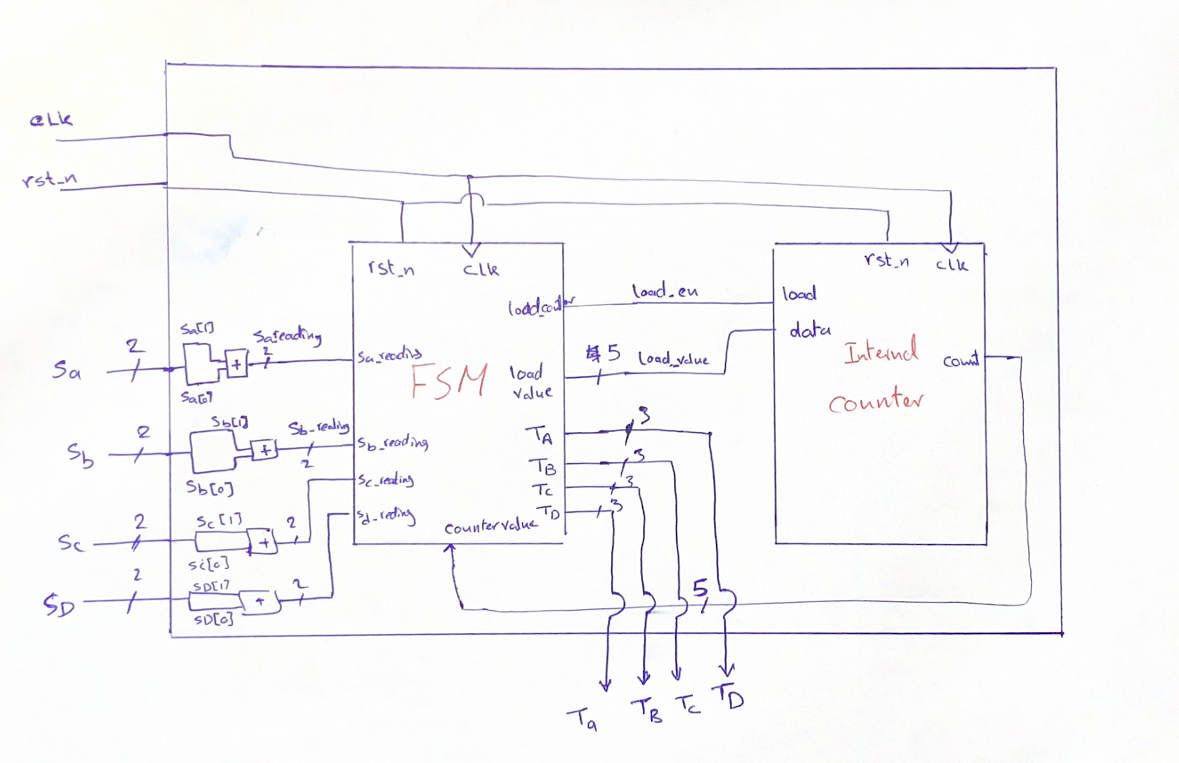
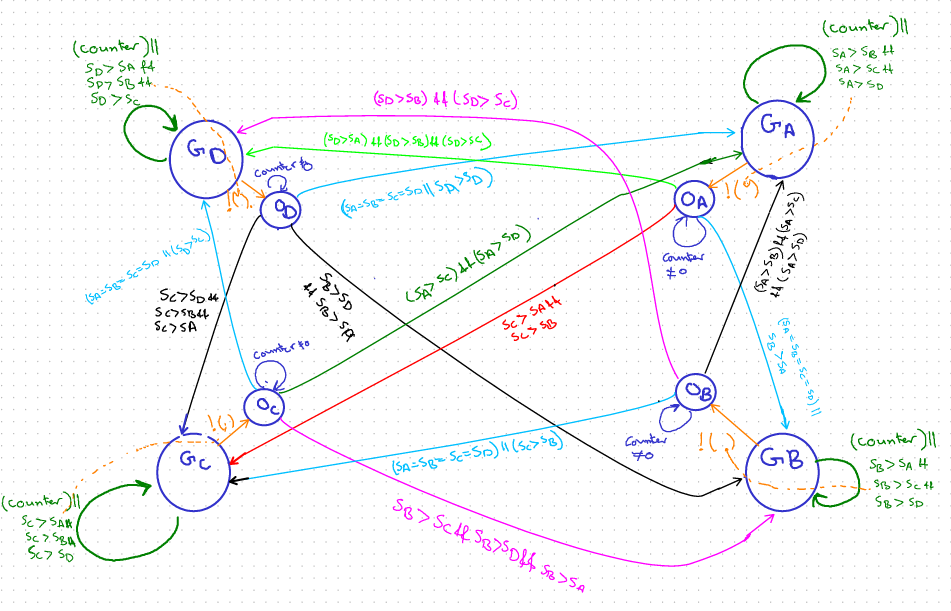
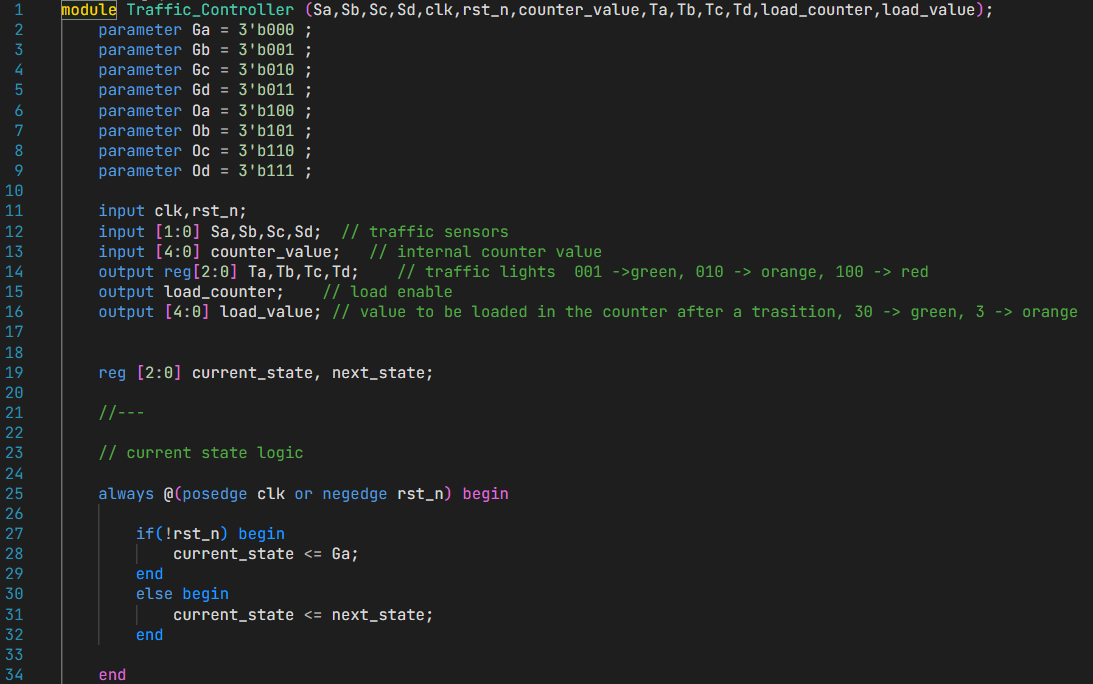
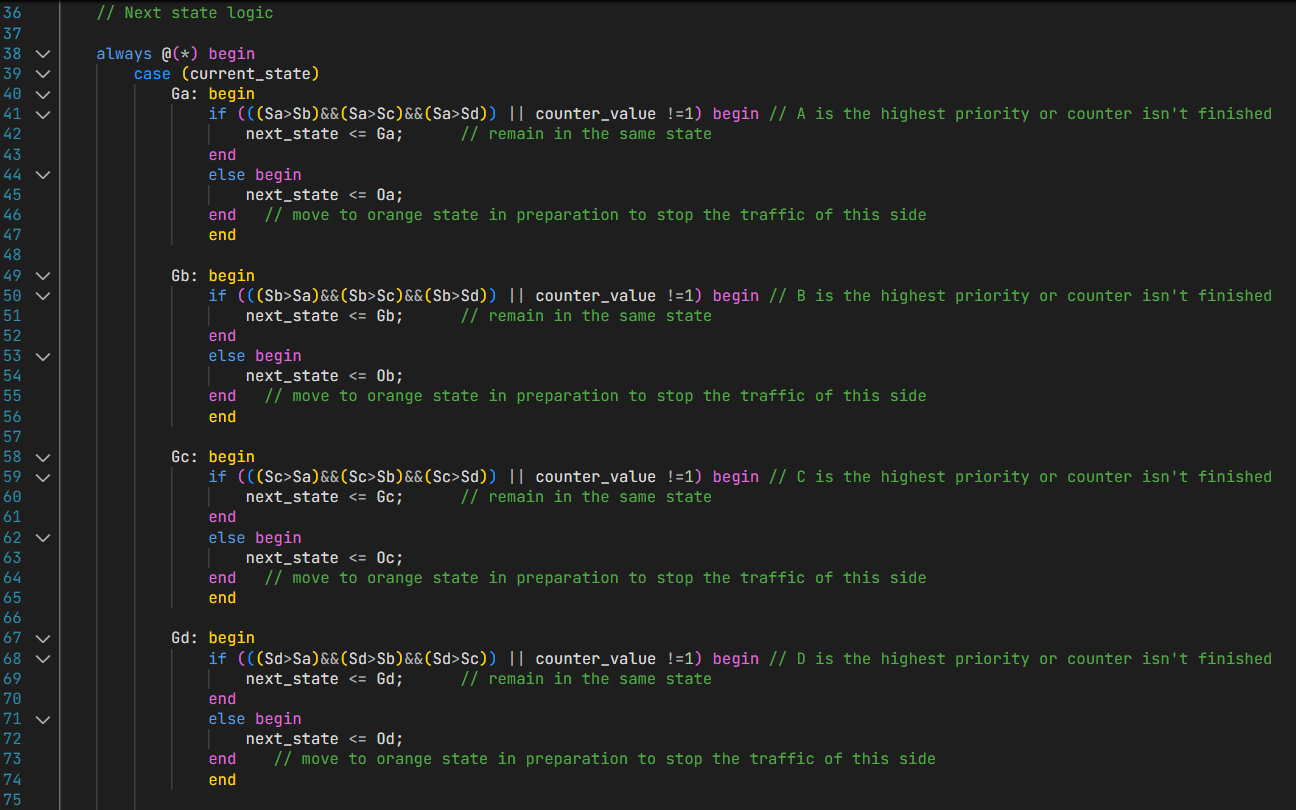
1. Set **current state** to **Ga** (Direction A green).
2. Load the counter with 30 for green light duration.

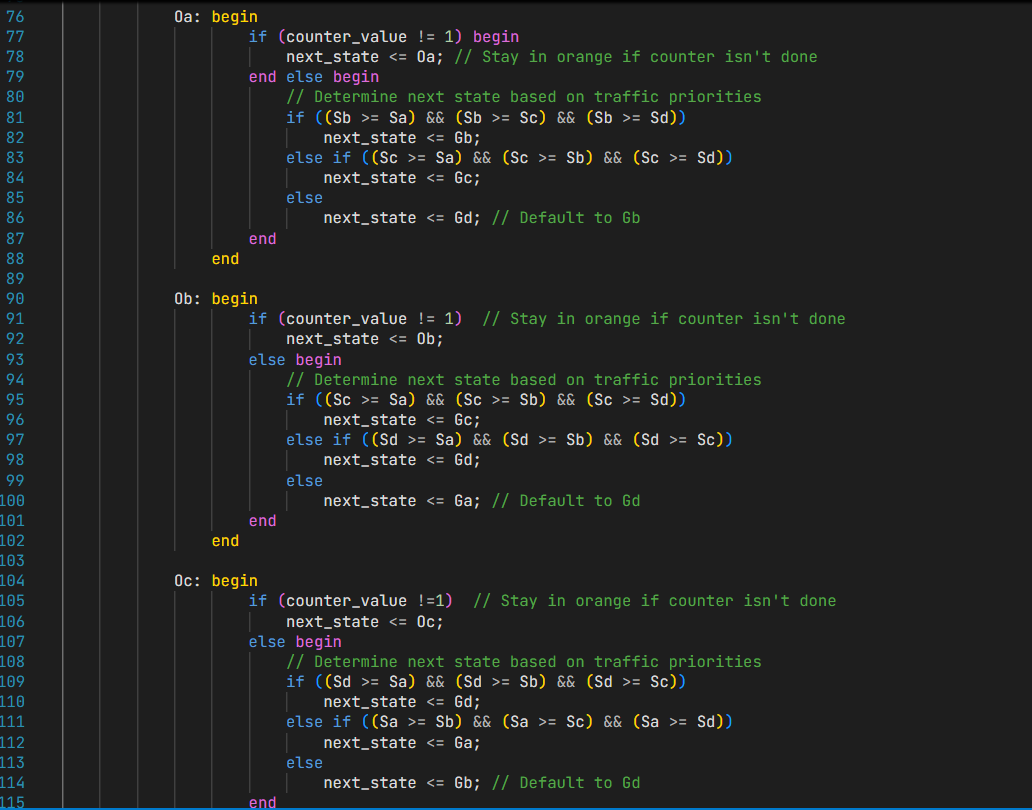
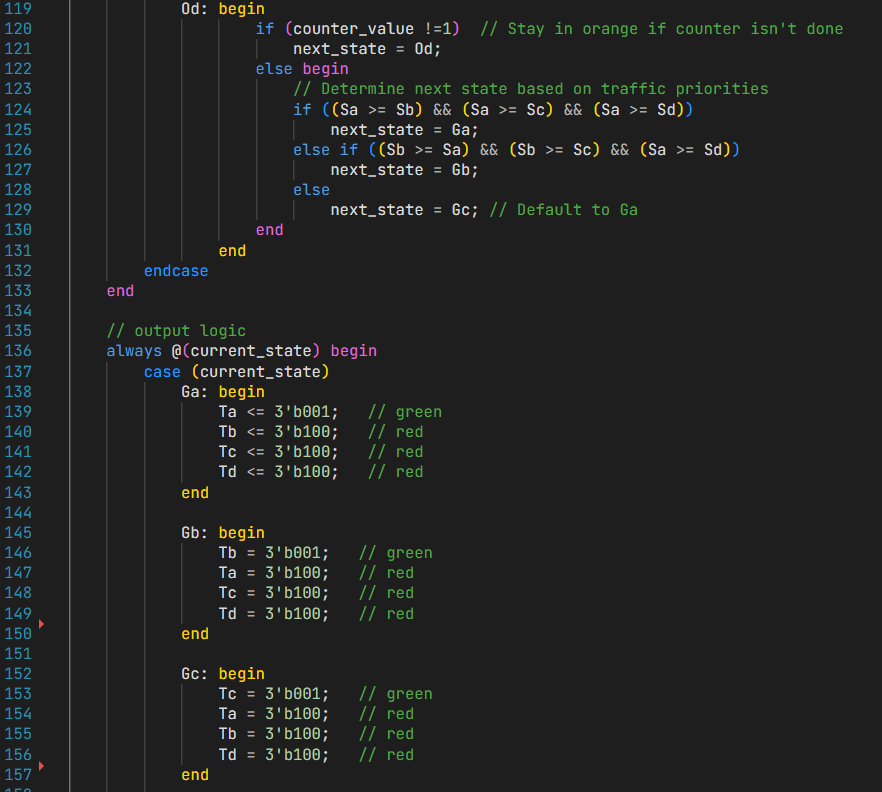
**2. FSM Logic:**

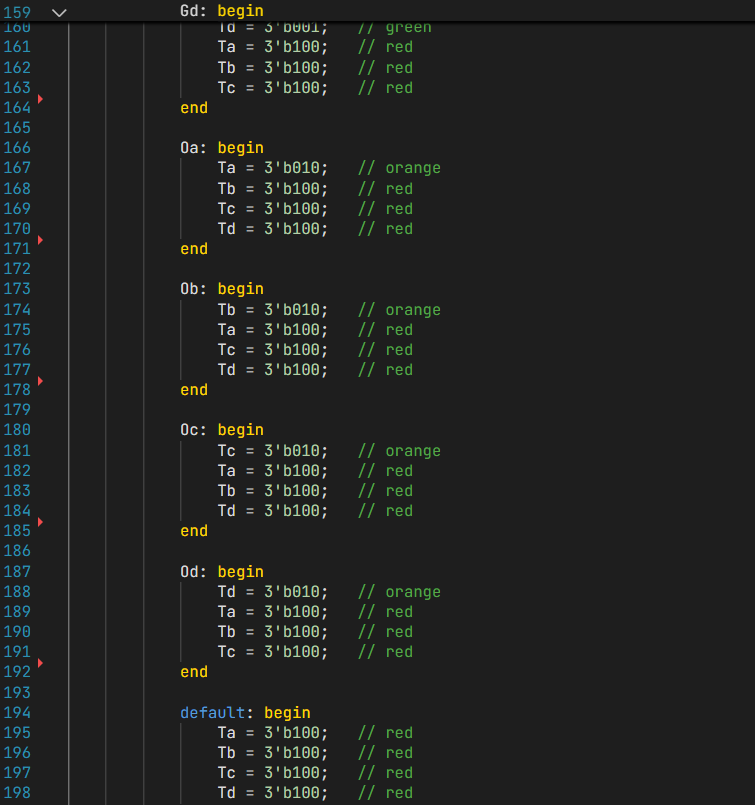
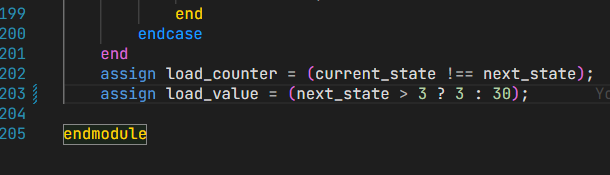
1. **Green States (Gx)**:
   * Check sensor readings:
     + If the current direction (Sa, Sb, Sc, or Sd) has the highest density, remain in the green state (Gx).
   * If the counter expires, transition to the corresponding orange state (Ox).
2. **Orange States (Ox)**:
   * Decrement the counter.
   * If the counter expires:
     + **Determine the next direction to prioritize based on sensor inputs:**
       - transition to the green state of the direction with the highest traffic density.
       - In case of ties, follow a fixed order (A → B → C → D).

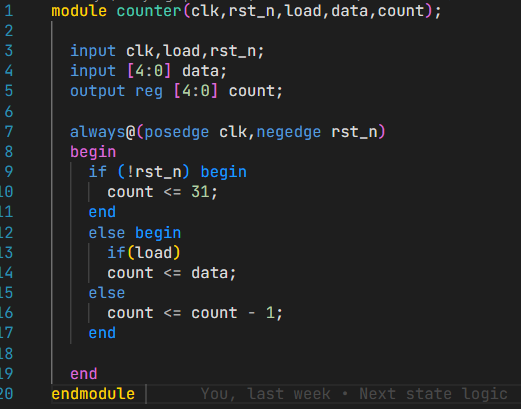
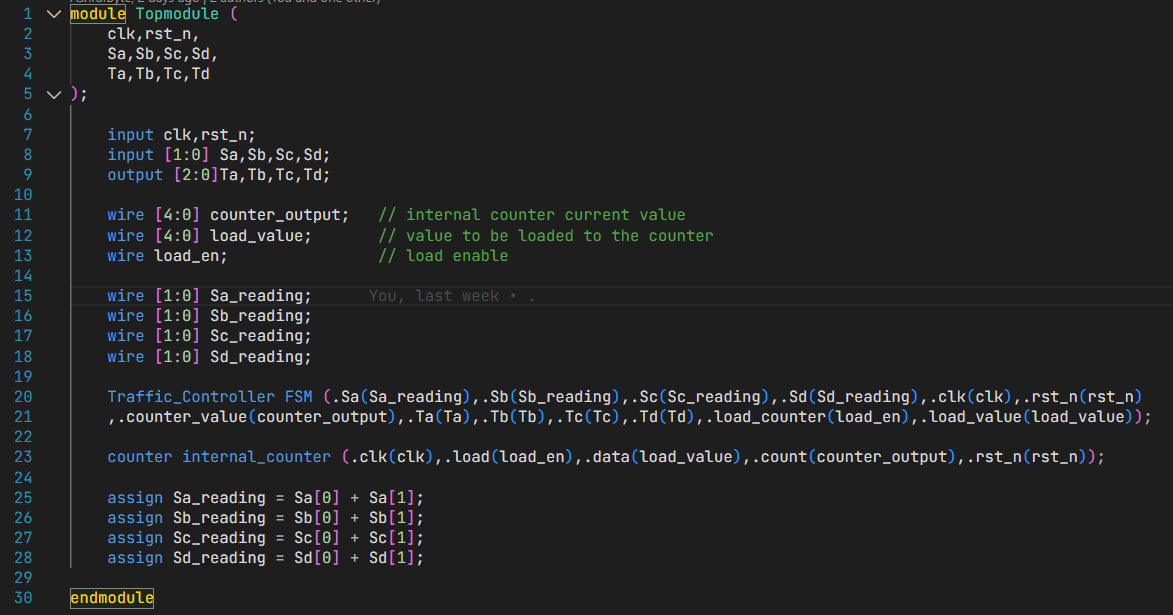
**3. Output Logic:**

Use the current state to determine light signals:

* + In Gx: The corresponding direction is green, and others are red.
  + In Ox: The corresponding direction is orange, and others are red.
  + When transitioning between states, reload the counter with the appropriate value (30 for green, 3 for orange).
* **System Design:**
* **FSM Diagram:**
* **RTL code:**
* **Traffic\_controller (FSM):**

****

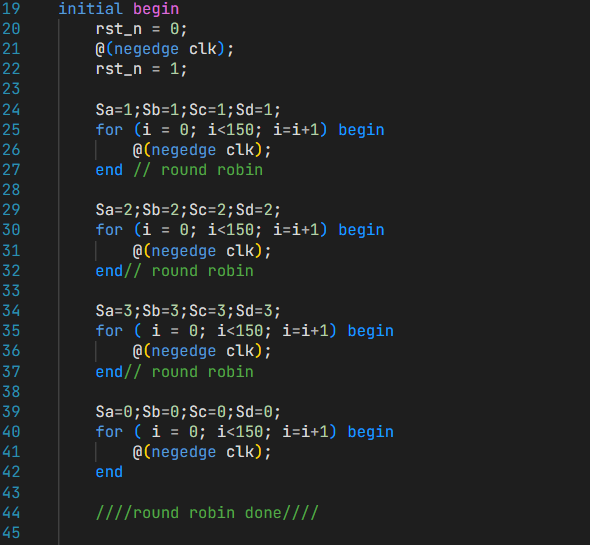


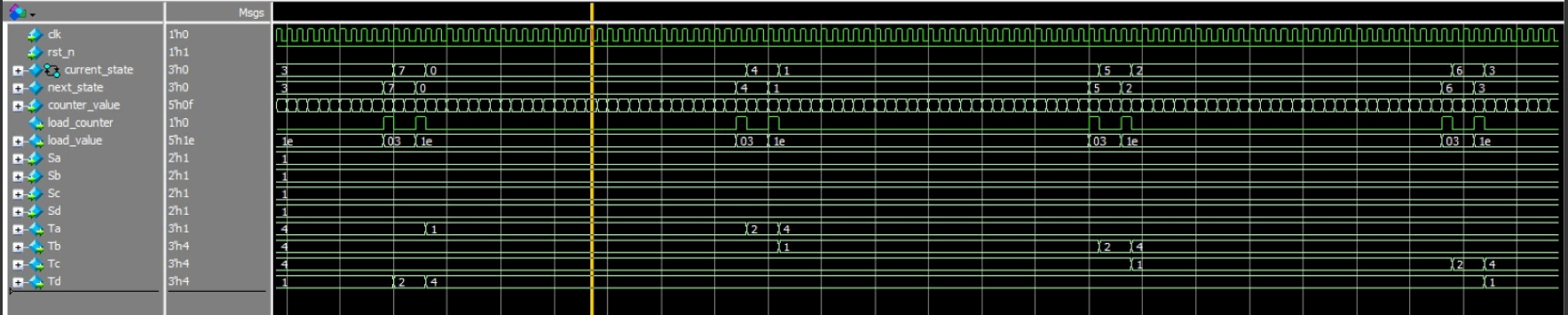
* **Counter:**
* **Top Module:**
* **Verification:**

# **Test Strategy for Traffic Light Controller Test bench:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Scenario | Objective | Inputs (Stimuli) | Expected Outputs/Behaviour |
| Reset Functionality | Ensure all traffic lights reset to a default state when rst\_n = 0. | rst\_n = 0 | All traffic lights switch to the reset state (e.g., Red for all directions). |
| Round-Robin Scheduling | Verify fair and cyclic light transitions in  all directions (no starvation). | Sa = 1, Sb = 1, Sc = 1, Sd = 1 (or similar priority inputs). | Traffic lights transition in a round-robin manner: one direction turns Green, and others remain Red. |
| Sensor-Based Adjustments | Confirm light timings adjust dynamically based on sensor signals (at positions 1 and 5). | Sensor at a specific lane (e.g., Sa = 3, others = 0). | Extended Green for the lane with higher priority sensor values, other lights maintain Red. |
| Fixed Priority Testing | Test priority-based light handling to ensure correct precedence among directions. | Assign higher priority (Sd = 3) while others are lower. | Priority lane (Sd) maintains Green until completion, other lanes wait (Red). |
| Dynamic Priority Changes | Check behaviour when priorities change dynamically during operation. | Vary priorities in real-time (e.g., Sa = 2, Sd = 3, etc.). | The system adapts to new priorities, transitioning traffic lights accordingly. |
| Edge Cases | Test system with all lanes inactive or all lanes highly active. | - Case 1:  Sa = Sb = Sc = Sd = 0. - Case 2:  Sa = Sb = Sc = Sd = 3. | - Case 1: All directions should remain Red. - Case 2: Ensure non-conflicting Green transitions or round-robin behaviour. |
| Conflicting Paths Prevention | Ensure system avoids intersecting traffic paths as per design. | Simulate conflicting paths using inputs representing multiple active lanes. | Lights for conflicting paths are never simultaneously Green. |
| Clock Dependency | Validate output transitions occur only on the negative clock edge (negedge clk). | Observe behaviour during clock transitions. | Outputs (lights) update only on the negative edge of clk. |
| Long Simulation Runs | Check system stability and robustness over extended simulation time. | Vary input patterns over hundreds of cycles. | The system continues to function correctly under long-term scenarios (no deadlocks or undefined states). |
| Reset Recovery | Confirm the system recovers seamlessly after coming out of reset (rst\_n = 1). | rst\_n = 1 after being held low. | Lights resume normal operation following the pre-defined scheduling or priority rules. |

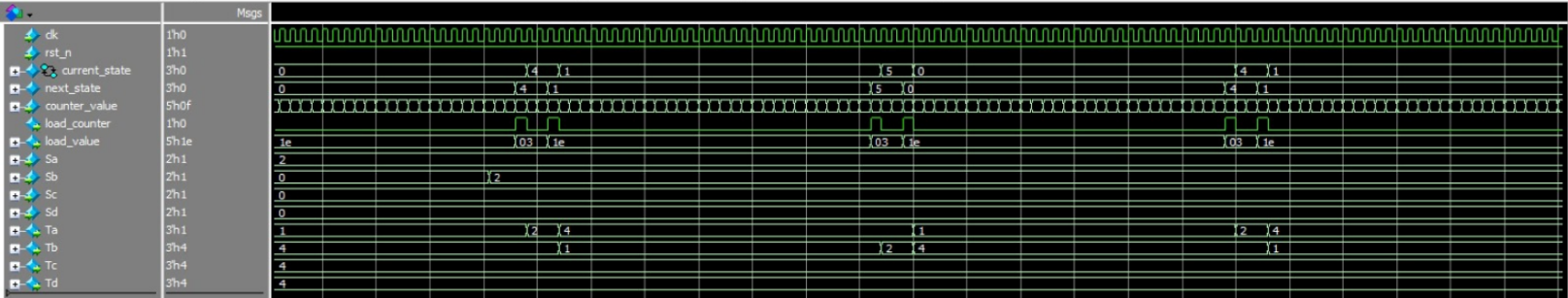
# **Test bench:**

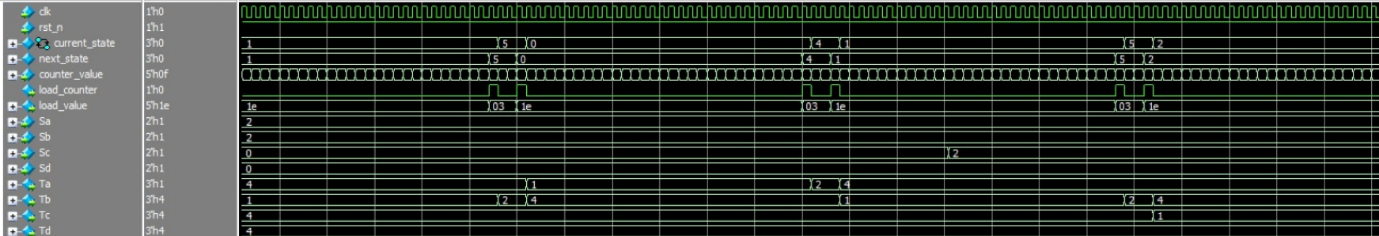
* + 1. **Round robin (Equal traffic):**

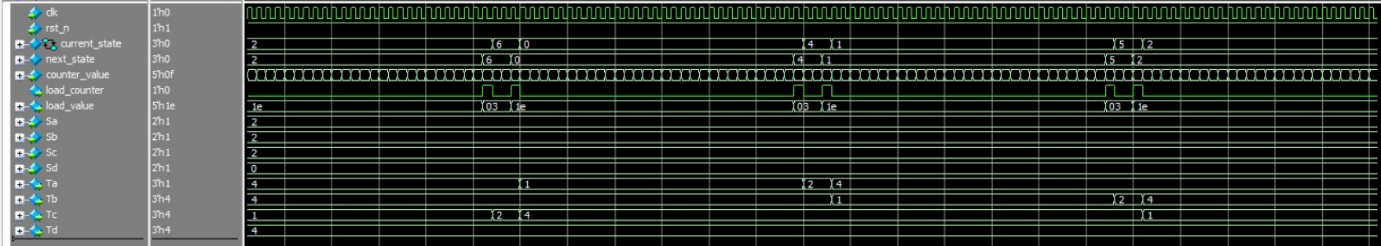
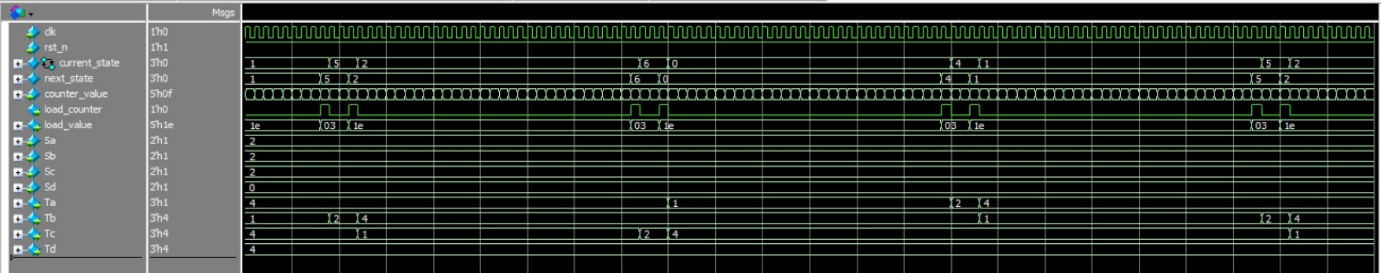




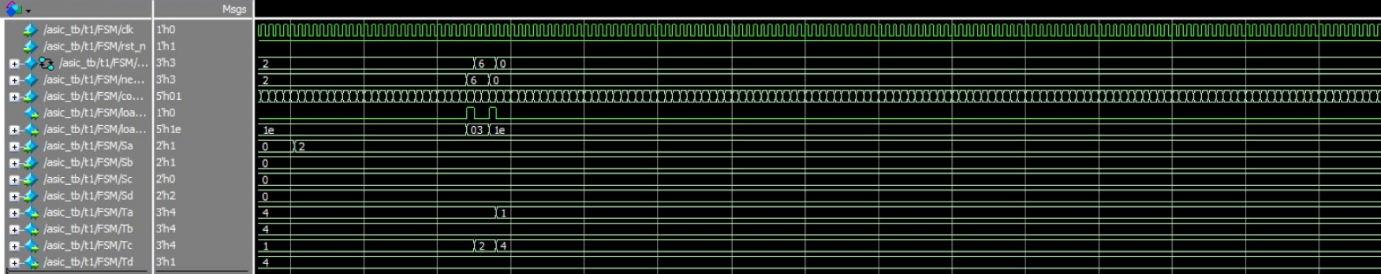
* + 1. **Round robin (Equal traffic between a&b):**



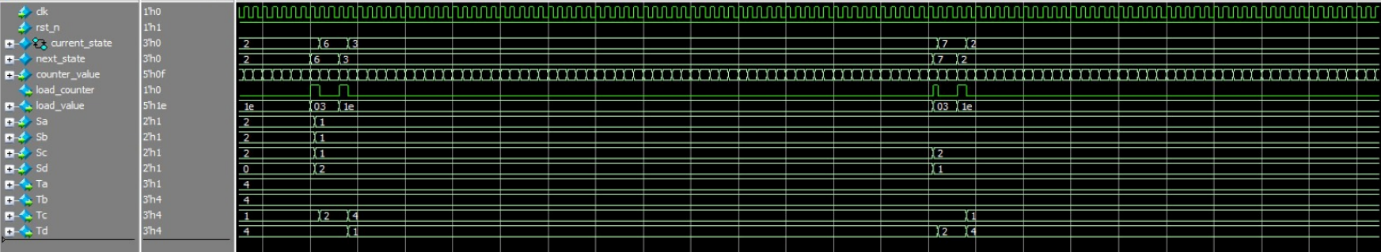
* + 1. **Round robin (Equal traffic between a&b&c):**



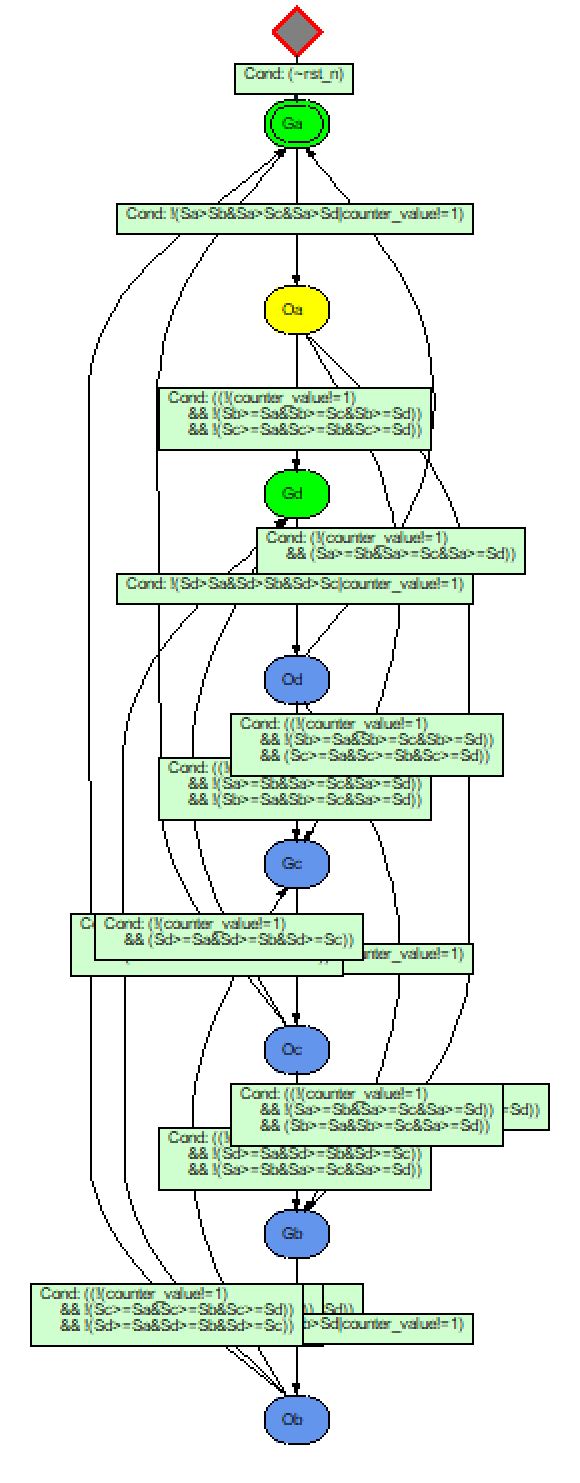
* + 1. **Stay at A:**



* + 1. **Priority given to d, stay at c:**



* **FSM diagram using Questasim:**



* **Contribution Table:**

|  |  |
| --- | --- |
| Task | Name |
| FSM Diagram structuring | Fares , Youssef Osama , Ahmed, Youssef Hany |
| Design RTL coding | Fares , Youssef Osama |
| Test strategy, Test bench | Ahmed , Youssef Hany |

Snippets results with identification for each test case on wave form and transcript also i need you to use fsm debug mode and attach a snippet for fsm from tool with your handwrititten one to ensure no difference

Hint : you can run vsim with attribute -fsmdebug to show the fsm in questasim for more info refer to the user guide of the tool

For sure , snippets for your code and test bench also i need your work load for each one